INTEGRATING MACHINE LEARNING AND OPTIMIZATION TECHNIQUES FOR SHORT-TERM MANAGEMENT OF SHARED E-SCOOTERS UNDER DEMAND UNCERTAINTY

A dissertation submitted to the Division of Engineering and Policy for Sustainable Environment, Graduate School of Engineering, Hokkaido University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Engineering

By
NARITH SAUM

Examination Committee
Associate Professor Sugiura Satoshi
Professor Uchida Ken-etsu
Professor Toru Hagiwara
Associate Professor Takahashi Sho

Division of Engineering and Policy for Sustainable Environment
Graduate School of Engineering
Hokkaido University
Sapporo, Japan

September 2023

INTEGRATING MACHINE LEARNING AND OPTIMIZATION TECHNIQUES FOR SHORT-TERM MANAGEMENT OF SHARED E-SCOOTERS UNDER DEMAND UNCERTAINTY

Division of Engineering and Policy for Sustainable Environment
Graduate School of Engineering
Hokkaido University
Sapporo, Japan

September 2023

ABSTRACT

Shared mobility has proliferated in global cities as an innovative transportation mode enhancing urban mobility and as a potential solution to address first- and last-mile problems. Recently, a new emerging shared transportation, dockless electric scooters (e-scooters), has gained popularity worldwide for their specific advantages, including environmentally friendly, time and cost-saving mode, congestion, parking constraint, and satisfied riding experience. Besides these advantages, this shared mode has several disadvantages, including volatile demand, excessive or starving stations, short service life, costly maintenance, battery recharging, and distribution regulations. As a new transportation mode, there are limited studies about shared e-scooters, especially related to daily operations. Therefore, this study aims to develop an efficient framework for better managing this dockless shared service by taking advantage of open-source historical ridership data, machine learning, and deep learning methods. This study thus is separated into three main sections as follows.

From the literature review, shared e-scooters are mainly used for recreational or tourism activities, which differs from shared bikes. These trip purposes with dockless policy led to high demand volatility while requiring a higher service level. To deal with the heteroscedasticity (i.e., non-constant variation) of the demand, both demand and variance prediction models are developed using deep learning (Recurrent Neural Networks) and Autoregressive Conditional Heteroskedasticity (ARCH), respectively. Moreover, Box Cox transformation was also employed to remove the heteroscedasticity. Based on numerical results from three real-world datasets (Austin TX, Minneapolis MN, and Thammasat TH), machine learning and deep learning achieved higher prediction accuracy than conventional regression models, SARIMAX. Box Cox transformation can improve the prediction accuracy, especially MAE by around 5.36%, while the supply planning with this transformation is very efficient for lower service levels. Nevertheless, the application of this transformation technique in supply planning for higher service levels exhibits decreased efficiency due to its exponential conversion characteristic, thereby revealing a weakness of Box Cox transformation. In this case, the supply planning model with original data and predicted variance by SGARCH achieves lower oversupply. At 95% served demand, accounting for heteroscedasticity in supply planning could reduce the oversupply by 26.22%.

Even machine learning and deep learning models can outperform conventional statistical models; their performance strongly depends on the choice of hyperparameters, while optimizing these hyperparameters is usually computationally expensive. To deal with this problem (i.e., Hyperparameter Optimization or HPO), this study proposed a novel algorithm, Iterative Decision Tree (IDT), which employs a Decision Tree regressor based on the Classification and Regression Tree (CART) algorithm as the surrogate function. Our algorithm suggests several new candidates per iteration as random or extreme points from a few best-performed leaves. This characteristic allows IDT to be trained in parallel, which solves the main disadvantage of previous sequential model-based algorithms (ex., Bayesian Optimization). To evaluate the performance of IDT, it was employed to optimize several benchmark problems, including nonconvex functions and HPO of machine learning and deep learning models. As a result, IDT showed very effective performance for both computational time and objective value compared to benchmark algorithms.

Based on the above results, a new framework for short-term rebalancing planning was proposed for the unique characteristic of shared e-scooters, including volatile and heteroscedastic demand, recharging the battery, and faulty e-scooters. Monte Carlo simulation based on the predicted trip gaps and standard deviations was employed to generate the stochastic demand scenarios. The framework was examined based on e-scooter data from Minneapolis MN, while *k*-means clustering algorithm was employed to aggregate the trip generation and attraction for the total clusters of 15, 30, and 60. For this data-driven stochastic optimization problem, two separated formulations were constructed and solved by the Integer Linear Programming (ILP) and the Hybrid of Ant Colony Optimization with ILP (ACO-ILP). Under limited computational time, ILP solver is efficient for solving small-size problems (15 and 30-cluster problems), but the Hybrid approach is more efficient for large-size problems (60-cluster problems). Based on the numerical result of the most practical case (60-cluster problems), our data-driven framework for rebalancing planning for shared e-scooters could reduce the expected objective value by around 13.27% and 16.68% compared to historical weekly and daily data.

In summary, dockless shared e-scooters require proper operational planning to minimize their negative impacts, so that this shared mode can become a potential solution for compacted urban mobility. This objective can be achieved through the proposed data-driven framework, which integrates machine learning and optimization techniques to minimize the demand uncertainty and driving distance for the rebalancing vehicle. For instance, start-of-art prediction models with hyperparameter optimization can effectively handle the volatile demand of shared e-scooters, while rebalancing optimization planning can be addressed through the exact approach (ILP solver) or the heuristic algorithm (ACO-ILP).

Keywords: Autoregressive Conditional Heteroskedasticity, Deep Learning, Demand Uncertainty, Heuristic Optimization Algorithm, Hybrid Optimization Algorithm, Hyperparameter Optimization, Integer Linear Programming, Machine Learning, Shared Electric Scooters, and Shared Service Rebalancing.

ACKNOWLEDGEMENTS

Behind the accomplishment of this research, numerous individuals deserve my sincere gratitude. Firstly, I would like to thank AUN/SEED-Net for the scholarship and the opportunity to pursue my Ph.D. Degree at two prestigious universities, Sirindhorn International Institute of Technology (SIIT) in Thailand and Hokkaido University (HU) in Japan. This experience has been truly transformative, as it allowed me to collaborate with professionals, acquire specialized skills, and gain invaluable life lessons during my time abroad.

Secondly, I would like to express my deep gratitude to my advisors, Assoc. Prof. Dr. Mongkut Piantanakulchai and Assoc. Prof. Dr. Satoshi Sugiura, for their invaluable guidance, constructive comments, and insightful advice that played a pivotal role in ensuring that my research yielded timely results for graduation. Additionally, I am sincerely grateful for the presence and active participation of Assoc. Prof. Dr. Kriengsak Panuwatwanich, Assoc. Prof. Dr. Chawalit Jeenanunta, and Dr. Warut Pannakkong, who graciously dedicated their valuable time to serve as members of my thesis committee during my proposal examination and final defense. Their constructive feedback and valuable suggestions have significantly contributed to enhancing the quality of my research. Furthermore, I would like to extend my gratitude to Prof. Dr. Uchida Ken-etsu, Prof. Dr. Toru Hagiwara and Assoc. Prof. Dr. Sho Takahashi for their unwavering support and valuable recommendations throughout my research journey at HU. Their expertise and guidance have been instrumental in shaping my research trajectory.

I would like to extend my deepest gratitude to all the staff members from both universities and the AUN/SEED-Net program. Their kindness, assistance, and unwavering support throughout my study are beyond measure. Their invaluable contributions have played a significant role in making my academic journey a smooth and fulfilling one. Lastly, I cannot express enough gratitude to all my family, friends, and colleagues. Their continuous motivation, assistance, and guidance have been immeasurable. I am truly grateful for their unwavering support and encouragement throughout my academic endeavors.

Narith Saum

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols / Abbreviations	Terms
ACO	Ant Colony Optimization
AE	Autoencoder
ANNs	Artificial Neural Networks
ARCH	Autoregressive Conditional Heteroscedasticity
ARIMA	Autoregressive Integrated Moving Average
BO	Bayesian Optimization
CNNs	Convolutional Neural Networks
DT	Decision Tree
E-Scooter	Electric Scooter
GA	Genetic Algorithm
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GB	Gradient Boosting
GNNs	Graph Neural Networks
GRU	Gated Recurrent Units
GS	Grid Search
HPO	Hyperparameter Optimization
IDT	Iterative Decision Tree
ILP	Integer Linear Programming
LSTM NNs	Long-Short Term Memory Neural Networks
RF	Random Forest
RNNs	Recurrent Neural Networks
RS	Random Search
SBRP	Static Bike Rebalancing Problem
SIIT	Sirindhorn International Institute of Technology
SVM	Support Vector Machine
TPE	Tree structured Parzen Estimator
TU	Thammasat University
XGBoost	Extreme Gradient Boosting

CHAPTER 1
INTRODUCTION

1.1 Overview

Interest in active and shared service transportation is growing as a result of urban congestion, technological advancement, and environmental concerns. Due to this, shared mobility has become increasingly popular in big cities worldwide as a cutting-edge mode of transportation that improves urban mobility and as a potential remedy for the issue of first- and last-mile connectivity with public transport [1]. Bike sharing, vehicle sharing, ride-sourcing, and, more recently, shared electric (e-) scooters are all examples of sharing service modes [2]. While fixed routes, driver availability, and vehicle scheduling frequently provide restrictions on public transportation, shared micromobility (bike, e-bike, and e-scooter) is a time- and money-efficient feeder. Bridging the existing transportation network gap can also expand the public transit service area. The idea of sharing services for transportation originates in economic models that date back to the 1990s and are based on peer-to-peer sharing or cooperative consumption of resources. The factors facilitating this sharing service among strangers include online social network platforms, online payment, and global positioning systems (GPS) enabled mobile technology.

In Santa Monica, California, shared dockless e-scooters were first deployed in September 2017 after Bird Rides Inc., a micromobility firm, scattered thousands of e-scooters throughout the city. Because of their affordability, comfort, and ease of use, these scooters quickly gained popularity among users [3]. The term “micromobility” refers to a short-range trip that is too far to walk and too short to drive, especially the first-/last-mile problems. A year later, this unicorn operator could reach 10 million rides with more than 2 million unique riders and operated in more than 100 cities. In 2018, the total number of dockless e-scooter trips in the US was 38.5 million, while those of station-based and dockless bikes were 36.5 million and 9 million, respectively [4]. And shared scooter ridership doubled (86 million trips) in 2019 [5]. By May 2019, more than 65 dockless e-scooter operators were providing services in more than 150 cities and 40 universities in more than 35 countries worldwide. The general steps for using shared e-scooters are as follows: download a phone application and online registration, log in to find the nearby devices, scan the QR code to unlock the scooter, enjoy your trip by e-scooter, park the scooter at the appropriate parking place, scan QR code to finish the trip and online payment (see Figure 1.1). The trip fare is calculated as an unlock fee of 1 USD plus 0.15 USD/minute.

Figure 1.1 General step of using shared dockless e-scooter

Most people (70%) viewed electric scooters positively, including expanding transportation options, a car-free lifestyle, convenience for short trips, complementing public transit, convenience for female users, and increasing vehicle equitability specifically for the low-income community [6, 7]. Several aspects of trip satisfaction of dockless e-scooter were evaluated, such as trip satisfaction (88%), satisfaction with scooter availability (85%), ease of sign-up (85%), ease of parking (82%), cost satisfaction (81%), fun to ride (75%), and positive impact on the environment (66%) [8]. Moreover, the trip purposes of dockless e-scooter are joyriding (34%), running errands (23%), commuting (19%), visiting someone (13%), and work break/lunch (9%). Since the trip fare of shared e-scooters is relatively higher than shared bikes, people do not use e-scooter for commuting but for other leisure or tourism activities. Based on MOVO scooter, the e-scooter is powered by Lithium-Ion batteries, so it produces CO_2 only 4.6g per person per kilometer compared to 190g and 120g for automobiles and motorcycles, respectively.

On the other hand, there were also some negative impacts of shared dockless e-scooters, such as accidents, conflict with the pedestrian, littering on the sidewalk or public/private spaces, vandalism, thief, battery explosion, frame defect, and ineffective distribution leading to the crowded or starved area. However, stricter regulation and more effective training measures could improve some of these problems, accidents, and pedestrian conflicts. And new technology and design could reduce the risk of battery explosion and frame defects. Another important issue of shared e-scooters is the emission from operations, including distribution, rebalancing, and charging collection [9]. Three common strategies for recharging e-scooters are paying freelance chargers (e-scooter juicers), battery swapping, and collecting low-battery e-scooters to nearby charging stations. Some operators pay freelance chargers to collect, charge and redistribute the low-battery e-scooters, but this strategy struggles with ineffective collection methods (too many chargers, longer collection distance, polluted collecting vehicles) and explosion accidents (due to unqualified facilities and inexperienced chargers). The battery swapping seems to be a good choice, but replacing each scooter takes a long time, challenging to combine with rebalancing routing. The last strategy is more common in practice as the operators collect the low-battery e-scooters to charge at nearby stations before redistribution again.

Unlike other transportation modes, shared e-scooters are preferable only for short-length trips, less than 3km, unless they will not be cost and time effective anymore. Moreover, the demand is highly volatile due to the nature of trip duration and the primary trip purposes, leisure and tourism activities. As a dockless mode, the spatiotemporal patterns are difficult to predict and cluster. This leads to the problem of high operational costs and complicated rebalancing with many criteria, including excessive or shortage, low battery, and defective e-scooters. Precise demand prediction and effective rebalancing are necessary to stay competitive among operators and reduce the negative impacts of this environmentally friendly mode. To address the operational challenges associated with shared dockless e-scooters, we examined various robust prediction models and employed them to forecast the spatiotemporal demand and supply level of these shared e-scooters. Chapter 2 and Chapter 3 of this study reviewed numerous machine learning and deep learning models proposed or utilized to predict challenging time series problems, such as the stock market, electricity demand, traffic demand, etc. It is important to note that deep learning is a subset of machine learning. However, these terms are commonly

(as well as in this study) utilized to refer to different groups of prediction models. For instance, deep learning encompasses prediction models with artificial neurons, such as ANNs, AE, CNNs, GNNs, RNNs, and others. On the other hand, machine learning models typically refer to different prediction algorithms, such as DT, K-Nearest Neighbors (KNN), RF, SVM, XGBoost, etc. In general, deep learning exhibits high configurational adaptability owing to its architecture, consisting of multiple layers, with each layer comprising a varying number of nodes or computational mechanisms aimed at extracting relevant features from the data. Consequently, deep learning often outperforms machine learning models in several aspects due to its inherent capacity to automatically learn hierarchical representations from data. These advantages include superior feature extraction, the ability to handle high-dimensional and complex data, capture non-linear correlations, support end-to-end learning, and demonstrate scalability. Furthermore, we reviewed and utilized several optimization algorithms to address the rebalancing challenges in e-scooter sharing, aiming to reduce operational costs, unsatisfied demands, and emissions associated with the rebalancing process.

1.2 Research gaps

This study aims to improve the short-term operational planning for shared e-scooters, which have several challenging characteristics such as trip characteristics (short-range trips), trip purposes, physical characteristics (short-service life and recharging the battery), regulations, and emissions from rebalancing and distribution. The general characteristics of shared e-scooters are reviewed in **Chapter 2**, which covers the history of e-scooter evolution, the adoption of e-scooters in sharing services, social perception towards this shared micromobility, impacts on urban mobility, accidents, policy regulations, environmental life cycle assessments, and operational planning. Understanding these characteristics enables effective management of the operational planning of shared e-scooters, minimizing negative impacts and maximizing positive ones. Furthermore, subsequent chapters examine operational planning approaches in similar shared services, including demand prediction models, rebalancing optimization planning, and optimization algorithms. Then several research gaps were discovered as follows:

- Many robust prediction models have been proposed to forecast transportation demand, specifically for shared bikes and e-scooters. However, most of these models have primarily focused on accuracy performance, neglecting the presence of heteroscedasticity (or non-constant variation) in transportation demand. As a result, the valuable information in historical data has not been effectively utilized. This research gap has raised two key questions: "Which approaches can be employed to address the issue of heteroscedasticity in shared e-scooter demand?" and "How can accounting for heteroscedasticity benefit operational planning?"
- The optimization of hyperparameters highly influences the performance of machine learning and deep learning models. Sequential-based algorithms, such as Bayesian Optimization (BO) and Tree of Parzen Estimators (TPE), are well-suited for expensive problems like hyperparameter optimization (HPO) of deep learning models. However, their implementation in parallel computing can be challenging. On the other hand, population-based algorithms are suitable for inexpensive problems. They can be trained in parallel but

cannot retain the historical evaluation points and only communicate within the current population. Therefore, there is a need for algorithms that bridge the gap between these two approaches, the sequential-based and population-based approaches. These algorithms should be well-suited for optimizing problems that fall into neither cheap nor expensive categories, such as HPO problems.

- Although machine learning and deep learning models can achieve state-of-the-art prediction performance, the distribution or rebalancing based solely on these predicted demand values often leads to a service level of only 50%. Conversely, some prior studies have addressed demand uncertainty by assuming that the demand follows specific distributions, such as the Poisson distribution. Consequently, there is a need for rebalancing frameworks designed explicitly for shared e-scooters that effectively minimize and account for the demand uncertainty.

1.3 Research objectives

Based on the above operational challenges for shared dockless e-scooters and the research gaps, this thesis aims to review their general background and to improve the operational management of this new transportation mode using historical ridership data and other related information. Therefore, the objectives of this study could be summarized as follows:

- Review the general characteristics of dockless e-scooter, including the history of the scooter, history of shared dockless e-scooter, regulations, environmental impact, and impacts on urban mobility.
- Propose and evaluate the framework for supply planning for shared e-scooters based on forecasted spatiotemporal demand and variance using deep learning and autoregressive conditional heteroscedasticity, respectively.
- Propose and evaluate a new hyperparameter tuning algorithm, Iterative Decision Tree (IDT), which is suitable for hyperparameter optimization of machine learning and deep learning models.
- Propose and evaluate the data-driven framework for short-term (ex., a few hours) rebalancing planning for shared e-scooters with new mathematical formulations combining three important characteristics of this sharing service: demand uncertainty, low battery, and faulty e-scooters.

1.4 Main contributions

The findings from this research, in response to the aforementioned research gaps and objectives, contribute to practical and academic implications. Firstly, this study comprehensively reviews shared e-scooters, including their development history, adoption in shared services, social perception, trip characteristics, accidents, policy regulations, environmental life cycle assessments, and operational planning challenges. These insightful perspectives enable decision-makers, regulators, and operators to formulate appropriate strategic policies to minimize negative impacts and maximize positive impacts.

Secondly, this study proposes several approaches to address heteroscedasticity in transportation demand, particularly in shared e-scooter demand. By employing machine learning models for demand prediction and utilizing data transformation techniques such as

Box Cox and variance prediction using ARCH, efficient estimation for supply planning at low and high service levels is achieved. The effectiveness of these approaches was evaluated using three different real-world datasets based on the proposed oversupply metric.

Thirdly, the proposed algorithms, IDT, demonstrate their effectiveness in searching for near-global optimal solutions within limited computational time. In practical applications, IDT reduces training time compared to sequential-based algorithms by parallel training. Moreover, the historically evaluated points are still utilized to update its surrogate function, decision tree regression, ensuring no loss of information like population-based algorithms.

Lastly, the optimization of static rebalancing planning was conducted on stochastic demand generated through Monte Carlo simulation with predicted demand and variance. To enhance practicality, this rebalancing optimization problem can be solved using either an exact algorithm (ILP GLPK solver) or a hybrid algorithm (ILP-ACO), depending on computational time constraints. Based on numerical results, our approach effectively reduces demand uncertainty through demand and variance prediction, resulting in shorter driving distances for the rebalancing vehicle and lower operational costs compared to baseline approaches that rely on historical daily or weekly data. The proposed framework is applicable and customizable for specific practical cases, ex., preferred service level and adjustable safety stock.

Some parts of this thesis were already published in international conferences, while others were (and will be) published in international journals. Those international conferences and journals are as follows:

- Saum, N., & Piantanakulchai, M. (2019). A Review on an Emerging New Mode of Transport: The Shared Dockless Electric Scooter. *In Proceedings of 13th International Conference of Eastern Asia Society for Transportation Studies (EASTS 2019), 9-12 September 2019, Colombo, Sri Lanka.*
- Saum, N., Sugiura, S., & Piantanakulchai, M. (2020). Short-Term Demand and Volatility Prediction of Shared Micro-Mobility: a case study of e-scooter in Thammasat University. *In Proceedings of Forum on Integrated and Sustainable Transportation Systems (FORUM ISTS 2020), 3-5 November 2020, Delft, The Netherlands.* https://doi.org/10.1109/FISTS46898.2020.9264852.
- Saum, N., Sugiura, S., & Piantanakulchai, M. (2022). Hyperparameter Optimization Using Iterative Decision Tree (IDT), *IEEE Access, vol. 10, pp. 106812-106827,* https://doi.org/10.1109/ACCESS.2022.3212387.
- Saum, N., Piantanakulchai, M., & Sugiura, S., "Supply Level Planning for Shared E-Scooters Considering Spatiotemporal Heteroscedastic Demand", *Transportation Research Interdisciplinary Perspectives*. (**Under Review**).
- Saum, N., Sugiura, S., & Piantanakulchai, M., "Optimizing Shared E-Scooter Operations under Demand Uncertainty: A Framework integrating Machine Learning and Optimization Techniques", *IEEE Access*. (**Under Review**).

1.5 Thesis outline

To accomplish the research objectives outlined above, the thesis was divided into four main chapters, with an additional chapter dedicated to summarizing all findings, recommendations, and suggestions for future studies. **Figure 1.2** illustrates the interconnection between these four

main chapters of the thesis. **Chapter 2** delves into the history and previous studies of shared dockless e-scooters, highlighting several significant research gaps, operational challenges, and relevant characteristics in the short-term operational planning of shared e-scooters. One research gap identified in **Chapter 2** pertains to supply level estimation, particularly for heteroscedastic datasets. **Chapter 3** builds upon this by extensively reviewing demand and variation prediction models and exploring various approaches to determine the most efficient supply level estimation models. Our investigation into demand prediction models in **Chapter 3** revealed that the choice of hyperparameters significantly influences the performance of machine learning and deep learning models. Consequently, in **Chapter 4**, we expanded our literature review on hyperparameter optimization (HPO) algorithms and proposed two novel algorithms, IDT-E and IDT-R. The insights gleaned from **Chapters 2**, **3**, and **4** were then applied to enhance the efficiency of rebalancing planning for shared dockless e-scooters in **Chapter 5**. The details in each chapter of this thesis are summarized in short as follows:

- **Chapter 1**: provides a general overview of the research, including the emergence of shared dockless e-scooter, its related operational problems, research gaps, research objectives, research contributions, and thesis organization.
- **Chapter 2**: provides a comprehensive review of the history and previous studies related to shared dockless e-scooters. This review chapter is structured into eight sections, including the history of electric scooters, the introduction of dockless shared e-scooters, social perception of shared e-scooters, trip characteristics and their impacts on urban mobility, accidents, policy regulations, environmental life cycle assessments, and short-term operational planning.
- **Chapter 3**: presents a detailed description of the proposed framework for designing short-term supply planning for shared e-scooters. The chapter begins with a general introduction and proceeds with a literature review of different approaches employed in supply planning management, such as demand prediction models, variance prediction models, and data transformation techniques. Subsequently, the supply planning framework and relevant formulations are developed. The effectiveness of this framework is evaluated using three different real-world datasets of shared e-scooter operating in Austin TX, Minneapolis MN, and Thammasat University TH. Finally, the discussions and conclusions are made based on the numerical results.
- **Chapter 4**: offers a general background on hyperparameter optimization problems and provides a comprehensive review of various techniques used to optimize hyperparameters. The hyperparameter optimization for recurrent neural network architectures (i.e., RNNs, LSTM NNs, and GRUs) is also present in this chapter. Additionally, we introduce the proposed algorithm, Iterative Decision Tree (IDT), and present numerical results comparing IDT to several baseline hyperparameter optimization (HPO) algorithms, namely Grid Search (GS), Random Search (RS), Tree-structured Parzen Estimator (TPE), Genetic Algorithm (GA), Bayesian Optimization (BO) with Gaussian process, and BO with Random Forest. The comparison is based on several benchmark problems, including nonconvex functions, HPO of machine learning models, HPO of deep learning models, and HPO of shared e-scooter demand prediction by RF and GRUs.

- **Chapter 5**: discusses the relevant challenges associated with the operational management of shared e-scooters. Additionally, we present a literature review of various approaches used in the operational planning of sharing services, explicitly focusing on bike sharing. Drawing from the insights gained from the previous chapters and the literature review in this section, a data-driven framework is developed for the short-term static rebalancing of shared e-scooters. The rebalancing problems, considering demand uncertainty, are formulated as integer linear programming, whereas an ILP solver "GLPK" and a hybrid algorithm "ACO-ILP" are employed to optimize 30 random instances for each cluster scenario (15, 30, and 60) grouping by the *k*-means algorithm. The effectiveness of the proposed framework is evaluated using the open data from Minneapolis MN.
- **Chapter 6**: the conclusions and discussions of the research findings are given. Based on these findings, several directions for future study are suggested.

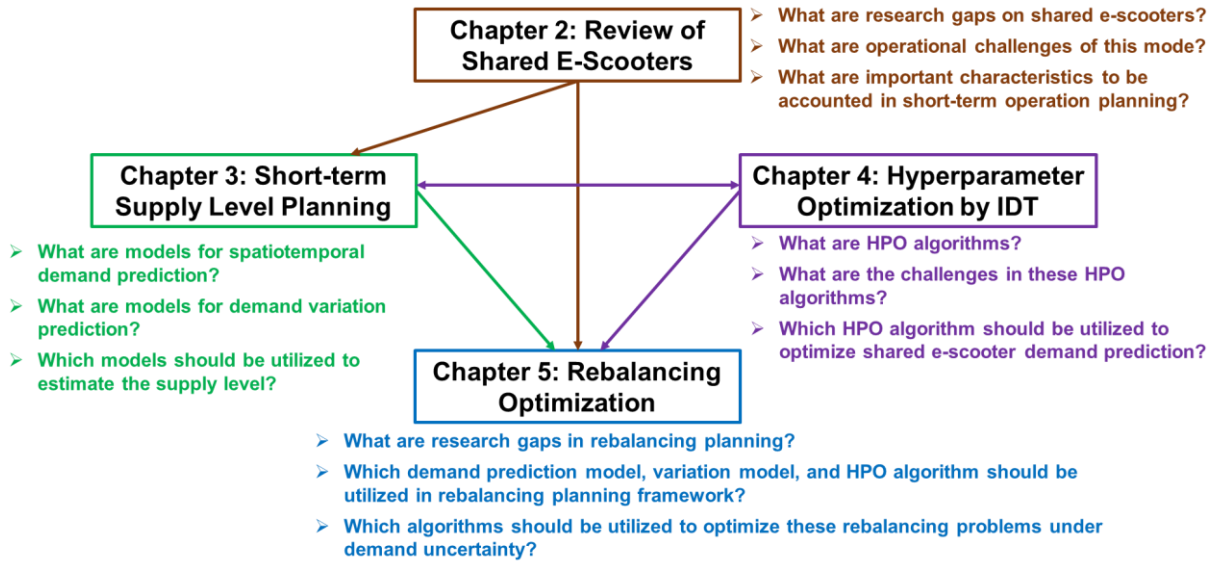

Figure 1.2 Relationship between thesis chapters

References

[1] S. Shaheen and N. Chan, "Mobility and the sharing economy: Potential to facilitate the first-and last-mile public transit connections," *Built Environment,* vol. 42, no. 4, pp. 573-588, 2016, doi: https://doi.org/10.7922/G2862DN3.

[2] C. S. Smith and J. P. Schwieterman, "E-scooter scenarios: evaluating the potential mobility benefits of shared dockless scooters in Chicago," Chaddick Institute for Metropolitan Development, Depaul University. Dec 2018.

[3] T. K. Trivedi *et al.*, "Injuries associated with standing electric scooter use," *JAMA network open,* vol. 2, no. 1, pp. e187381-e187381, 2019.

[4] NACTO. *Shared Micromobility in the U.S.: 2018*. Available: https://nacto.org/shared-micromobility-2018/#:~:text=In%202018%2C%20people%20took%2036.5,handful%20of%20cities%20in%202018.

[5] NACTO. *Shared Micromobility in the U.S.: 2019*. Available: https://nacto.org/shared-micromobility-2019/

[6] R. Clewlow, F. Foti and T. Shepard-Ohta, "Measuring Equitable Access to New Mobility: A Case Study of Shared Bikes and Electric Scooters," POPULUS 2018, Available: https://trid.trb.org/view/1576769.

[7] R. R. Clewlow, "The Micro-Mobility Revolution: The Introduction and Adoption of Electric Scooters in the United States," POPULUS 2018, Available: https://trid.trb.org/view/1528426.

[8] M. Toll. (2018). *The results are in and Americans are loving electric scooter share programs*. Available: https://electrek.co/2018/08/14/americans-love-electric-scooter-shares/

[9] M. Chester, "The Electric Scooter Fallacy: Just Because They're Electric Doesn't Mean They're Green," ed, 2018.

CHAPTER 2
A REVIEW OF SHARED E-SCOOTERS

2.1 History of electric scooter

Scooter, derived from “scoot” which means fast movement, represents an entertainment product sliding on land, water, ice, and children’s toy skateboard car [1]. Like other transportation modes, electrification was also adapted to the scooter, called electric or e-scooter, as soon as 1991 by Honda intending to replace gasoline-powered scooters rooted from 1902. Currently, the word “Scooter” is given to various transportation modes such as self-balance, motorized, motor scooters, and mobility scooters (see **Figure 2.1**).

	Name Power range Speed range Running range	**Self Balance mono-wheels Scooter** 350 - 1800 W 10 - 45 km/h 15 - 90 km		Name Power range Speed range Running range	**Motor Scooter (Mini Size)** 250 - 2000 W 20 - 45 km/h 40 - 60 km
	Name Power range Speed range Running range	**Hoverboard or** **Self Balance two-wheels Scooter** 200 - 1000 W 12 - 15 km/h 15 - 25 km		Name Power range Speed range Running range	**Motorcycle or** **Motor Scooter (Large Size)** > 2000 W > 50 km/h 40 - 150 km
	Name Power range Speed range Running range	**Motorized Scooter or** **Electric Kick Scooter** 200 - 1300 W 20 - 60 km/h 10 - 120 km		Name Power range Speed range Running range	**3-Wheels Mobility Scooter** 180 - 1000 W 6 - 28 km/h 17 - 60 km
	Name Power range Speed range Running range	**Two wheel scooter with handle** or **Segway** 1000 - 4000 W 18 - 30 km/h 25 - 60 km		Name Power range Speed range Running range	**4-Wheels Mobility Scooter** 180 - 1200 W 6 - 28 km/h 17 - 60 km

Figure 2.1 Types of electric scooters

A motor scooter is a sort of motorcycle with a step-through chassis and a platform for the rider's feet; well-known models include Vespa and Lambretta. A motorized scooter is a powered stand-up scooter employing a small utility gas or electric engine. Due to their low or zero emissions, motor scooters are becoming increasingly popular in China, Taiwan, and Europe [2]. In this instance, countries in the Asia-Pacific area with a large motorbike population, such as Taiwan, China, Vietnam, Indonesia, and Thailand, anticipate burgeoning demand for electric motor scooters. Electric self-balancing scooters are becoming increasingly popular because of their affordability, lightweight, fashionable appearance, and off-road potential. Additionally, since the development of dockless sharing services in the past several years, standing electric scooters (such as Segways and motorized scooters) have been highly sought-after. Mobility e-scooters are another mode that has helped older people live better lives by allowing them to participate in social activities like shopping, running errands, or going to the doctor [3]. Last but not least, motorized scooters have been increasingly popular in recent years thanks to dockless services. They first became popular for short trips in densely populated urban areas in the 2000s.

At the moment, shared services provide three different types of scooters: Segways, motor scooters, and motorized scooters. Since the motorized or electric kick scooter is the newest growing mode and has a high acceptance rate, we concentrated on it following the thesis's purview. **Figure 2.2** only depicts the most significant developments in the electric scooter (e-scooter), which resembles the motorized scooters used today for shared transportation. The first scooters were simple children's toys made from a soapbox, some scraps of board, and an old pair of roller skates. Then it evolved into a commercial product and was made available for kid's sports. For miniskirt riders like Autoped, ABC Skootomota, and Austro Motorette, the powered scooter was created specifically for them between the 1910s and 1920s [4, 5]. Such a scooter was later converted into a motor scooter, most notably the iconic Japanese Rabbit and Italian Vespa [1]. The "Kick-n-Go," a scooter powered by a pedal on a lever, was created by the Honda Corporation in 1974. Although it still needed as much work as a standard scooter, kids loved this inventive scooter. Before the rise of bicycles, kids may benefit from using steel scooters with two little bicycle wheels, which were popular among dog scooters. In 1996, Wim Ouboter, the inventor of the micromobility system that addresses the first-/last-mile problem (i.e., the distance is too short to drive but too far to walk), created a stylish foldable aluminum scooter that was a very portable and lightweight mode of transportation. In 1999, this design was sold to Razor and unveiled in Tokyo, where it quickly caught on as a fashion trend. With the Go-Ped brand, one of the first and most well-known producers of motorized scooters, Patmont Motor Werks began operations in 1985 and debuted its gasoline and electric scooters in 2001 and 2003, respectively [6].

Figure 2.2 Evolution of motorized or electric kick scooters (e-scooters)

Electric vehicles' propulsion systems can be divided into four categories: fuel-cell electric vehicles, plug-in hybrid electric vehicles, hybrid electric vehicles, and battery electric vehicles (FCEVs). HEVs have both a gasoline engine and a battery, but the gasoline engine or regenerative braking generates the battery energy and cannot be replenished by the electrical grid. A BEV relies solely on its battery for power, but a PHEV uses both its battery and an internal combustion engine. Another form of electric vehicle is an FCEV, which runs on hydrogen and oxygen. Even though it is still under development, it is regarded as the most environmentally friendly electric car because it only emits water [7]. However, the e-scooter used for shared mobility only has a built-in or swappable battery that the power grid can

recharge. The top 5 brands of motorized scooters now on the market are Razor, Segway-Ninebot, Xiaomi, Swagtron, and EcoReco. As shown in **Figure 2.1**, the power, speed, and charging range of the contemporary electric kick scooter are 200–1300 W, 20–60 km/h, and 10–120 km, respectively. According to information on the item on the Alibaba website, the battery is mostly made of lithium, while some producers also use LG or Samsung batteries. Additionally, these batteries have a warranty of one to two years and charge for three to eight hours. They also have a recharging cycle of about 300 (in some cases, more than 900). The majority of the scooter's frame is made of carbon fiber, aluminum alloy, or steel. Depending on the battery, frame type, and brand, a motorized scooter can cost between 50 to 700 USD.

2.2 Introduction of shared dockless e-scooters

In recent years, the growth of micromobility enterprises has been extensively reported. Customers can choose from easy first-mile/last-mile transportation options thanks to businesses in China like Ofo and Mobike, and in the US like Citi Bike and Jump Bike. The shared dockless e-scooters developed by Lime and Bird in the US in 2018 re-energized the micromobility trend. Compared to shared bikes at 13 percent in 8 years and shared vehicles at 16 percent in 18 years, the adoption rate of shared e-scooters in major US cities reached 3.6 percent in less than a year [8]. In the first seven months and the first 14 months, respectively, one million and six million e-scooter rides were completed, according to Lime data [9]. The demand for e-scooters has increased as e-scooter-sharing services become more widely used in nations like the US, France, Germany, Spain, Singapore, and Thailand. Electric scooters are purchased by companies like Bird, Lime, Spin, Jump, Razors, and Neuron, which provide these sharing services, primarily from well-known manufacturers.

Figure 2.3 displays the month and year of the initial rollout of shared scooters across all operators and nations. To address the last-mile issue in smart cities, university campuses, and other major workplaces, Samocat's founders developed the smart payment platform for station-based kick-scooter sharing. For this innovative concept, Samocat earned numerous national and international awards. In August 2015, they began testing their rental kick scooter in Russia before quickly expanding to other European nations. However, this startup company could not garner much social attention because of the inconvenience of station-based mode and kicking weariness. The first dockless e-scooter sharing service from Telepod was introduced in August 2016 following several months of testing in Singapore. However, the rigorous regulations, constrained permitted space, high prices, graffiti, and thieves prevented this new enterprise from becoming more well-known. A year later, two more operators, named Neuron and Popscoot, joined this sharing service. It wasn't until September 2017 that a large dockless e-scooter company, Bird, received approval to deploy their e-scooters in California. After that, dockless scooters gained popularity and spread to other places. After Bird's success, many months later, another major dockless e-scooter supplier, Lime, followed by Spin, Skip, BlueDuk, and Goat, began rolling out their e-scooters across numerous US states.

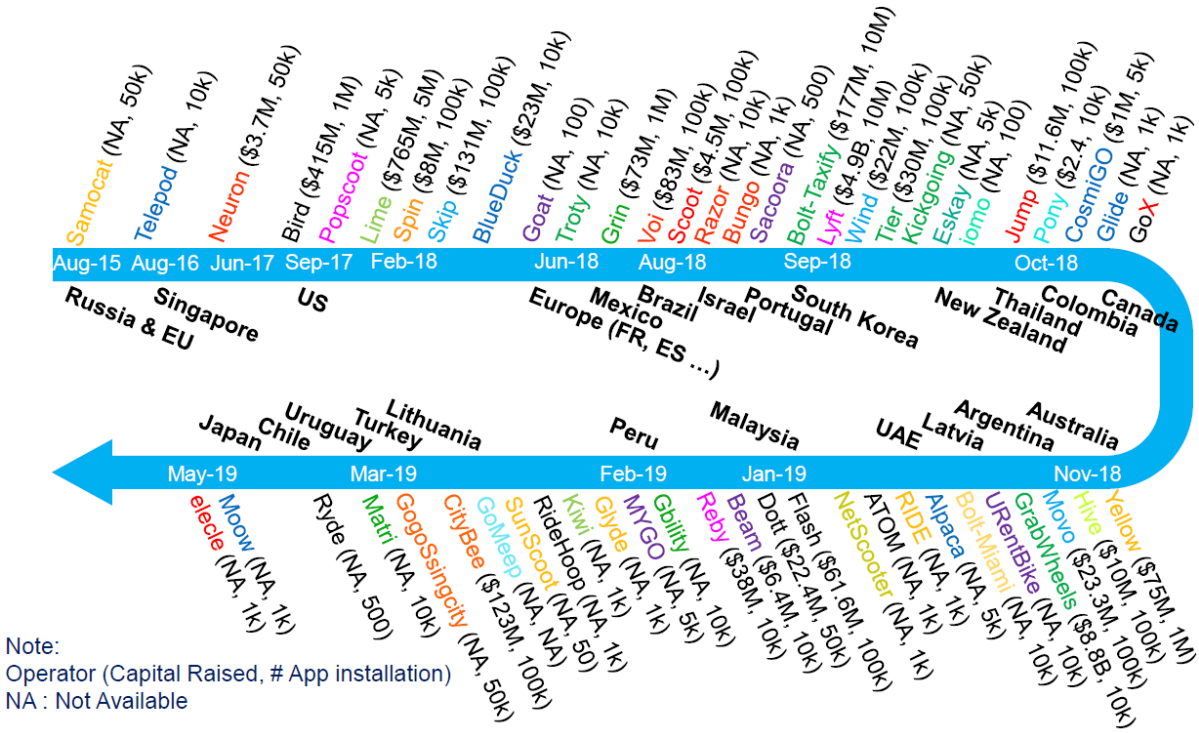

Figure 2.3 Deployment month and year of shared scooters by countries and operators with the capital raised and number of phone app installations (Source: operator's Facebook & Instagram, Crunchbase website, and Google play store)

The local operators like Troty, Grin, and Voi have introduced shared e-scooters to certain European nations (France and Spain), Brazil, and Mexico, in addition to the US-based operators Bird and Lime. After the shared e-scooters' outstanding success for the whole year, many additional nations started to acknowledge this mode's influence on urban mobility and started testing it. Many unicorn businesses, like Lyft, Grab, and Bolt-Taxify, entered the dockless e-scooter era after spotting the potential opportunity. Due to the significant demand in Asian nations, Neuron Mobility expanded its operations outside of Singapore to Thailand in October 2018, Malaysia in the early months of 2019, and Australia in the following few months. Kickgoing scooter made the first e-scooter deployment in South Korea in September 2018, and four additional operators (Gbility, GogoSsingcity, Ryde, and elecle) may begin making e-scooter deployments in the first few months of 2019. By the end of May 2019, more than 150 cities and 40 institutions were served by the dockless e-scooter fleets provided by about 60 companies.

According to **Table 2.1**, European users pay the highest rate of $6.27 for a 30-minute trip, followed by users in Israel, the United States, and Mexico, who pay $5.6, $5.5, and $5.2, respectively. A 30-minute e-scooter trip in ASEAN nations costs between $3 and $3.52. While the price for a Lime E-Bike and E-Scooter is the same, it is slightly more expensive than the prices for shared bikes (conventional bikes), such as the Jump-Bike ($2 for 30 minutes) and Ford Gobike ($3 for 30 minutes). Dockless electric scooters have a recharge range of 24 to 60 kilometers and can travel at a top speed of 23 to 48 kilometers per hour. It is recommended to use e-scooters with faster speeds since the fee is determined by the amount of time used [10].

For a 3.5 km trip, traveling at an average speed of 10 and 15 km/h saves 7 minutes and $1, respectively.

Table 2.1 Standard fee of dockless shared e-scooters in each region

Region	Standard Fee		USD Equivalent	
(e-scooter operator)	Unlock Fee	Riding Fee	Unlock Fee	Riding Fee
US (Bird)	$1	0.15 $/min	$1.00	0.150 $/min
Europe (Lime)	€1	0.15 €/min	$1.14	0.171 $/min
Mexico (Lime)	MEX $10	MEX 3 $/min	$0.52	0.156 $/min
Israel (Bird)	NIS 5	0.5 NIS/min	$1.40	0.140 $/min
Singapore (Neuron)	SGD 1	SGD 0.12 $/min	$0.74	0.089 $/min
Thailand (Neuron)	฿20	3 ฿/min	$0.64	0.096 $/min
Malaysia (Neuron)	RM 3	0.3 RM/min	$0.75	0.075 $/min

Note: exchange rate based on April 2019

2.3 Social perception on shared e-scooters

Two early studies by the research team in POPULUS published in 2018 discussed the potential of shared dockless e-scooters. The first report, published in July 2018, surveyed over 7,000 people in major US cities [8]. Their results found that more than 70% of respondents view this mode positively, including expanding transportation options, car-free lift style, convenient replacement for personal vehicles or ride-hailing, and a complement of public transit. Moreover, it could reduce the active transportation gender gap, and improve vehicle equitability, especially for low-income communities [8, 11]. Another questionnaire survey in 2018 by Qualtrics based on 500 adults across the US showed that dockless shared e-scooter was agreed to be a lasting innovation transportation mode (55%), particularly the experienced riders for 72% [12]. The majority of riders showed high satisfaction with this mode, including trip satisfaction (88%), satisfaction with scooter availability (85%), ease of sign-up (85%), ease of parking (82%), cost satisfaction (81%), fun to ride (75%), and positive impact on the environment (66%). Shared e-scooters were used for the purposes of joyriding 34%, running errands 23%, commuting 19%, visiting someone 13%, and work break/lunch 9%. For this reason, 19% of respondents still prefer using e-scooters even if they cost more than other modes, while 13% chose not to use e-scooter again as it is either unsafe or inconvenient. This survey also found that 75% agreed with the positive impact on air pollution by more e-scooter usage, but only 17% believed that this mode could deal with congestion.

In addition, Fitt and Curl (2019) conducted a questionnaire survey to understand the perception of users and non-users in several cities in New Zealand. 71% of the respondents experienced shared e-scooters, while 75% used them more than once. The main reason for trying e-scooters for the first time is to try e-scooters and have fun. And they, mostly younger people, men, and full-time employers, will likely use it again for commuting, social engagement, and the supermarket. With the availability of e-scooter, 58% of the trips come from active mode, 23% come from private or shared vehicles, and 11% would not be made without e-scooters [13]. Another study by the author employed a social practice approach based on an online qualitative survey in four large cities in New Zealand [14]. This study aimed to

explore the early changes in the materials, competencies, and meanings associated with urban mobility and the disruptive potential of these changes for urban transport and broader social relations. Sixty online surveys from the student at the University of Minnesota were done to assess the correlation between e-scooter usage and five prominent personalities [15]. And they found that students with higher extraversion scores were significantly more likely to use e-scooters than those with lower scores.

Furthermore, the mode choice model was developed based on a stated preference survey to understand the factors and potential shift from carsharing to e-scooter sharing of young users in Munich [16]. The Technology Acceptance Model was extended to identify the factors that affect the intention to continue using e-scooters based on survey data in Chicago [17]. Two important salient factors determining users' decisions are perceived usefulness and perceived reliability (i.e., availability in time and space, particularly for mandatory trips). At the same time, other critical drivers are social influence, perceived ease of use, variety seeking, and perceived enjoyment. In Turkey, an Online questionnaire was conducted to understand the predictors influencing behavioral intention toward shared e-scooter [18]. As a result, behavioral intention is significantly affected by social influence, effort expectancy, performance expectancy, and price sensitivity, but environmental awareness and hedonic motivation.

2.4 Trip characteristics and impacts on urban mobility

One of the most popular topics about shared e-scooters is the spatiotemporal trip characteristics and their impacts on urban mobility. Smith and Schwieterman (2018) evaluated the potential of shared dockless e-scooters on urban mobility based on several scenarios. They found that dockless e-scooters could be time- and cost-effective for short-range trips, less than 3 km, compared to the private automobile. Therefore, this mode could increase the non-auto trip options from 45% to 75% and increase job reachability by 16% [10]. And the new results based on multimodal transportation accessibility analysis in Chicago, US, found that dockless e-scooter possibly reduces travel time by 24% - 29% compared to walking and public transit (around 3 to 5 minutes) and increases job reachability by 12.3% and over 20% for 30-minutes and 60-minutes public transit commuting trip respectively [19]. McKenzie published two papers comparing the spatiotemporal usage pattern of e-scooter to dock-based bikes and ride-hailing [20, 21]. Based on the traffic condition in Washington, the author found that e-scooters are faster than ride-hailing during rush hour & traffic congestion. His results also show a clear difference in both temporal and spatial patterns between dockless e-scooters and capital bikes. Some temporal similarity was found between e-scooter and casual bike usage, but still significantly different spatial distribution. The station-based shared bikes are mainly used for commuting, while the shared e-scooters (casual bike users) are used for other purposes such as leisure, recreation, or tourism. Likewise, Younes, Zou, Wu, and Baiocchi (2020) employed negative binomial regression to compare the temporal characteristics of dockless e-scooter and station-based bikes in Washington. They also confirmed the dissimilarity between dockless e-scooter and member capital bike users [22]. They also found that member bike-sharing tends to be the least sensitive to changing weather conditions due to habitual travel behavior, the less expensive pricing structure, or not having an alternative mode. In addition, dockless e-scooters

are not statistically sensitive to precipitation, probably because of the ease of trip ending. Lastly, all micromobility users are susceptible to changing gas prices, especially dockless e-scooters.

In Singapore, Zhu, Zhang, Kondor, Santi, and Ratti (2020) also compared the spatiotemporal pattern between dockless e-scooters and bike-sharing. E-scooters have spatially compact and quantitatively denser distribution, while their high demand is associated with attractive places such as metros and dormitories. In addition, scooter sharing has a better performance than bike-sharing in terms of the increased sharing frequency and decreased fleet size, but dockless e-scooter requires high maintenance cost for rebalancing and charging. Rainfall and high temperatures at noon suppress the usage but non-conclusively [23]. Dockless e-scooters and e-bikes were compared using the Austin (TX) data from December 2018 to May 2019 [24]. E-bikes (3.23 m/s) are faster than e-scooters (2.49 m/s). However, both modes were ridden slower for recreational purposes than for commuting. The riding speed of these two modes was similar over days of the week but different over hours of the day.

Furthermore, Jiao and Bai (2020) employed spatial analysis and negative binomial regression to examine the spatial pattern based on the open-sourced data in Austin, Texas. Higher e-scooter trips are associated with high population density, higher education, compact landuse, closer distance to the city center, better connectivity, and transit station. However, ridership surprisingly negatively correlates with the proportion of the young population [25]. Comparably, the e-scooter departure and arrival trips in Austin are positively associated with the proportion of residential, commercial, education, and industrial area [26]. And based on the Spatial Durbin model, the morning departures are associated with residential landuse but not educational landuse, and vice-versa. Positive correlations were also found with bike facilities, bus stops, and employment density, but the parameter of median income. Once again, Bai and Jiao (2020) used the data from Austin (TX), and Minneapolis (MN) to examine the spatial usage pattern against urban environment [27]. They found different usage patterns in the Minneapolis dataset based on spatial analysis and negative binomial regression. Ridership of dockless e-scooter in Minneapolis had a positive correlation with household income but negative correlations with the industrial area, open space and parks, and transportation facilities. The temporal pattern in these two cities is also different, i.e., afternoon and weekend preference for Austin, but evening preference for Minneapolis. In addition to factors found in previous studies, holidays and special events was found to be significantly increased the e-scooter but not bike sharing [28]. H. Li, Yuan, Novack, Huang, and Zipf [29] uncovered 100 proxy trip purposes of shared e-scooter in Washington, D.C., by spatiotemporal topic modeling method based on OD data, POI, landuse, and landcover.

2.5 Related accidents

There were many reports regarding shared e-scooter-related accidents, attracting more research interest. The reasons of these accidents could be:

- Insufficient regulations (ex., helmet) and training, especially for new riders.
- Small wheels, lightweight, and standing characteristics make it susceptible to hazardous road surfaces, especially during nighttime, including potholes, speed bumps, curb ramps, uphill and downhill streets, etc.

- Unreliable or unstandardized materials cause battery explosions, malfunctions, defects of frame and handlebar, brake failure, etc.

Based on 324 posts, just 6.17 percent included a person wearing protective gear, and 6.79 percent featured protective gear in some other way [30]. In Southern California, 249 e-scooter-related patients from two metropolitan emergency rooms were gathered in order to study the injury features and typical usage patterns of dockless e-scooters [31]. 228 riders and 21 non-rider pedestrians made up the data that was gathered between September 2017 and August 2018. These patients are, on average, 34 years old, and 58 percent are men. Falls (80%), collisions with objects (11%), and being struck by moving vehicles (9%) are the most frequent incidents. The accident rates were highest from 3 pm to 11 pm (57%) and from 7 am to 3 pm (26%) and 11 pm to 7 am (17%), respectively. In addition, three risky behaviors—not wearing a helmet (95 percent), tandem riding (8 percent), and disobeying traffic laws (9 percent)—were noted. Despite the fact that the leasing agreement required them to be at least 18 years old, the investigation also discovered that about 11% of the patients were under 18. Although it is forbidden to ride on the sidewalk, 26% of them did so, nonetheless. Only 4.4 percent of riders reported wearing a helmet, even though it is mandatory by law in California. As a result, head injuries—which account for 40% of cases—are the most frequent, followed by fractures/cuts and sprains/bruises, which account for 32% and 28% of cases, respectively. 30 percent of the 149 patients had to stay in the hospital for more than 4 hours, and two had to be admitted to intensive care units.

The electronic medical records from two emergency rooms in Utah, the US, are used by Badeau et al. (2019) to quantify and describe the injuries caused by dockless e-scooters. Fifty data points from June 15 to November 15 of 2018, were used in this analysis, as opposed to 8 data points from June 15 to November 15 of 2017. They divided the injuries into three categories based on the data from 2018: major head injury (8%), major musculoskeletal injury (36%), and minor injury (56%) [32]. A research team from California and New York also constructed the anatomic distribution of e-scooter injuries using the motorized scooter-related injuries from the US National Electric Injury Surveillance System (NEISS) [33]. Between 2013 and 2017, data on 32,400 injuries were obtained, and between 2016 and 2017, the number of injuries rose by 77%. On the other hand, Bresler et al. (2019) also studied craniofacial injuries using motorized scooter-related injuries from NEISS. They received 990 cases between 2008 and 2017, with the most common injuries being to the head (62%), face (24%), mouth (7%), neck (6%), air (1%), and eye (1%) [34].

Medical records of 90 patients related to e-scooter from the emergency department in Dallas (TX) were used to investigate craniofacial injuries [35]. The results found that 62% were male, 58% of craniofacial injuries were severed, and 18% were related to alcohol. 13 electronic medical records of The George Washington University Hospital revealed the increasing number of Severe injuries of skull fracture, central cord syndrome, and vertebral compression fracture, raising the awareness on the issues of safety and public health related to e-scooter [36]. Similarly, 169 e-scooter crashes in news reports across the US from 2017 to 2019 were collected to construct a crash dataset [37]. Overall, there was a growing trend for the reported e-scooter-involved crashes, while 73% and 50% of victims are male and 18 to 40 years old, respectively. 50% of the cases happened at night, and the three main locations of crashes are street/arterials (50%), intersections (25%), and sidewalks (15%). The collision types are hit-vehicle (65%) and

fall-off (25%), resulting in fatalities (30%) and severe injuries (40%). A detailed study of the collision between pedestrians and e-scooters was used to highlight the safety risks and the incidence to help shape public policy to ensure the safety of both riders and pedestrians [38]. In New Zealand, two pieces of research used accident records from the emergency department [39, 40]. The result from Mayhew and Bergin (2019) showed that 57% of victims are male, primarily European ethnics (57%), and between 20 to 40 years old (65%). Moreover, 25% of cases need surgery for extremities (84%) and head/face (44%). Several recent studies are related to e-scooter accidents in Sweden [41], factors contributing to the number of e-scooter injury accidents in Austin TX [42], early fatalities associated with shared e-scooters in the US [43], and association of scooter-related injury and hospitalization in the US [44].

2.6 Policy regulations

As a new vehicle introduced to the shared service, the regulations for this mode may vary from one country to another or even city, and from time to time. Therefore, the regulations could be split into: regulations for e-scooter applying to personal or shared vehicles and regulations for shared service providers. The regulations for e-scooter are vehicle size (width, length, height, and weight), power capacity, fire safety, speed, helmet, driving license or certificate, rider age, license plate, riding path (footpath is allowed or not), etc. For operators, the regulations include a business license, vehicle registration, distribution plan, response plan to abandoned/damaged/improper parking, educating rider, commercial general liability insurance, clean-hand certificate, etc. The violations could be fined, the vehicle seized, or jailed.

Anderson-Hall, Bordenkircher, O'Neil, and Scott (2019) assessed the operators' inventory, device specification, and regulation in many important cities in the US. Their results could be used to draw the lesson learned from trial and error of this mode and regulation changes in these cities [45]. Similarly, regulation, equity policy, guidelines, and pilot programs from 61 municipalities in the US were used to explore the best practice municipal e-scooter policy [46] or the later version [47]. They found that 59% of these cities have either fleet or operator caps, and the pilot program was implemented in only 54% of the studied cities, so the majority (70%) have an equity policy. In Rosslyn of Virginia (US), 181 surveys of riders and non-riders were used to analyze the safety perception of pedestrians towards the presence of e-scooters and the experience of sidewalks blocked by e-scooters [48]. On average, around half of them feel uncomfortable with e-scooters, especially non-users (76% for unsafe and very unsafe, and 75% for often and always-frequency of sidewalk blocked). However, only 16% of 606 observed e-scooters were not appropriately parked, and 6% blocked pedestrian right-of-way. Moreover, 3666 parking practices in 5 cities in the US were used to analyze the parking violation frequency of bikes, e-scooters, and motor vehicles [49]. However, they found that motor vehicles, especially ride-hailing and food delivery services, impede access far more (24.7%) than bikes (0.3%) and e-scooters (1.7%).

In Sweden, Gössling (2020) used the content analysis of 173 reports from local media, including printed media, radio websites, and TV, to assess the concerns before and after the introduction of shared e-scooters. As a result, they suggest the urban planner propose several necessary regulations regarding this mode, such as maximum speed, mandatory bicycle infrastructure use, dedicated parking, and the number of operators [50]. In Vienna (Austria),

weekly data from six operation geofences and no-parking zones were recorded to examine the spatial analyses [51]. They discovered that the geofence of each operator is dynamically changed to maximize the ridership under the fleet size regulation. Moreover, they recommend that the public sector establish incentives to ensure outlying and/or transit-poor neighborhoods, increase the fleet cap if the operator expands their geofence, and reach the goals of the city's pilot regarding safety, equity, and the sustainability of the scheme. In France, ethnomethodology and multimodal conversation analysis were employed to examine the conflict with pedestrians based on the video record from three e-scooter riders on Paris's street [52]. Several recent studies on policy regulation are economic regulation of e-scooter in the US [53], parking regulations across 37 cities in the US [54], municipal guidelines among e-scooter use from 150+ cities in the US [55], e-scooter regulations in Bergen, Norway [56], and visual attention on the shared road between pedestrian, cyclist, and e-scooter [57].

2.7 Environmental life cycle assessment

Since dockless e-scooters are powered by battery recharging from the electric grid, it is considered a sustainable transportation mode, just like walking and cycling. The question is, "Is this dockless sharing mode really green?". Energy analyst Matt Chester wrote an essay about the emissions from dockless e-scooters (2018) in Washington, DC. Based on three popular e-scooter models (Ecoreco S5, Ninebot, and Swagtron), their relative CO2 emissions per kilometer are 5.6, 4.7, and 2.5 grams (using DC electric grid emission rates of 0.622 grams per watt-hour). Only 1% to 2% of the CO2 emissions from driving a typical US automobile at the same distance are attributed to riding an e-scooter. Additionally, he considered three scenarios for the emission from shared e-scooters while carrying them to and from places where they may be charged (20 caped by Bird, and competitive between charging contractor limit to fewer). 3.22 km, 8.1 km, and 16.1 km were employed in the analysis because there was no information on the distance of collecting e-scooters. The most effective scenario for recharging journeys is an e-scooter making five trips per day at a distance of 2.4 kilometers, which results in only 2 percent of car emissions, whereas the least efficient case accounts for 28% of the car emissions and the medium-efficient case accounts for 8% of them. Shared e-scooters produce more overall CO2 emissions than vehicles in the least-efficient scenario, which occurs if less than 28% of scooter journeys displaced car trips [58].

Moreau et al. (2020) conducted the life cycle assessment of electric scooters under the modal split of Brussels (Belgium) and life span sensitivity. At the base case of 7.5 months life cycle, dockless e-scooter produces CO_2 of 131g per person per kilometer, but this could be reduced to 91g, 51g, and 40g for one year, 2.5 years and five years life cycle, respectively, compared to 110g for the modal share replaced by dockless e-scooter [59]. However, personal e-scooters produce only 67g per person per kilometer, which is better than dockless e-scooters because of their shorter life span, improper usage, vandalism, and rebalancing. In this case, shared e-scooters require 9.5 months lifespan to become green mobility in the current situation. In Germany, scenario analysis about the emission of shared e-scooters was examined based on the condition of Berlin and Bochum. In Berlin, the authors analyze the life cycle assessment considering several conditions, including longer lifespan, swappable battery, solar power, and the possibility of transporting from the manufacturer by plane. For the base case (2 years

lifespan, swappable battery, and battery swapping and broken devices collection by diesel-van), dockless e-scooters emit CO_2 of 73 g per person per kilometer, and the emission could be up to 235 g if the lifespan is just six months [60]. These authors also used the modal share scenarios to evaluate the potential emission, parking space, and traffic space demand of dockless e-scooter in Bochum. The results show that dockless e-scooters could improve urban mobility and emission if they could replace individual motorized transport and especially become intermodal mobility services with public transit [61].

In addition, the Consequential LCA (CLCA) was proposed to assess the environmental impacts of urban mobility disruption by free-floating e-scooter in Paris [62]. The result found that shared e-scooter generated an extra 13 ktCO_2eq in one year under the assumption of one million users mainly shifting from lower-emitting modes (active mode, metro, and mass rapid transit). From scenario analysis, the suggestions for improving the carbon footprint from this mode are the increased lifetime mileage and choice of servicing (collecting e-scooters for recharging or battery swapping). Moreover, de Bortoli [63] later employed an integrated modal LCA to assess Paris's three private and shared micromobility vehicles, including bikes, second-generation e-scooters, and e-mopeds. The highlights of this study found that the ownership does not contribute to the emission but the vehicle lifetime mileage, while the emission ranking of this shared micromobility is between the personal ICE modes and the active modes (including the public railing system). Similarly, using Minneapolis as the case study, Peng, Nishiyama, and Sezaki [64] assessed the emission reduction from shared micromobility (shared bike and scooter). The Monte Carlo simulation showed that 60% of shared micromobility trips likely came from personal vehicle and public transit trips resulting in GHG emission reduction of 126.4 to 151.3 tons (about 0.012% of the total on-road emission).

2.8 Short-term operational planning

Regarding operational-related research, we can separate the previous studies into two main parts, short-term demand prediction and operational planning optimization. Several demand prediction models, including statistical models, machine learning, and deep learning, were proposed or employed to predict the spatiotemporal demand for shared e-scooters. On the other hand, previous studies examined the operational planning for shared e-scooter, including fleet size optimization, deterministic or stochastic rebalancing, collecting e-scooter for recharging, charging station design, and facility planning.

He and Shin [65] proposed a novel spatiotemporal graph capsule neural network called GCScoot, to predict the spatiotemporal shared e-scooters' trip flow in three different cities Austin TX, Louisville KY, and Minneapolis MN. The improved architecture, GCScoot2, was later proposed and evaluated on one additional dataset in Chicago IL [66]. As a result, they got huge accuracy improvement compared to baseline models, but it also requires lots of topological information along with long training time. Recurrent Neural Network (RNN) was employed to predict the temporal latent features from Convolutional Autoencoder, called encoder-recurrent neural network–decoder (ERD) framework [67]. The Convolutional Autoencoder works as dimensional reduction, especially the sparse data. This ERD framework showed enhanced performance compared to the baseline Long Short-Term Memory (LSTM) Neural Network based on shared e-scooter data operating in Gwangjin district, Seoul, South

Korea. On the other hand, Phithakkitnukooon, Patanukhom, and Demissie [68] dealt with demand sparsity of the grid e-scooter ridership in Calgary (Canada) by using a mask model or region of interest (ROI) to guide the fully convolutional network, called Masked Fully Convolutional Network (MFCN). Spatio-Temporal Multi-Graph Transformer (STMGT) is a graph convolutional network based on adjacency, functional similarity, demographic similarity, and transportation supply similarity graphs was proposed to forecast the hourly shared e-scooter demand in Austin TX and Washington DC [69]. Kim Sujae, Choo, Lee, and Kim Sanghun [70] employed LSTM model to predict the hourly demand in the grids grouped into several clusters by community structure method. This framework was evaluated on one-month data of e-scooter operating in Seocho and Gangnam districts of Seoul, South Korea. Moreover, the bagging ensemble approach based on XGBoost, RF, and Extra Tree (ET) was employed to predict the clustered daily demands of shared e-scooter deploying on Jeju Island, South Korea [71].

On the other hand, there were several studies about the operational planning for dockless shared e-scooter. Masoud et al. [72] constructed the mathematical formulation to minimize total charging collection distance and solved it by several optimization algorithms in allocating freelance e-scooter-chargers. The simulated assignment problems were solved by the adaptive College Admission Algorithm (ACA) in comparison to the MILP solver and Black Hole Optimizer (BHO) algorithm. The open data in Minneapolis MN, and Louisville was employed to construct the data-driven demand model based on Poisson processes (temporal) and Kernel Density Estimation (spatial) for comparison of different electric scooter sharing design options, including the impact of fleet size and the cost of managing their charging [73]. Tolomei et al. [74] employed a deep learning model called 3D-CLoST to forecast the shared e-scooter demand, then greedily assigned workers to relocate the surplus and shortage using data from Austin TX, and Louisville KY, as the case study. Osorio, Lei, and Ouyang [75] formulated the overnight rebalancing of shared e-scooter accounting for the possibility of charging on the vehicle as the MIP and solved the large instance by the discrete-continuous hybrid model for integrating the line-haul and local operations. This proposed framework was evaluated based on simulated demand following the normal distribution. Two-stage stochastic programming was proposed for long-term planning (i.e., investment cost on charging facilities, fleet size, and relocation schedule) and short-term planning (i.e., minimizing relocation cost, charging cost and penalty of unserved demand) [76]. Multi-agent deep reinforcement learning, called ESB-DQN, was proposed to optimize the rebalancing operation and battery swap by encouraging customers (incentive) to pick up the e-scooter in the nearby regions [77]. Finally, Altintasi and Yalcinkaya [78] employed GIS-based multi-criteria to optimize charging station locations to integrate this mode with the existing public facilities, points of interest, and population density. The proposed model was examined based on the data in Karsiyaka, Izmir, Turkey.

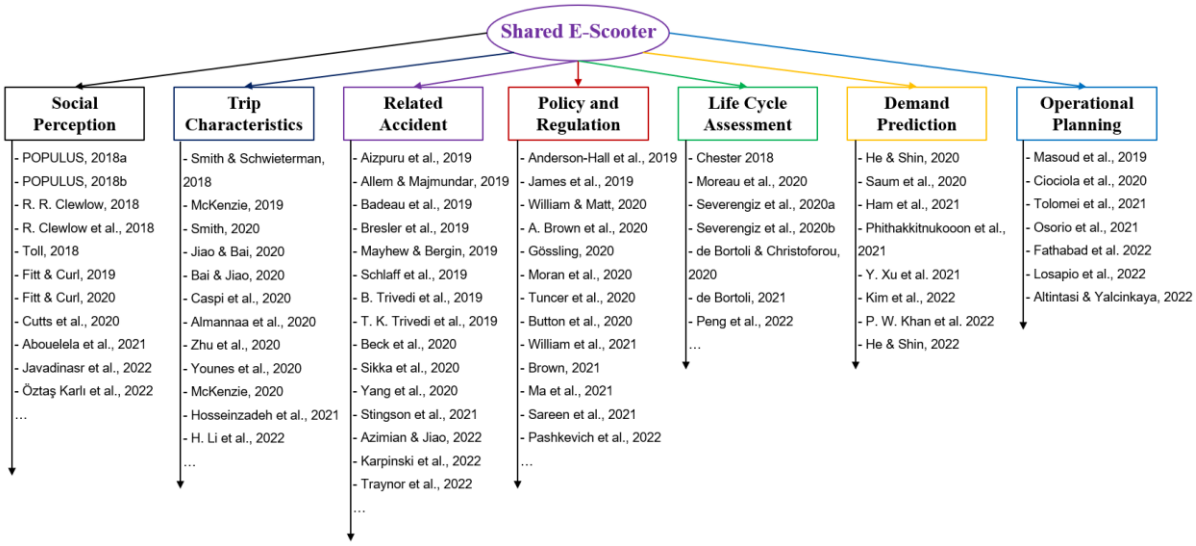

Figure 2.4 Summary of previous studies about shared dockless e-scooters

2.9 Discussion and conclusion

Even though the primary sources of the energy grid continue to be emission sources like coal and natural gas, electrification is now one of the best ways to reduce emissions from vehicle traffic. The use of shared micromobility, particularly e-scooters and e-bikes, has been crucial in addressing urban transportation problems like congestion, mobility, emissions, and parking shortages. These shared services aim to address the first- and last-mile issues in densely populated urban regions. In this situation, dockless e-scooters are gaining popularity and have been embraced in many cities across the globe. Lime and Bird are the two most well-known companies that offer shared e-scooters, and their markets are worth up to a billion dollars. Because they are time and money efficient for short-distance trips between 0.8 and 3.5 km, shared e-scooters are becoming increasingly popular. The simplicity of finding/parking, the lack of restrictions, the adaptability of the routes, and the enjoyable experience are further appealing elements. The unlock cost ($0.52 - $1.40) and the riding price (0.075 - 0.17 $/minute) for shared e-scooters vary from region to region. As mentioned above, dockless e-scooters have many impacts on urban mobility, such as vehicle equitability, job reachability, reduced vehicle trips, safe and convenience for females, and extended public transit catchment area. Additionally, most customers (88 percent) are happy with the trip, and more than 70% think this mode has a good impact on congestion and the environment.

With the rise in popularity of dockless e-scooters, many problems are also becoming more prevalent, including safety concerns, conflicts with pedestrians, street littering, and theft. The number of e-scooter incidents has also increased public awareness of scooter use or the introduction of dockless e-scooters in various locations, particularly in Europe. Numerous things, including unsafe riding practices, battery explosions, scooter flaws, and inadequate road infrastructure, contribute to these accidents. Furthermore, service providers cannot address the violations and instruct users on how to use e-scooters properly. As a result, ever-stricter laws and norms are put into place. These restrictions include speed limit, riding lane or area, parking infrastructure, helmet, improved phone application, payment method, and device requirements (weight, size, and fire safety standards).

In conclusion, dockless shared e-scooters present both advantages and disadvantages. Thus, it is essential for all stakeholders, including authorities, operators, and users, to collaborate and cooperate in order to minimize negative impacts and maximize positive impacts. As a newly emerging shared transportation mode, there is still limited research on the short-term operational planning of dockless shared e-scooters, which is the primary focus of this study. However, valuable insights can be gained from other sharing services, particularly bike sharing, which shares similarities with e-scooter sharing. As discussed earlier, shared e-scooters encounter more operational planning challenges compared to other sharing services, such as fluctuating demand due to short-range trips and unusual trip purposes, emissions resulting from inefficient rebalancing planning, and high maintenance costs due to vandalism, property vulnerability, and battery charging. Unlike most shared transportation modes that experience two peak demand periods during morning and evening rush hours, shared e-scooters face high demand throughout the day and evening, necessitating more frequent rebalancing efforts. Moreover, their operational planning must consider tasks like relocating low-battery e-scooters to charging stations and collecting broken e-scooters for repairs. The rebalancing planning for shared e-scooters should also take into account significant regulatory factors, including distribution regulations, registration fees per e-scooter, limitations on the number of e-scooters, and timely response to flooded areas with an excessive presence of e-scooters.

References

[1] M.-t. WANG and M. YOU, "The study of evolution of motor scooters," *Bulletin of Japanese Society for the Science of Design,* vol. 56, no. 2, pp. 23-32, 2009.

[2] K. Kendall, M. Kendall, B. Liang and Z. Liu, "Hydrogen vehicles in China: replacing the Western Model," *International Journal of Hydrogen Energy,* vol. 42, no. 51, pp. 30179-30185, 2017.

[3] D. Eck *et al.*, "Mobility assistance for older people," *Applied Bionics and Biomechanics,* vol. 9, no. 1, pp. 69-83, 2012.

[4] Goner, "A brief history of scooters," ed, 2018.

[5] Madcharge, "ELECTRIC SCOOTER: ORIGINS, HISTORY AND EVOLUTION," ed, 2018.

[6] Urbanscooters, "Go-Ped Scooters," ed, 2019.

[7] K. Ogura and M. L. Kolhe, "Battery technologies for electric vehicles," in *Electric Vehicles: Prospects and Challenges*: Elsevier, 2017, pp. 139-167.

[8] R. R. Clewlow, "The Micro-Mobility Revolution: The Introduction and Adoption of Electric Scooters in the United States," POPULUS 2018, Available: https://trid.trb.org/view/1528426.

[9] A. Adeyemi, "Electric Scooters And Micro-Mobility: Here's Everything You Need To Know," ed, 2019.

[10] C. S. Smith and J. P. Schwieterman, "E-scooter scenarios: evaluating the potential mobility benefits of shared dockless scooters in Chicago," Chaddick Institute for Metropolitan Development, Depaul University. 2018.

[11] R. Clewlow, F. Foti and T. Shepard-Ohta, "Measuring Equitable Access to New Mobility: A Case Study of Shared Bikes and Electric Scooters," POPULUS 2018, Available: https://trid.trb.org/view/1576769.

[12] M. Toll. (2018). *The results are in and Americans are loving electric scooter share programs*. Available: https://electrek.co/2018/08/14/americans-love-electric-scooter-shares/

[13] H. Fitt and A. Curl, "E-scooter use in New Zealand: Insights around some frequently asked questions," University of Canterbury, 2019, Available: https://ir.canterbury.ac.nz/handle/10092/16336.

[14] H. Fitt and A. Curl, "The early days of shared micromobility: A social practices approach," *Journal of Transport Geography,* vol. 86, pp. 102779, 2020, doi: https://doi.org/10.1016/j.jtrangeo.2020.102779.

[15] J. Cutts, M. Coleman and T. Vu, "Big Five Personality and E-Scooter Usage," *Undergraduate Journal of Psychology,* vol. 19, 2020.

[16] M. Abouelela, C. Al Haddad and C. Antoniou, "Are young users willing to shift from carsharing to scooter–sharing?," *Transportation Research Part D: Transport and Environment,* vol. 95, pp. 102821, 2021, doi: https://doi.org/10.1016/j.trd.2021.102821.

[17] M. Javadinasr, S. Asgharpour, E. Rahimi, P. Choobchian, A. K. Mohammadian and J. Auld, "Eliciting attitudinal factors affecting the continuance use of E-scooters: An empirical study in Chicago," *Transportation Research Part F: Traffic Psychology and Behaviour,* vol. 87, pp. 87-101, 2022, doi: https://doi.org/10.1016/j.trf.2022.03.019.

[18] R. G. Öztaş Karlı, H. Karlı and H. S. Çelikyay, "Investigating the acceptance of shared e-scooters: Empirical evidence from Turkey," *Case Studies on Transport Policy,* vol. 10, no. 2, pp. 1058-1068, 2022, doi: https://doi.org/10.1016/j.cstp.2022.03.018.

[19] C. S. Smith, "E-Scooter Mobility: Estimates of the Time-Savings and Accessibility Benefits Achieved via Chicago's 2019 E-Scooter Pilot Program," *Chaddick Institute Policy Series,* 2020.

[20] G. McKenzie, "Spatiotemporal comparative analysis of scooter-share and bike-share usage patterns in Washington, D.C," *Journal of Transport Geography,* vol. 78, pp. 19-28, 2019, doi: https://doi.org/10.1016/j.jtrangeo.2019.05.007.

[21] G. McKenzie, "Urban mobility in the sharing economy: A spatiotemporal comparison of shared mobility services," *Computers, Environment and Urban Systems,* vol. 79, pp. 101418, 2020, doi: https://doi.org/10.1016/j.compenvurbsys.2019.101418.

[22] H. Younes, Z. Zou, J. Wu and G. Baiocchi, "Comparing the Temporal Determinants of Dockless Scooter-share and Station-based Bike-share in Washington, D.C," *Transportation Research Part A: Policy and Practice,* vol. 134, pp. 308-320, 2020, doi: https://doi.org/10.1016/j.tra.2020.02.021.

[23] R. Zhu, X. Zhang, D. Kondor, P. Santi and C. Ratti, "Understanding spatio-temporal heterogeneity of bike-sharing and scooter-sharing mobility," *Computers, Environment and Urban Systems,* vol. 81, pp. 101483, 2020, doi: https://doi.org/10.1016/j.compenvurbsys.2020.101483.

[24] M. H. Almannaa, H. I. Ashqar, M. Elhenawy, M. Masoud, A. Rakotonirainy and H. Rakha, "A Comparative Analysis of E-Scooter and E-Bike Usage Patterns: Findings from the City of Austin, TX," *arXiv preprint arXiv:2006.04033,* 2020.

[25] J. Jiao and S. Bai, "Understanding the Shared E-scooter Travels in Austin, TX," *ISPRS International Journal of Geo-Information,* vol. 9, no. 2, pp. 135, 2020.

[26] O. Caspi, M. J. Smart and R. B. Noland, "Spatial associations of dockless shared e-scooter usage," *Transportation Research Part D: Transport and Environment,* vol. 86, p. 102396, 2020.

[27] S. Bai and J. Jiao, "Dockless E-scooter usage patterns and urban built Environments: A comparison study of Austin, TX, and Minneapolis, MN," *Travel Behaviour and Society,* vol. 20, pp. 264-272, 2020, doi: https://doi.org/10.1016/j.tbs.2020.04.005.

[28] A. Hosseinzadeh, A. Karimpour and R. Kluger, "Factors influencing shared micromobility services: An analysis of e-scooters and bikeshare," *Transportation Research Part D: Transport and Environment,* vol. 100, pp. 103047, 2021, doi: https://doi.org/10.1016/j.trd.2021.103047.

[29] H. Li, Z. Yuan, T. Novack, W. Huang and A. Zipf, "Understanding spatiotemporal trip purposes of urban micro-mobility from the lens of dockless e-scooter sharing," *Computers, Environment and Urban Systems,* vol. 96, pp. 101848, 2022, doi: https://doi.org/10.1016/j.compenvurbsys.2022.101848.

[30] J.-P. Allem and A. Majmundar, "Are electric scooters promoted on social media with safety in mind? A case study on Bird's Instagram," *Preventive Medicine Reports,* vol. 13, pp. 62-63, 2019.

[31] T. K. Trivedi *et al.*, "Injuries associated with standing electric scooter use," *JAMA network open,* vol. 2, no. 1, pp. e187381-e187381, 2019.

[32] A. Badeau, C. Carman, M. Newman, J. Steenblik, M. Carlson and T. Madsen, "Emergency department visits for electric scooter-related injuries after introduction of an urban rental program," *The American Journal of Emergency Medicine,* vol. 37, no. 8, pp. 1531-1533, 2019.

[33] M. Aizpuru, K. X. Farley, J. C. Rojas, R. S. Crawford, T. J. Moore and E. R. Wagner, "Motorized scooter injuries in the era of scooter-shares: A review of the national electronic surveillance system," *The American Journal of Emergency Medicine,* vol. 37, no. 6, pp. 1133-1138, 2019, doi: https://doi.org/10.1016/j.ajem.2019.03.049.

[34] A. Y. Bresler, C. Hanba, P. Svider, M. A. Carron, W. D. Hsueh and B. Paskhover, "Craniofacial injuries related to motorized scooter use: a rising epidemic," *American Journal of Otolaryngology,* vol. 40, no. 5, pp. 662-666, 2019.

[35] B. Trivedi, M. J. Kesterke, R. Bhattacharjee, W. Weber, K. Mynar and L. V. Reddy, "Craniofacial injuries seen with the introduction of bicycle-share electric scooters in an urban setting," *Journal of Oral and Maxillofacial Surgery,* vol. 77, no. 11, pp. 2292-2297, 2019.

[36] C. D. Schlaff, K. D. Sack, R.-J. Elliott and M. K. Rosner, "Early Experience with Electric Scooter Injuries Requiring Neurosurgical Evaluation in District of Columbia: A Case Series," *World Neurosurgery,* vol. 132, pp. 202-207, 2019.

[37] H. Yang, Q. Ma, Z. Wang, Q. Cai, K. Xie and D. Yang, "Safety of micro-mobility: analysis of E-Scooter crashes by mining news reports," *Accident Analysis & Prevention,* vol. 143, pp. 105608, 2020, doi: https://doi.org/10.1016/j.aap.2020.105608.

[38] N. Sikka, C. Vila, M. Stratton, M. Ghassemi and A. Pourmand, "Sharing the sidewalk: A case of E-scooter related pedestrian injury," *The American Journal of Emergency Medicine,* vol. 37, no. 9, pp. 1807.e5-1807.e7, 2019, doi: https://doi.org/10.1016/j.ajem.2019.06.017.

[39] S. Beck, L. Barker, A. Chan and S. Stanbridge, "Emergency department impact following the introduction of an electric scooter sharing service," *Emergency Medicine Australasia,* vol. 32, no. 3, pp. 409-415, 2020.

[40] L. J. Mayhew and C. Bergin, "Impact of e-scooter injuries on Emergency Department imaging," *Journal of Medical Imaging and Radiation Oncology,* vol. 63, no. 4, pp. 461-466, 2019.

[41] H. Stigson, I. Malakuti and M. Klingegård, "Electric scooters accidents: Analyses of two Swedish accident data sets," *Accident Analysis & Prevention,* vol. 163, pp. 106466, 2021, doi: https://doi.org/10.1016/j.aap.2021.106466.

[42] A. Azimian and J. Jiao, "Modeling factors contributing to dockless e-scooter injury accidents in Austin, Texas," *Traffic Injury Prevention,* vol. 23, no. 2, pp. 107-111, 2022, doi: https://doi.org/10.1080/15389588.2022.2030057.

[43] E. Karpinski, E. Bayles, L. Daigle and D. Mantine, "Characteristics of early shared E-Scooter fatalities in the United States 2018–2020," *Safety Science,* vol. 153, pp. 105811, 2022, doi: https://doi.org/10.1016/j.ssci.2022.105811.

[44] M. D. Traynor *et al.*, "Association of scooter-related injury and hospitalization with electronic scooter sharing systems in the United States," *The American Journal of Surgery,* vol. 223, no. 4, pp. 780-786, 2022, doi: https://doi.org/10.1016/j.amjsurg.2021.06.006.

[45] K. Anderson-Hall, B. Bordenkircher, R. O'Neil and S. C. Scott, "Governing micro-mobility: A nationwide assessment of electric scooter regulations," presented at the Transportation Research Board 98th Annual Meeting, Washington DC, 2019. Available: https://trid.trb.org/view/1572811, doi: https://trid.trb.org/view/1572811.

[46] R. William and K. Matt, "Exploring Best Practice for Municipal E-Scooter Policy in the United States.," presented at the 99th Annual Meeting of the Transportation Research Board, 2020. Available: https://ssrn.com/abstract=3512725, doi: https://doi.org/10.1016/j.tra.2021.06.025.

[47] R. William, K. Matt and B. David, "Exploring Best Practice for Municipal E-Scooter Policy in the United States," *Transportation Research Part A: Policy and Practice,* vol. 151, pp. 18-27, 2021, doi: https://doi.org/10.1016/j.tra.2021.06.025.

[48] O. James, J. I. Swiderski, J. Hicks, D. Teoman and R. Buehler, "Pedestrians and E-Scooters: An Initial Look at E-Scooter Parking and Perceptions by Riders and Non-Riders," *Sustainability,* vol. 11, no. 20, pp. 5591, 2019.

[49] A. Brown, N. J. Klein, C. Thigpen and N. Williams, "Impeding access: The frequency and characteristics of improper scooter, bike, and car parking," *Transportation Research Interdisciplinary Perspectives,* vol. 4, pp. 100099, 2020, doi: https://doi.org/10.1016/j.trip.2020.100099.

[50] S. Gössling, "Integrating e-scooters in urban transportation: Problems, policies, and the prospect of system change," *Transportation Research Part D: Transport and Environment,* vol. 79, pp. 102230, 2020, doi: https://doi.org/10.1016/j.trd.2020.102230.

[51] M. E. Moran, B. Laa and G. Emberger, "Six scooter operators, six maps: Spatial coverage and regulation of micromobility in Vienna, Austria," *Case Studies on Transport Policy,* 2020, doi: https://doi.org/10.1016/j.cstp.2020.03.001.

[52] S. Tuncer, E. Laurier, B. Brown and C. Licoppe, "Notes on the practices and appearances of e-scooter users in public space," *Journal of Transport Geography,* vol. 85, pp. 102702, 2020, doi: https://doi.org/10.1016/j.jtrangeo.2020.102702.

[53] K. Button, H. Frye and D. Reaves, "Economic regulation and E-scooter networks in the USA," *Research in Transportation Economics,* vol. 84, pp. 100973, 2020, doi: https://doi.org/10.1016/j.retrec.2020.100973.

[54] A. Brown, "Micromobility, Macro Goals: Aligning scooter parking policy with broader city objectives," *Transportation Research Interdisciplinary Perspectives,* vol. 12, pp. 100508, 2021/12/01/ 2021, doi: https://doi.org/10.1016/j.trip.2021.100508.

[55] Q. Ma, H. Yang, Y. Ma, D. Yang, X. Hu and K. Xie, "Examining municipal guidelines for users of shared E-Scooters in the United States," *Transportation Research Part D: Transport and Environment,* vol. 92, pp. 102710, 2021, doi: https://doi.org/10.1016/j.trd.2021.102710.

[56] S. Sareen, D. Remme and H. Haarstad, "E-scooter regulation: The micro-politics of market-making for micro-mobility in Bergen," *Environmental Innovation and Societal Transitions,* vol. 40, pp. 461-473, 2021, doi: https://doi.org/10.1016/j.eist.2021.10.009.

[57] A. Pashkevich, T. E. Burghardt, S. Puławska-Obiedowska and M. Šucha, "Visual attention and speeds of pedestrians, cyclists, and electric scooter riders when using shared road – a field eye tracker experiment," *Case Studies on Transport Policy,* vol. 10, no. 1, pp. 549-558, 2022, doi: https://doi.org/10.1016/j.cstp.2022.01.015.

[58] M. Chester, "The Electric Scooter Fallacy: Just Because They're Electric Doesn't Mean They're Green," ed, 2018.

[59] H. Moreau, L. de Jamblinne de Meux, V. Zeller, P. D'Ans, C. Ruwet and W. M. Achten, "Dockless E-Scooter: A Green Solution for Mobility? Comparative Case Study between Dockless E-Scooters, Displaced Transport, and Personal E-Scooters," *Sustainability,* vol. 12, no. 5, pp. 1803, 2020.

[60] S. Severengiz, S. Finke, N. Schelte and N. Wendt, "Life Cycle Assessment on the Mobility Service E-Scooter Sharing," in *2020 IEEE European Technology and Engineering Management Summit (E-TEMS)*, 2020, pp. 1-6: IEEE.

[61] S. Severengiz, S. Finke, N. Schelte and H. Forrister, "Assessing the Environmental Impact of Novel Mobility Services using Shared Electric Scooters as an Example," *Procedia Manufacturing,* vol. 43, pp. 80-87, 2020, doi: https://doi.org/10.1016/j.promfg.2020.02.114.

[62] A. de Bortoli and Z. Christoforou, "Consequential LCA for territorial and multimodal transportation policies: method and application to the free-floating e-scooter disruption in Paris," *Journal of Cleaner Production,* vol. 273, pp. 122898, 2020, doi: https://doi.org/10.1016/j.jclepro.2020.122898.

[63] A. de Bortoli, "Environmental performance of shared micromobility and personal alternatives using integrated modal LCA," *Transportation Research Part D: Transport and Environment,* vol. 93, pp. 102743, 2021, doi: https://doi.org/10.1016/j.trd.2021.102743.

[64] H. Peng, Y. Nishiyama and K. Sezaki, "Assessing environmental benefits from shared micromobility systems using machine learning algorithms and Monte Carlo simulation," *Sustainable Cities and Society,* vol. 87, pp. 104207, 2022, doi: https://doi.org/10.1016/j.scs.2022.104207.

[65] S. He and K. G. Shin, "Dynamic Flow Distribution Prediction for Urban Dockless E-Scooter Sharing Reconfiguration," in *Proceedings of The Web Conference 2020*, 2020, pp. 133-143.

[66] S. He and K. G. Shin, "Distribution Prediction for Reconfiguring Urban Dockless E-Scooter Sharing Systems," *IEEE Transactions on Knowledge and Data Engineering,* vol. 34, no. 12, pp. 5722-5740, 2022, doi: https://doi.org/10.1109/TKDE.2021.3062074.

[67] S. W. Ham, J.-H. Cho, S. Park and D.-K. Kim, "Spatiotemporal Demand Prediction Model for E-Scooter Sharing Services with Latent Feature and Deep Learning," *Transportation Research Record,* vol. 2675, no. 11, pp. 34-43, 2021, doi: https://doi.org/10.1177/03611981211003896.

[68] S. Phithakkitnukooon, K. Patanukhom and M. G. Demissie, "Predicting Spatiotemporal Demand of Dockless E-Scooter Sharing Services with a Masked Fully Convolutional Network," *ISPRS International Journal of Geo-Information,* vol. 10, no. 11, pp. 773, 2021.

[69] Y. Xu, X. Zhao, X. Zhang and M. Paliwal, "Real-Time Forecasting of Dockless Scooter-Sharing Demand: A Spatio-Temporal Multi-Graph Transformer Approach," 2021.

[70] S. Kim, S. Choo, G. Lee and S. Kim, "Predicting Demand for Shared E-Scooter Using Community Structure and Deep Learning Method," *Sustainability,* vol. 14, no. 5, pp. 2564, 2022.

[71] P. W. Khan, S.-J. Park, S.-J. Lee and Y.-C. Byun, "Electric Kickboard Demand Prediction in Spatiotemporal Dimension Using Clustering-Aided Bagging Regressor," *Journal of Advanced Transportation,* vol. 2022, pp. 8062932, 2022, doi: https://doi.org/10.1155/2022/8062932.

[72] M. Masoud, M. Elhenawy, M. H. Almannaa, S. Q. Liu, S. Glaser and A. Rakotonirainy, "Heuristic approaches to solve e-scooter assignment problem," *IEEE Access,* vol. 7, pp. 175093-175105, 2019.

[73] A. Ciociola, M. Cocca, D. Giordano, L. Vassio and M. Mellia, "E-Scooter Sharing: Leveraging Open Data for System Design," in *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2020, pp. 1-8, doi: 10.1109/DS-RT50469.2020.9213514.

[74] L. Tolomei, S. Fiorini, A. Ciociola, L. Vassio, D. Giordano and M. Mellia, "Benefits of Relocation on E-scooter Sharing - a Data-Informed Approach," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3170-3175, doi: 10.1109/ITSC48978.2021.9564809.

[75] J. Osorio, C. Lei and Y. Ouyang, "Optimal rebalancing and on-board charging of shared electric scooters," *Transportation Research Part B: Methodological,* vol. 147, pp. 197-219, 2021/05/01/ 2021, doi: https://doi.org/10.1016/j.trb.2021.03.009.

[76] A. M. Fathabad, X. Li, J. Cheng and Y.-J. Wu, "Data-Driven Optimization for E-Scooter System Design," (in English), Tech Report 2022.

[77] G. Losapio, F. Minutoli, V. Mascardi and A. Ferrando, "Smart balancing of E-scooter sharing systems via deep reinforcement learning," in *22nd Workshop "From Objects to Agents"*, Bologna, Italy, 2022, vol. 2963, pp. 83–97: CEUR-WS.org.

[78] O. Altintasi and S. Yalcinkaya, "Siting charging stations and identifying safe and convenient routes for environmentally sustainable e-scooter systems," *Sustainable Cities and Society,* vol. 84, pp. 104020, 2022, doi: https://doi.org/10.1016/j.scs.2022.104020.

CHAPTER 3
3. SHORT-TERM SUPPLY LEVEL PLANNING FOR SHARED E-SCOOTERS

3.1 Introduction

Supply level planning plays a crucial role in operational management, as it is influenced by various factors such as operating costs, resource constraints, customer satisfaction, and environmental emissions. As discussed in the previous chapters, dockless shared e-scooters encounter several operational challenges, including volatile demand, high operating costs, unforeseen trip purposes, emissions, and regulations. As a dockless mode, operators must strategically allocate their limited e-scooters to the right place and time, utilizing several strategies such as overnight distribution, regular or real-time rebalancing or relocation, and customer incentives. Addressing these issues requires effective estimation of the supply level for this emerging shared micromobility. In this study, the term "short-term supply level planning" refers to the estimation of the total supply of shared e-scooters within a planning horizon of one day (e.g., hourly intervals). Various prediction models have been employed and proposed to forecast spatial and temporal demand, including statistical models, machine learning, and deep learning, to improve the effectiveness of supply level planning of shared transportation modes.

In time-series analysis, the Autoregressive Integrated Moving Average (ARIMA) model, along with its seasonal variant known as Seasonal-ARIMA (SARIMA), holds prominence. Moreover, reference [1] provides insights into additional advancements in this domain. Implementing the ARIMA model mandates the assurance of both normality and stationarity. Notably, the challenge of normality can be effectively addressed using the Box-Cox transformation method, as expounded in [2]. Among machine learning models, Random Forest regression (RF) stands out for its remarkable predictive accuracy. Remarkably, this model's performance could be on par with certain deep learning architectures [3]. Following a similar conceptual framework as RF, XGBoost undertakes data fitting via gradient-boosted decision trees, thereby effectively reducing the computational time [4].

Machine learning models show their effective performance compared to conventional statistical models and even have comparable results with deep learning models [3, 5]. Another advantage of machine learning is interpretability which is the main limitation of deep learning models. Decision Tree (DT) algorithm is one of the most powerful models and has several extensions to improve the prediction performance and generalization, such as random forest or RF, extra tree or ET, gradient boosting or GB [6], extreme gradient boosting or XGBoost [7], etc. For instance, RF showed huge prediction improvement compared to DT and Naïve Bayes in predicting the usage frequency of shared e-scooters at the University of Malaya [8]. Support Vector Machine (SVM) was employed to predict the daily and hourly bike-sharing demand [9]. In predicting the daily demand of station-based bike sharing in Seoul, k-nearest neighbors (KNN) had better prediction performance than linear regression, but it is still lots worse than RF and SVM [10].

Over the recent decades, deep learning has garnered significant attention due to its auspicious performance surpassing traditional methodologies. In managing sequential data, Recurrent Neural Networks (RNNs) exhibit superior efficacy compared to conventional neural

network architectures. This advantage stems from their incorporation of a recurrent cell (tanh or sigma cell), enabling the retention of prior memory. The evolution of RNNs led to the conception of Long-Short Term Memory Neural Networks (LSTM NNs), a progression achieved by introducing specialized gates (forget gate, input gate, and output gate) into the recurrent cell. These gates work synergistically to adeptly manage long-term dependencies [11]. Moreover, LSTM NNs effectively circumvent the shortcomings of RNNs, such as the predicaments of vanishing and exploding gradients [12, 13]. Subsequently, to streamline the architecture and reduce trainable parameters, the amalgamation of the forget gate and input gate into a singular update gate resulted in the inception of the Gated Recurrent Unit (GRU) [14]. Unlike LSTM, the GRU comprises only two gates: the update gate and the reset gate. Additionally, a variety of other extensions of LSTM NNs are explored in comprehensive surveys of RNNs [12].

Even though the family of RNNs successfully improves the performance of sequential or temporal prediction, these architectures were considered to have a limited performance for spatiotemporal datasets. This is because RNNs are usually trained spatially independently or based on only local information. To account for the spatial pattern for traffic flow, Convolutional Neural Networks (CNNs) and RNNs were combined using CNNs to extract the spatial pattern while RNNs to learn the temporal patterns [15]. Similarly, 3-Dimensional CNNs with LSTM NNs were employed to forecast PM2.5 concentration in China [16]. To deal with non-Euclidean spatial pattern extraction of CNNs, Graph Neural Networks (GNNs) are recently developed and extended, as reviewed by [17]. For GNNs, the non-Euclidean features are used as information filters to control the parameter-sharing between nodes. For instance, the link connection is used as the adjacency matrix for Hybrid GNNs for road traffic prediction [18]. Furthermore, three non-Euclidean features (neighborhood, functional similarity, and transportation connectivity) were used to construct the spatial dependency for GNNs to predict ride-hailing demand [19].

Conversely, the exploration of volatility or variance analysis has been predominantly concentrated within the field of econometrics, wherein it holds the potential to furnish invaluable insights to bolster decision-making endeavors. The Autoregressive Conditional Heteroscedasticity (ARCH) model emerges as a statistical regression framework tailored for prognosticating forthcoming variance or volatility [1]. This paradigm encompasses two distinct formulations: ARCH in variance and ARCH in mean (ARCH-M). ARCH solely incorporates the squared residuals from preceding lags as independent variables, while the Generalized ARCH (GARCH) model encompasses historically predicted variances as well. The GARCH model has spurred numerous extensions, including but not limited to Power ARCH, Threshold ARCH, and Exponential ARCH. Notably, several ARCH models were employed to predict the return rate of daily closing prices for the Shanghai and Shenzhen 300 Index [20]. Likewise, in a similar vein, ARMA-GARCH and ARMA-TARCH were utilized to anticipate volatility in both traditional and sustainable stock indices within the FTSE4Good index series family by Ti et al. [21]. In addition, diverse adaptations, including ARMA-GARCH, SARIMA-GARCH, and SARIMA-SGARCH, were trained to forecast the precipitation index [22], daily peak electricity demand [23], and internet traffic [24], respectively. While ARCH-M exhibits incremental enhancements in predictive accuracy over ARIMA, it may grapple with convergence criteria and protracted training durations. Recently, there has been a confluence of GARCH and deep

learning models to foresee price volatility across pivotal metals such as Gold, Silver, and Copper [25, 26].

In this case, several prediction models have been employed and adapted to predict the spatiotemporal demand of dockless shared e-scooters such as spatiotemporal graph capsule neural network (GCScoot [27] and GCScoot2 [28]), encoder-recurrent neural network–decoder (ERD) [29], Masked Fully Convolutional Network (MFCN) [30], Spatio-Temporal Multi-Graph Transformer (STMGT) [31], LSTM NNs [32], and bagging ensemble approach of XGBoost, RF, and Extra Tree (ET) [33]. To encapsulate, many formidable prediction models have been advanced for anticipating transportation demand, specifically for shared bicycles and e-scooters. However, the preponderance of these models primarily centers on enhancing accuracy metrics. Consequently, the inherent heteroscedasticity observed in transportation demand is often overlooked, resulting in an underutilization of historical data insights. Moreover, the variance associated with heteroscedastic datasets is not constant, necessitating its incorporation in supply strategizing. Articulated differently, the determination of inventory or supply levels is intrinsically linked to both the residuals emanating from the demand forecasting model and the dataset's heteroscedastic nature. Hence, a comprehensive variance analysis emerges as imperative for crafting a refined supply level estimation model. Achieving this nuanced approach can be facilitated through the formulation of conditional variance models, exemplified by constructs like SGARCH, or by adopting data transformation methodologies such as the Box Cox transformation.

This study furnished prospective contributions to the domain of shared e-scooters and supply level strategizing. Initially, the study unveiled spatiotemporal trends in shared e-scooter demand through the analysis of three distinct datasets from Thammasat University (Thailand), Minneapolis (Minnesota), and Austin (Texas). Secondly, the study addressed the heteroscedastic nature inherent in shared e-scooter demand when shaping supply level strategies. This was manifested by the introduction of the Mean Oversupply (MO) metric, designed to facilitate the assessment of efficiency at specific proportions of served demand. Lastly, the inquiry illuminated the merits and demerits associated with the application of the Box Cox transformation, encompassing its repercussions on demand prediction precision and supply-level strategizing.

3.2 Methodology

3.2.1 Research framework

Based on the literature review in the previous section, we could see that many methods were proposed to deal with spatiotemporal demand prediction, including statistical regression models, machine learning algorithms, and shallow or deep learning approaches. Even though these regression models could achieve state-of-art performance, there are still prediction errors or residuals commonly presenting in the form of mean squared error (MSE), Root MSE (RMSE), mean absolute error (MAE), etc. Reducing these metrics leads to lower demand uncertainty resulting in higher planning efficiency. Since the inventory or supply level planning party depends on the variation of demand prediction models, variance analysis is necessary to further reduce the uncertainty, particularly for a heteroscedastic dataset. Therefore, this section

aims to answer the second objective by combining the demand and variance predictions to design an efficient supply planning model.

To realize this aim, the methodology (see **Figure 3.1**) employed in this section was delineated into five core segments: data preprocessing, data manipulation, demand projection, variance projection, and supply level estimation. The initial phase encompassed the collection, encoding, and incorporation of diverse attributes, encompassing shared e-scooter data, meteorological variables, annual events, public holidays, days of the week, and temporal intervals. Drawing upon insights from the literature review, a prevalent practice was the normalization of data within the range of 0 to 1, aligning them for compatibility with specific activation functions; however, instances of training on the original scale were also identified. Recognizing the potential enhancements in prediction accuracy and mitigation of heteroscedastic effects attributed to the Box Cox transformation [34], the subsequent stage incorporated this transformation as an additional option for data manipulation.

In the subsequent phase, an array of machine learning and deep learning models were conceived to undertake the prediction of shared e-scooter demand on an hourly basis. Concurrently, Grid Search (GS) and Bayesian Optimization (BO) were utilized to tune the hyperparameters of the prediction models, including Seasonal Autoregressive Integrated Moving Average with exogenous variables (SARIMAX), RF, XGBoost, Fully Connected Neural Networks (FCNNs), RNNs, and GRUs. The primary objective underpinning the comparative performance assessment of Box Cox-transformed data against original/normalized data aimed to highlight a distinct contrast: while the residuals stemming from prediction models employing Box Cox-transformed data exhibited an absence of heteroscedasticity, such effects were discernible in the case of models using original or normalized data. In this stage, several variation models (constant, daily, and SGARCH variances) were employed to forecast the residuals of these demand prediction models. Under its capacity in mitigating heteroscedastic tendencies [2, 34], the Box Cox transformation ensured a constant variance in the transformed data.

Then again, within the scope of accuracy evaluation, the optimal models among original and normalized data were selected. The residuals of these models were subsequently subjected to variance analysis across three distinct scenarios: constant variance, daily variance, and variance prediction through the employment of SGARCH. Consequently, the transformed Box Cox data exhibited singular variance modeling (Constant Variance), while the original or normalized data were subjected to three variance models (constant, daily, and SGARCH variances). In the pursuit of variance analysis and the formulation of supply level estimation, exclusive scrutiny was devoted to three models—SARIMAX, XGBoost, and GRUs. The exclusion of XGBoost and RF stemmed from their analogous predictive performance, while GRUs were chosen based on their parity with the performance levels observed in FCNNs and RNNs. Subsequently, the anticipated demand (step 3) and the projected variance (step 4) were synergistically leveraged in devising the Supply Level framework of step 5. In this phase, a novel metric, the Mean Oversupply (MO), was introduced to facilitate a comparative assessment of the efficacy of the four supply-level models across a specified spectrum of served demand, ranging from 70% to 98%.

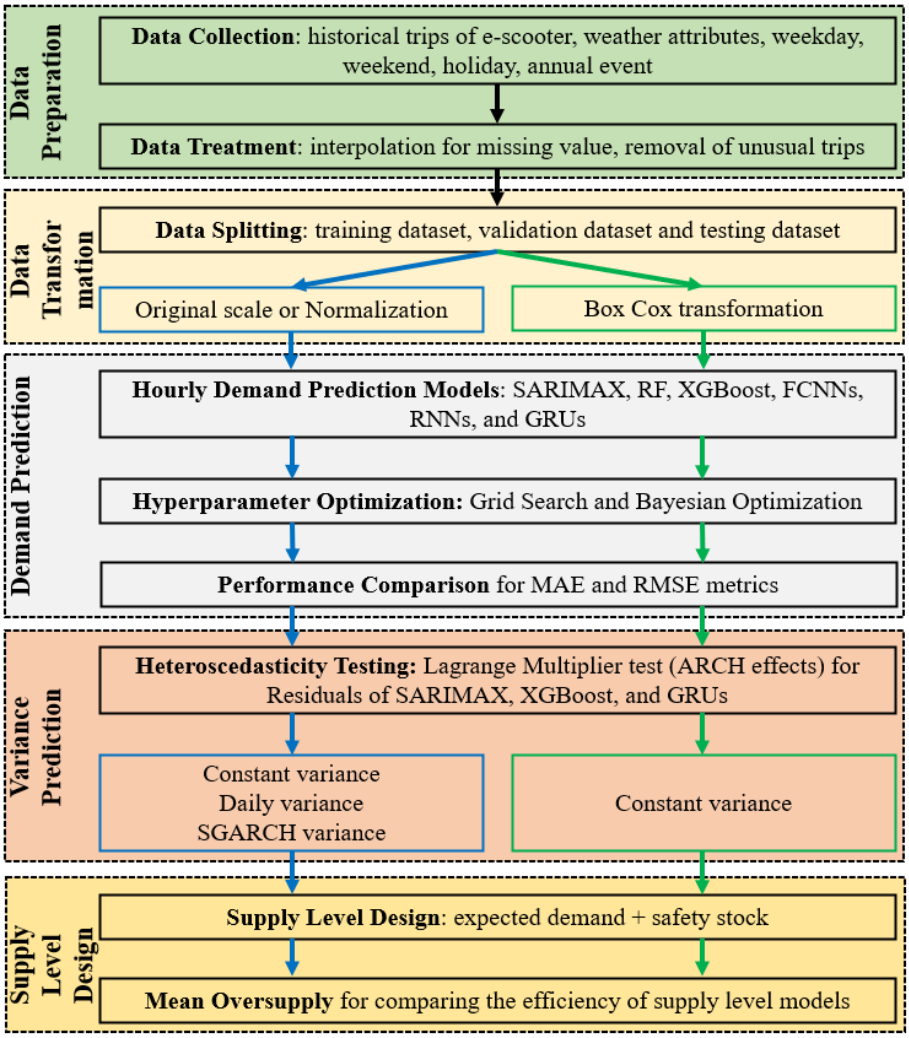

Figure 3.1 Framework for supply level planning

3.2.2 Data transformation

Data transformation could impact prediction performance and conform the data to assumptions specifically for statistical models. From previous studies, the demand prediction models, especially machine learning and deep learning models commonly trained with original or normalized scale. Two popular normalization techniques aim to convert the data to have the same distribution (mean and z-score normalization) or the same scale (min-max normalization or 0-1 scale). Min-max normalization in **Eq. 3.1** was considered in this study as it is suitable for some activations of deep learning models (ex., ReLU). Alternatively, the Box Cox transformation stands as a widely adopted power transformation methodology grounded in a likelihood maximization estimator. Its principal objectives encompass the stabilization of variance, the reduction of skewness, and the alignment of data with a normal distribution. Box Cox transformation supports only the positive value, so Yeo and Jonhson [35] extended the formulation in **Eq. 3.2** to support both positive and negative values while improving the normality and symmetry. The formulation of these two transformations is as follows:

$$x_t^{norm} = x_t - \min(x_t)/(\max(x_t) - \min(x_t)) \quad (3.1)$$

$$x_{t,r}^{BC} = \begin{cases} \lambda_r^{-1}\left[\left(x_{t,r}+1\right)^{\lambda_r}-1\right] & \text{if } \lambda_r \neq 0, x_{t,r} \geq 0 \\ \ln\left(x_{t,r}+1\right) & \text{if } \lambda_r = 0, x_{t,r} \geq 0 \\ -\left[\left(-x_{t,r}+1\right)^{2-\lambda_r}-1\right]/(2-\lambda_r) & \text{if } \lambda_r \neq 2, x_{t,r} < 0 \\ -\ln\left(-x_{t,r}+1\right) & \text{if } \lambda_r = 2, x_{t,r} < 0 \end{cases} \quad (3.2)$$

Where x_t^{norm} is the normalized scale of the variable x_t, while $x_{t,r}^{BC}$ is the Box Cox scale of e-scooter demand $x_{t,r}$ at time t and region r. λ_r is Box Cox transformation's parameter for region r. Since e-scooter demand is a nonnegative variable, the change was the first case of **Eq. 3.2**, but it has the maximum requirement. In other words, the predicted transformed demand $\hat{x}_{t,r}^{BC}$, including the supply level, must be less than $-1/\lambda$, specifically when $\lambda < 0$.

3.2.3 Demand prediction

3.2.3.1 Autoregressive integrated moving average (ARIMA)

ARIMA or Box-Jenkins model is a popular time series model applying differencing to make data stationary (Integrated) while the disturbances follow a linear autoregressive moving average (ARMA) specification. To remove the seasonal patterns, ARIMA was extended by deseasonalizing and including the seasonal ARMA, called SARIMA. Occasionally, the independent variables of these two models also include the exogenous variables, called ARIMAX or SARIMAX. The general formulation of SARIMA(p,d,q)(P,D,Q,S) [1] are as follows:

$$\rho(L^p)\rho_S(L^P)\Delta^d\Delta_S^D \boldsymbol{y_t} = \alpha + \theta(L^p)\theta_S(L^P)\epsilon_t \quad (3.3)$$

Where:

$\rho_S(L^P) = 1 - \rho_{S,1}(L^S) - \rho_{S,2}(L^{2S}) - \cdots - \rho_{S,P}(L^{PS})$
$\theta_S(L^P) = 1 + \theta_{S,1}(L^S) + \theta_{S,2}(L^{2S}) + \cdots + \theta_{S,P}(L^{PS})$
L is lag operator ($L^j y_t = y_{t-j}$)
α is constant term
Δ^d is differencing by Δ operator d times (0 – 2)
Δ_S^D is deseasonalizing by Δ operator between seasonal lag S
$\epsilon \sim i.i.d.N(0,\sigma^2)$ is white noise disturbance

3.2.3.2 Random forest (RF)

Random Forest (RF) is a powerful machine learning algorithm dealing with high dimensional data while requiring just a small amount of data and training time, introduced by Breiman [36]. RF leverages the results from many random decision trees' predictions (see **Section 4.2.1**), while large numbers of trees are built from randomly selected inputs or combinations of inputs (bootstrapping or bootstrap sampling), see **Figure 3.2**. In this case, hundreds of bootstrapping samples (i.e., randomly resampling with replacement) were drawn, while decision tree regression was built for each sample (called week learners). For regression

problems, RF is simply the average prediction results from each regression tree, while majority-voting was employed for classification problems. This bootstrap aggregating (or bagging) technique could improve the prediction performance of DT as it has high-variance and low-bias procedures. This study trained the RF model using a Python module, RandomForestRegressor of scikit-learn. Several parameters of RF were examined, such as number of trees in the forest, criterion, and depth of regression tree.

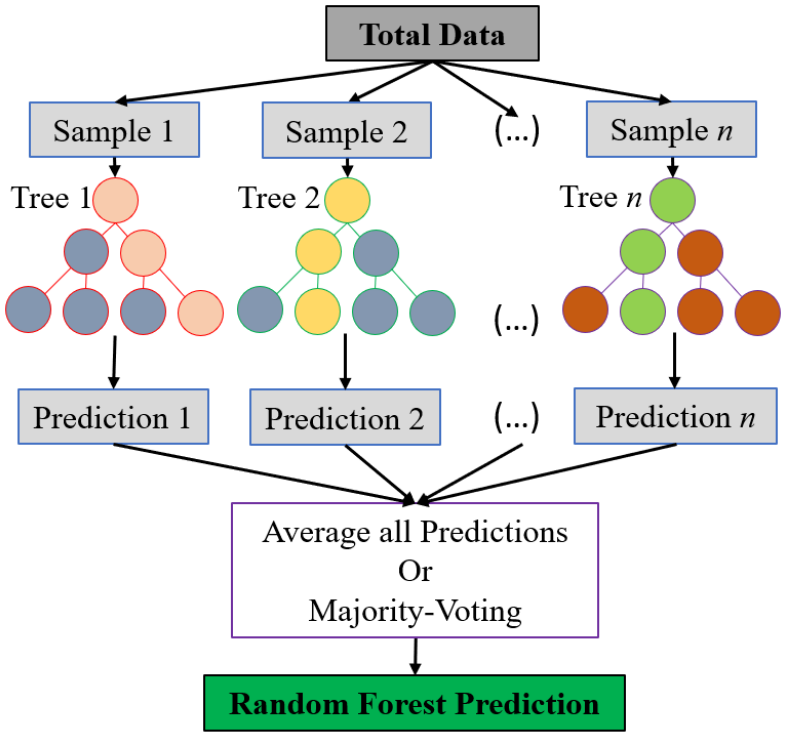

Figure 3.2 Flowchart of Random Forest (RF): average all predictions for regression problem and majority-voting for classification problem

3.2.3.3 Extreme gradient boosting (XGBoost)

Extreme Gradient Booting (XGBoost) is an efficient, scalable, and distributed gradient booting regressor, started as a research project in the Distributed (Deep) Machine Learning Community (DMLC) group [4]. Unlike other gradient boosting algorithms, XGBoost has clever penalization of base estimators, a proportional shrinking of terminal nodes, Newton Boosting, extra randomization parameters, automatic feature selection, and parallel computing. Moreover, it also accepts sparse input and the input types as a dense matrix, sparse matrix, data file, or their own class xgb.DMatrix. XGBoost can quickly optimize the loss function as it considers both the first-order gradient $\hat{g}_m(x) = \left[\frac{\partial L(y,F(x))}{\partial F(x)}\right]_{F(x)-F_{n-1}(x)}$ and the second order gradient $\hat{h}_m(x) = -\left[\frac{\partial L^2(y,F(x))}{\partial F(x)^2}\right]_{F(x)-F_{n-1}(x)}$. Instead of residuals, the base learner of XGBoost was trained on the negative ratio of these two gradient functions, $\left(-\hat{g}_m(x)/\hat{h}_m(x)\right)$. In this case, the python package of XGBoost was used, while two parameters were tuned including depth of the trees and the number of gradient-boosted trees.

3.2.3.4 Recurrent neural networks (RNN, LSTM, GRU)

Figure 3.3 shows the three popular cells of RNNs used in transportation demand prediction, i.e., simple RNNs cell (*a*), LSTM cell (*b*), and GRU cell (*c*). There are four common architectures of RNNs based on the number of inputs and outputs: one-to-one, one-to-many, many-to-one, and many-to-many. In the previous studies, most of the RNNs were employed as many-to-one architecture based on the local information, while several spatial features were included, such as POI, employment density, population density, etc. Therefore, the architecture could be trained spatially combined or spatially independent. Since the spatial features are static, separate models could provide better prediction performance, but this technique is time-consuming for many spatial data. On the other hand, many-to-many architecture RNNs could be another option as historical data of all spatial demands and external features were combined as the input to predict all the future spatial demands. Since this study did not process the exterior spatial features, the demand prediction was examined for both spatially independent and spatially combined cases, while the best prediction models were selected.

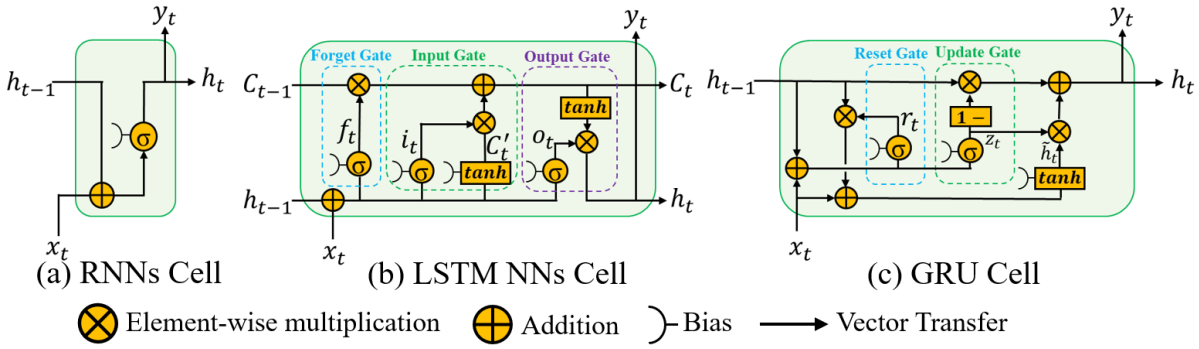

Figure 3.3 Schematic illustrations of (a) recurrent neural networks, (b) long short-term memory neural networks, and (c) gated recurrent unit

The simple RNNs have sigma (sigmoid function) or tanh cells working as memory cells which fuse the current input with previous states. These cells enable RNNs to perform better than conventional neural networks. The formula of RNNs cell could be written as:

$$y_t = h_t = sigmoid(W_h h_{t-1} + W_x x_t + b) \quad (3.4)$$

$$sigmoid(x) = 1/(1 + e^x) \quad (3.5)$$

Where x_t , h_t and y_t denote the cell's inputs, recurrent information, and output at the time *t*. And W_h , W_x, and *b* are training weights and biases.

The standard LSTM NNs have three gates: Forget Gate, Input Gate, and Output Gate. Forget Gate could be removed from the LSTM cell, but its performance was poor [12]. Therefore, the mathematical expressions of LSTM NNs are:

$$f_t = sigmoid\left(W_{fh} h_{t-1} + W_{fx} x_t + b_f\right) \quad (3.6)$$

$$i_t = sigmoid(W_{ih} h_{t-1} + W_{ix} x_t + b_i) \quad (3.7)$$

$$C'_t = tanh(W_{Ch} h_{t-1} + W_{Cx} x_t + b_C) \quad (3.8)$$

$$tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \quad (3.9)$$
$$C_t = f_t \circ C_{t-1} + i_t \circ C'_t \quad (3.10)$$
$$O_t = sigmoid(W_{Oh}h_{t-1} + W_{Ox}x_t + b_O) \quad (3.11)$$
$$h_t = O_t \circ tanh(C_t) \quad (3.12)$$

Where ∘ denotes the pointwise multiplication of two matrices called Hadamard product. W and b are training weight matrices and bias vectors, respectively. f_t is forget gate, i_t is the input gate and C'_t is current memory. C_t is the combination of current memory C'_t and long-term memory C_{t-1}. Finally, the output gate O_t controls the temporal information for the output h_t.

GRU has only two gates, reset gate and update gate, so it requires shorter training time than LSTM NNs [3] with comparable performance [37]. For this reason, GRU is more suitable for hyperparameter tuning than LSTM NNs. The training process of GRU could be expressed as follows:

$$r_t = sigmoid(W_{rh}h_{t-1} + W_{rx}x_t + b_r) \quad (3.13)$$
$$z_t = sigmoid(W_{zh}h_{t-1} + W_{zx}x_t + b_z) \quad (3.14)$$
$$\widetilde{h_t} = tanh(W_{\tilde{h}h}(r_t \circ h_{t-1}) + W_{\tilde{h}x}x_t + b_{\tilde{h}}) \quad (3.15)$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \widetilde{h_t} \quad (3.16)$$

Likewise, *W* and *b* denote the matrices of training weights and vectors of bias, respectively. r_t denotes the reset gate, while z_t denotes the update gate. Within this context, the GRUs' output h_t at a given time instance *t* is realized as a linear combination of the previous output h_{t-1} and the projected output $\widetilde{h_t}$. The sequential layers of GRU in this study are the input layer with GRU cell, dropout layer, *n* number of hidden layers with GRU cell, and dense layer as the output layer.

3.2.4 Variance prediction

As acknowledged, it is understood that the true data cannot be precisely forecasted (i.e., $y = \hat{y} + \sigma(X)\varepsilon$), given the assumption of an absence of associated errors in the observed data. In this particular context, $\hat{y} = \mathrm{E}(y|X), \mathrm{E}(\varepsilon|X) = 0, \mathrm{Var}(\varepsilon|X) = \mathrm{E}(\varepsilon^2|X) - \mathrm{E}^2(\varepsilon|X) = 1$, $\mathrm{Var}(y|X) = \sigma^2(\mathrm{X}) > 0$, and X and ε are independent. Homoscedasticity pertains to the circumstance where variance remains consistent, whereas heteroscedasticity indicates variable variance. For models grounded in probabilistic principles, such as those predicated on assumptions of stationary data, data distribution, and homoscedasticity, diagnostic assessments hold significance. Regrettably, this facet has been frequently overlooked within the context of machine learning and deep learning. In situations featuring heteroscedastic data, the variance can be constructed as a function of the random variables X. In the domain of time-series analysis, the assessment of heteroscedasticity of residuals is typically conducted through the examination of autocorrelation in squared residuals and the employment of the Lagrange Multiplier (ARCH-LM) test. Typically, the formulation of variance involves the utilization of the previous variances and squared residuals. This conception of conditional variance is rooted in the notion that periods marked by high and low variance are clustered together [1]. At this juncture, two distinct alternatives emerge: the incorporation or exclusion of the conditional

variance in influencing the conditional mean. The simultaneous prediction approach, which entails integrating the conditional variance within the conditional mean, can yield a nonconvex objective function, thereby leading to heightened computational demands. This is primarily due to the augmented parameter set that necessitates estimation, particularly during hyperparameter tuning. Given these considerations, the present study has opted to pursue the avenue of disjointly predicting the anticipated mean and the conditional variance. This decision entails disregarding the influence of the conditional variance on the expected mean. This approach offers several merits, notably a streamlined model formulation through univariate variance modeling, alongside simplified hyperparameter tuning for both demand and variance prediction. However, a drawback lies in the potential foregone accuracy enhancement achievable by encompassing the conditional variance in the demand prediction model. For example, in a similar vein, ARIMA and GARCH were employed to predict the demand and variance, respectively, for safety stock estimation by Trapero et al. [38]. In this study, resulting from the demand prediction process detailed in **Section 3.2.3** were employed to train the variance models. This yielded the formulation of three distinct variance models: constant variance as described in **Eq. 3.17**, daily seasonal variance as articulated in **Eq. 3.18**, and forecasted variance achieved via SGARCH as delineated in **Eq. 3.19**, as presented below:

$$\sigma_{con}^2(r) = \frac{1}{T}\sum_{t=1}^{T} \varepsilon_{(t,r)}^2 \tag{3.17}$$

$$\sigma_{seas}^2(t,r) = \frac{1}{N}\left(\varepsilon_{(t-24,r)}^2 + \varepsilon_{(t-2*24,r)}^2 + \cdots + \varepsilon_{(t-N*24,r)}^2\right) \tag{3.18}$$

$$\sigma_{SGARCH}^2(t,r) = a_0 + a_1\varepsilon_{(t-1,r)}^2 + a_2\varepsilon_{(t-2,r)}^2 + a_3\varepsilon_{(t-24,r)}^2 + b_1\sigma_{(t-1,r)}^2 + b_2\sigma_{(t-2,r)}^2 + b_3\sigma_{(t-24,r)}^2 \tag{3.19}$$

In **Eq. 3.17**, the constant variance of the region r, denoted as $\sigma_{con}^2(r)$, manifests as the average of squared residuals stemming from the projected demand within that specific region. Correspondingly, **Eq. 3.18** presents the formulation of the seasonal daily variance, $\sigma_{seas}^2(t,r)$, which signifies the average of squared residuals pertaining to the anticipated demand during the identical hour of each day. Here, N denotes the total number of days. Although the theoretical equivalence of the seasonal variance on average to the constant variance is expected, the former usually exhibits a marginal diminution due to the propensity for evaluation residuals' mean to deviate from zero. Formally, $\sigma_{con}^2(r) = \frac{1}{24}\sum_{t=1}^{24}\sigma_{seas}^2(t,r) + \frac{1}{24}\sum_{t=1}^{24}\left(E[\varepsilon_{seas}(t,r)] - E\left[\varepsilon_{(r)}\right]\right)^2$. Notably, the computation of the constant and daily seasonal variances is grounded in the training dataset. Lastly, **Eq. 3.19** presents the predicted variance by SGARCH, $\sigma_{SGARCH}^2(t,r)$, which was trained separately for each region r. The SGARCH model's training leveraged maximum log-likelihood estimation [1]. In this context, a daily seasonal pattern (S = 24) was chosen, and insignificant parameters (at the 95% confidence level) in this equation would be discarded.

3.2.5 Supply level planning

As depicted in **Figure 3.1**, the conceptual framework introduced in this research endeavors to facilitate periodic rebalancing operations by adeptly harnessing insights from historical data. This strategic framework endeavors to forecast forthcoming demand and variance, thereby facilitating the design of well-informed supply levels or inventory levels. Within this

investigation, the term "supply level" is defined as the aggregate of supplies stemming from operator-initiated rebalancing, drop-offs, and available e-scooters within the vicinity. Notably, operators tasked with rebalancing e-scooters must meticulously factor in fluctuations in drop-off demand, inventory levels, and lead time, all of which can be approximated through the models outlined in this study. In accordance with our approach, the estimation of total supply is grounded in the demand side, precisely pick-up demand. The study's focal point revolves around the anticipation of comprehensive demand before the initiation of rebalancing actions. As a result, the aggregate supply level is inferred from the demand side, specifically employing solely the pick-up data. To elaborate further, the term "supply level" within this context—pertaining to inventory or order-up-to levels—adheres to the same formulation as the confidence interval, which represents the summation of the projected pick-up demand (as detailed in **Section 3.2.3**) and the safety stock (hinged on the predicted variance outlined in **Section 3.2.4**). The comparative examination of various supply level models was conducted to elucidate the efficacy of considering the heteroscedastic nature inherent in shared e-scooter demand, thereby informing operational planning with heightened precision.

Safety stock signifies the inventory allocation aimed at averting stockouts, a consequence of demand fluctuations, inaccuracies in forecasts, and supply lead time [39]. In the context of station-based shared bicycles, supply levels are devised in alignment with the target service level, denoting the likelihood of encountering a shortage event for both pick-up and drop-off demands [39, 40]. However, in the scenario of dockless shared e-scooters, users possess the flexibility to terminate their trips at any locations, thus permitting the disregard of service levels for drop-off trips. Given the variance across supply level models in terms of service levels and backorder levels, the typical approach involves juxtaposing the deviations from the target cycle service level and backorder level (scaled by safety stock), often illustrated through curves. This facilitates the comparative assessment of inventory or supply level models [38]. Notably, this study took a distinct approach by evaluating supply level models at equivalent percentages of served demand (as depicted in **Figure 3.4**). This methodology consequently permits a comparison through a singular metric, denoted as the Mean Oversupply. The definitions of key expressions—Supply Level ($S_{(t,r)}$), Served Demand ($SD_{t,r}$), Percentage of Served Demand (P), Oversupply ($O_{t,r}$) and Mean Oversupply (MO) —are delineated below:

$$S_{(t,r)} = \widehat{D}_{(t,r)} + d * \sigma_{(t,r)} \tag{3.20}$$

$$SD_{(t,r)} = \min\{ D_{(t,r)} ,\ S_{(t,r)}\} \tag{3.21}$$

$$P = \sum_{r=1}^{R}\sum_{t=1}^{T} SD_{(t,r)} / \sum_{r=1}^{R}\sum_{t=1}^{T} D_{(t,r)} \tag{3.22}$$

$$O_{t,r} = \max\{S_{(t,r)} - D_{(t,r)} ; 0\} \tag{3.23}$$

$$MO = \frac{1}{RT}\sum_{r=1}^{R}\sum_{t=1}^{T} O_{t,r} \tag{3.24}$$

Eq. 3.20 illustrates the Supply Level, denoted as $S_{(t,r)}$, as the sum of the forecasted hourly demand $\widehat{D}_{(t,r)}$ and the forecasted safety stock within the time interval t of region r. In this equation, safety stock is represented as the outcome of multiplying the predicted standard deviation $\sigma_{(t,r)}$ and safety stock parameter d, which is contingent upon the function involving the target service level Z_{score}, lead time, and time increment [38, 39, 41]. The forecasted

standard deviations $\sigma_{(t,r)}$, as detailed in **Section 3.2.4**, is the square root of the predicted variance—potentially emanating from the constant variance, daily seasonal variance, or conditional variance computed by SGARCH. Lead time and time increment are influenced by the rebalancing frequency, thereby remaining consistent across various supply level models. As a consequence, these parameters are held constant at a unit value. In this specific context, d and Z_{score} exhibit equivalence; nevertheless, the safety stock parameter(d) is deliberately adjusted to attain the targeted percentage of served demand (refer to **Figure 3.4**).

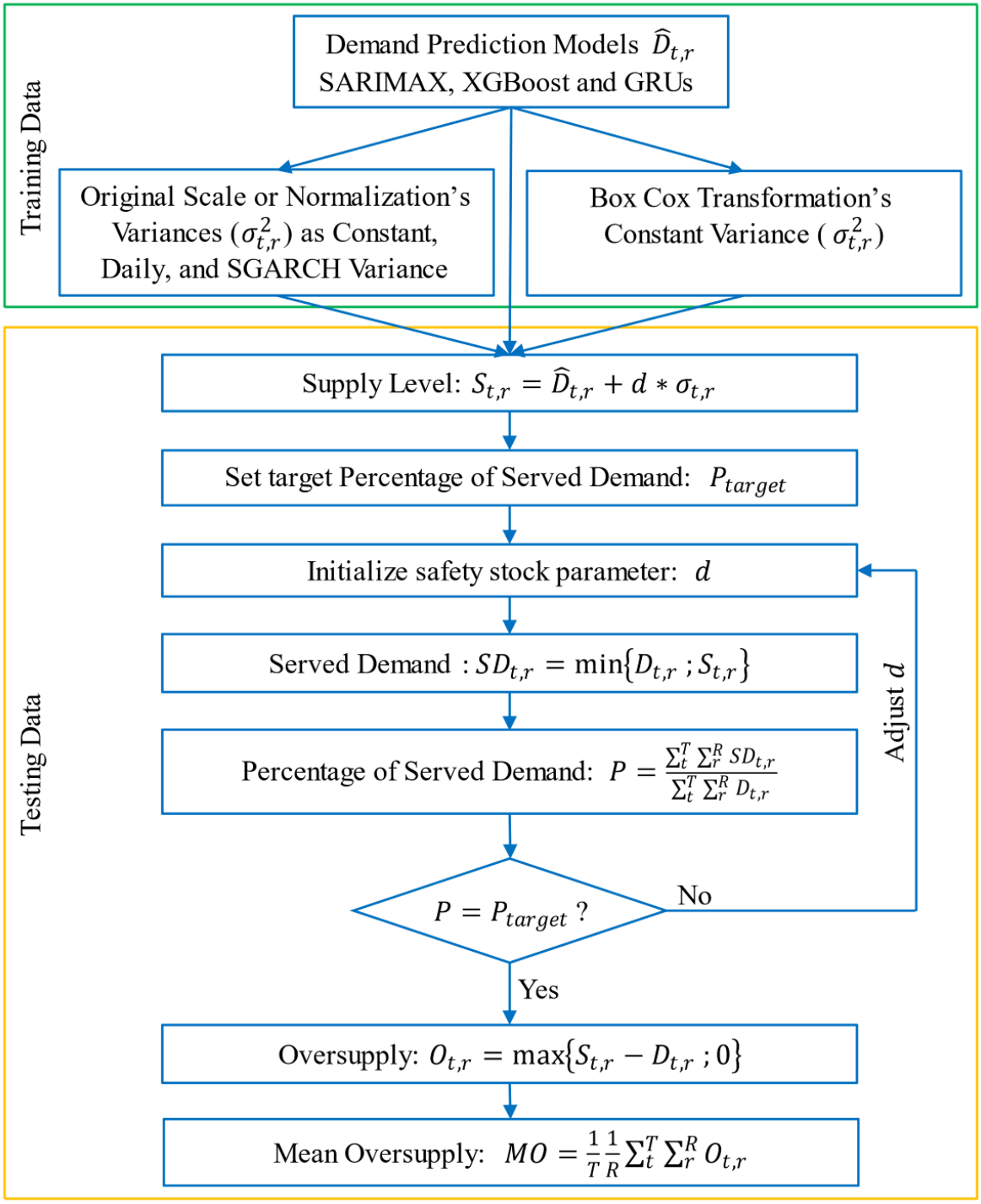

Figure 3.4 Flowchart of supply level models comparison

The served demand in **Eq. 3.21** signifies the lesser value between the actual demand $D_{(t,r)}$ and the supply level $S_{(t,r)}$. In cases where the supply level falls below the actual demand, a portion of the demand remains unmet (i.e., $SD_{(t,r)} = S_{(t,r)}$). Conversely, if the supply level surpasses the actual demand, a surplus of supply—outlined in **Eq. 3.23** (i.e., $SD_{(t,r)} = D_{(t,r)}$ and $O_{(t,r)} = S_{(t,r)} - D_{(t,r)} \geq 0$)—is incurred. In **Eq. 3.22**, the percentage of served demand signifies the ratio of the total projected served demand to the overall actual demand. Given that the total actual demand constitutes the summation of both served and unserved demand, the percentage of unserved demand corresponds to one minus the percentage of served demand $(1 - P)$. Consequently, an efficient supply level model is characterized by the smallest mean oversupply, as detailed in **Eq. 3.24**, while concurrently preserving an equivalent percentage of served (or unserved) demand. It's noteworthy that at a particular percentage of served demand, distinct supply level models may exhibit varying safety stock parameter values, as depicted in **Figure 3.4**. Here, R represents the total count of regions, while T a signifies the total count of time intervals.

3.3 Data collection and featuring

In this study, an exploration was undertaken encompassing three distinct datasets. Initially, data was procured from Neuron Mobility, the operator overseeing shared e-scooters within Thammasat University's Rangsit Campus in Thailand. The additional datasets were sourced from publicly accessible platforms relating to Austin, Texas, and Minneapolis, Minnesota, within the United States. In the instance of Austin, a preprocessing step was employed wherein trips during the initial months of introduction were excluded, primarily due to concentrated operations within the downtown region. The identification and removal of anomalous trips were conducted through a series of criteria, including criteria based on trip duration (below 30 seconds or exceeding two hours), trip distance (below 20 meters or beyond 10 kilometers), and date (falling outside the final date boundary). Referring to the concluding date in **Table 3.1**, the cumulative samples for Thammasat, Minneapolis, and Austin were established as 2,352 (24 hours multiplied by 98 days), 4,704 (24 hours multiplied by 196 days), and 13,680 (24 hours multiplied by 570 days), respectively.

Table 3.1 Dataset's information

Description	Thammasat (TH)	Minneapolis (MN)	Austin (TX)
Start Date	23-Jan-19	14-May-19	1-Aug-18
End Date	30-Apr-19	25-Nov-19	21-Feb-20
# Days	98	196	570
# Trips	29,132	913,781	8,689,720
# Time Intervals (T)	2,352	4,704	13,680
# Regions (R)	1	15	30
Trip Distance (km)	1.3	1.7	1.5
Trip Duration (min)	11.6	12.7	10.4

In this experimental context, the term "demand" pertains to the collective count of pick-up trips occurring within a specific time interval (1 hour) and region. For the Thammasat Rangsit

Campus, which spans approximately 3.21 square kilometers, our focus encompassed generating demand predictions for the comprehensive area. Concerning the Austin dataset, trips were categorized based on census tracts, each averaging an area of 2.05 square kilometers. While shared e-scooter operations extended across more than 50 census tracts within the Austin metropolitan zone, our analysis exclusively targeted the top 30 censuses characterized by an average hourly demand exceeding one trip. A substantial discrepancy in demand was observed between the downtown census and other censuses, with average hourly demands measuring around 208 and 10, respectively (refer to **Figure 3.5**). In the Minneapolis dataset, trip locations were recorded in terms of street names, thereby necessitating the utilization of street centers as trip coordinates. To introduce diversity in spatial clustering, the *k*-means algorithm was implemented for trip grouping in Minneapolis. By adhering to the Elbow method, the optimal number of spatial clusters was determined as 15. Consequently, the average area of these clusters approximated 10 square kilometers, albeit the inner clusters—characterized by denser trip activity—were proportionately smaller than the outer regions.

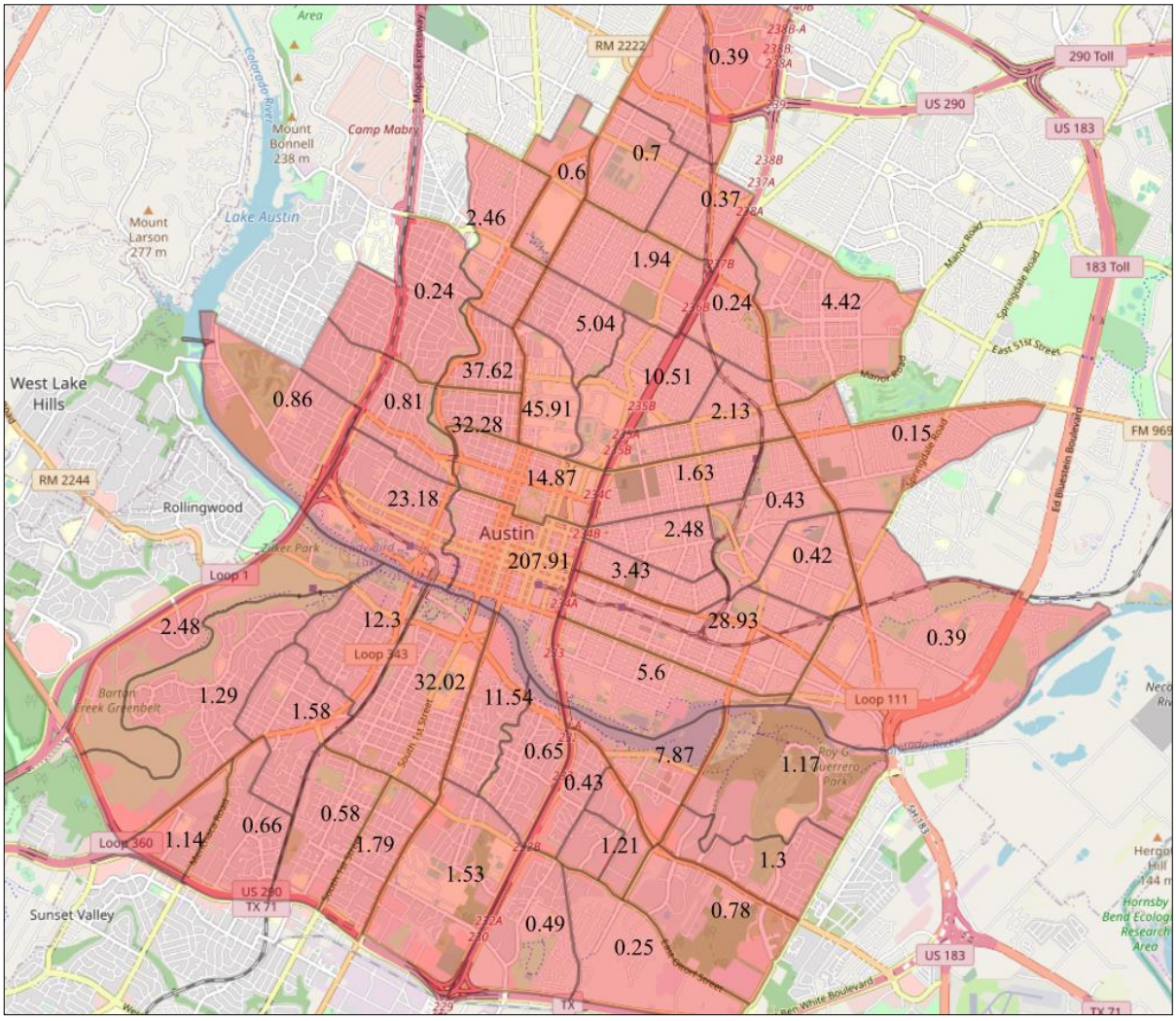

Figure 3.5 Average hourly e-scooter demand by census in Austin, Texas

Much like the impact of weather conditions on shared bicycles, the utilization of e-scooters is similarly influenced. Weather Underground, a worldwide meteorological network, furnishes

an array of weather-related variables at hourly intervals (www.wunderground.com). Notably, this historical weather data was exclusively gathered from international airports, thereby occasionally engendering significant geographic disparity from the area of study's center. Seven pertinent weather attributes were sourced from Wunderground for training purposes, encompassing temperature, precipitation, wind speed, humidity, wind gust, atmospheric pressure, and dew point. In instances of missing values, this research adopted the approach of linear interpolation to ensure data completeness.

Figure 3.6 illustrates anomalous trends in shared e-scooter demand. Upon closer examination, these heightened demand instances were discernibly associated with specific annual festivals or gatherings. In the context of Austin, notable annual events included the Annual SXSW, Pecan Street Festival, H-E-B Austin Symphony, and City Limits Music Festival. Similarly, in Minneapolis, peak ridership aligned with events such as OpenStreets, Pride Festival Parade, Stone Arch Bridge Festival, Uptown Art Fair, and State Fair Festival. Notably, at Thammasat University, an upsurge in demand was evident during the two-day "Season Market" promotional event. The influence of public holidays was also noteworthy, particularly in the Thammasat dataset, where a marked decline in ridership was observed owing to reduced student presence on campus during such holidays.

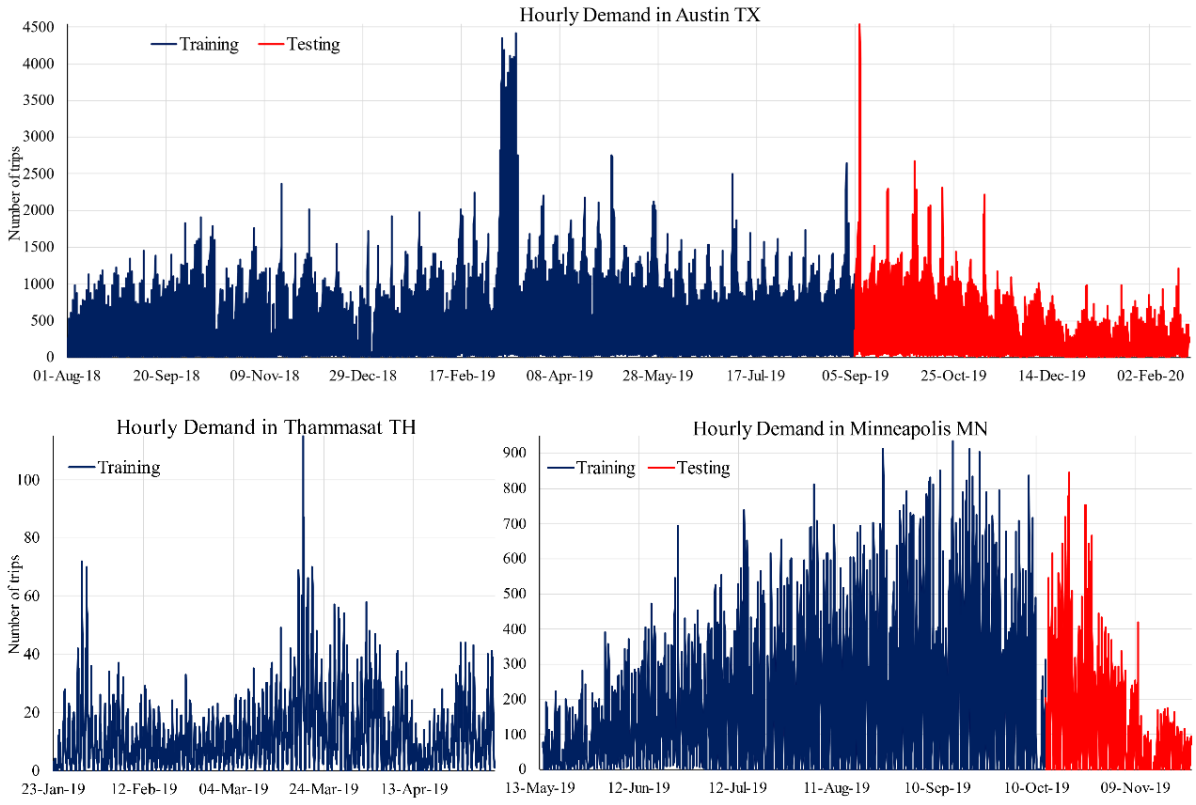

Figure 3.6 Hourly demand of shared e-scooters in Austin TX (*top*), Thammasat University (*bottom-right*), and Minneapolis MN (*bottom-left*)

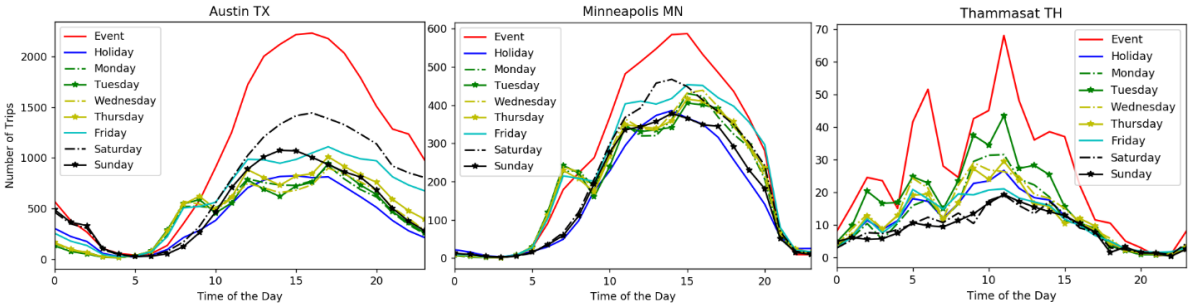

Figure 3.7 Average hourly demand of shared e-scooters by day of the week, public holiday, and annual events (festival or fair)

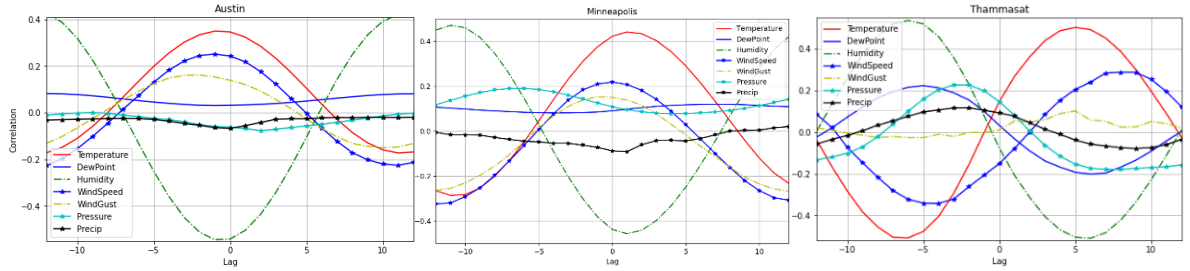

Figure 3.8 Lag-wise Pearson correlation of weather's attributes on shared e-scooter demand

Beyond the recurring daily and weekly patterns, the usage trends of shared e-scooters exhibited marked seasonality. This was evident through the pronounced surge in demand during summer, juxtaposed with a relative decline during winter. Notably, the operators in Minneapolis were compelled to suspend operations during the winter season due to safety concerns, a trend characterized by a gradual decline in ridership leading up to the onset of snowy conditions. Furthermore, shared e-scooter operators adhered to advisory measures, such as postponing operations during significant events like the state visit of the US president to Minneapolis on October 10, 2019. Consequently, these characteristics were recorded as binary attributes, encompassing factors such as annual events or fairs, public holidays, time of day, day of the week, day of the month, and temporary suspensions (specifically in Minneapolis). From a short-trip perspective, the average journey distance for Thammasat, Austin, and Minneapolis stood at approximately 1.3, 1.5, and 1.7 kilometers, respectively. Additionally, riders spent an average of about 11.6, 10.4, and 12.7 minutes on the e-scooter. By considering the average fare [42], the revenue generated per trip in these respective cities amounted to around 1.75, 2.56, and 2.91 US dollars. At this pricing structure, the cost per e-scooter trip was relatively higher compared to shared bikes, a likely factor contributing to the limited popularity of shared e-scooters for commuting purposes.

As shown in **Figure 3.7** for Minneapolis, the ridership patterns manifest a notable resemblance from Monday to Thursday, while the ridership on Friday is relatively higher that other weekdays, specifically during the evening. The ridership in this city on Saturday is marginally high during the afternoon, but this ridership is lower than that on Friday during the evening. This phenomenon can be attributed to individuals opting for e-scooters to engage in leisurely pursuits following an exhaustive workweek, especially Friday evening and night. On Sunday, the ridership on this day in Minneapolis exhibits a pattern akin to weekdays, but with a relatively diminished nighttime demand compared to the rest of the week. For the Austin

dataset, the ridership patterns on weekdays closely resemble those observed in Minneapolis. However, demand in Austin experiences a significant increase on Saturdays in contrast to other weekdays, while Sunday's ridership mirrors that of typical weekdays, except for a slightly heightened demand in the afternoon. From these two datasets, a discernible divergence between weekdays and weekends is noticeable, featuring a minor morning peak. This indicates the utilization of shared e-scooters for commuting purposes, albeit at a relatively modest ratio. During public holidays, Austin witnesses subdued demand in comparison to regular days, yet still surpassing the majority of weekdays during the afternoon, while the demand trend in Minneapolis echoes that of Sundays. Notably, both cities experience a substantial surge in demand during annual festival days, with Austin's demand reaching nearly double that of regular days.

At Thammasat University, weekend demand exhibits a relatively diminished trend compared to weekdays, with Friday afternoons registering lower demand compared to other weekdays. This pattern underscores the interrelation between e-scooter demand and the presence of students and faculty on the campus. Typically, demand peaks on Tuesdays, surpassing other weekdays. Furthermore, ridership also corresponds to student activities—demand rises from early morning until the afternoon but notably declines around 7 am, coinciding with students being primarily engaged in class. Analogous to the preceding datasets, Thammasat's ridership experiences significant surges during annual events.

Figure 3.8. illustrates the Pearson correlation between the prevailing ridership and lagged weather attributes. In this context, both Austin and Minneapolis exhibit a congruent trend wherein the current demand displays a robust correlation with weather attributes, notably temperature and humidity. The correlation coefficients adopt a contrasting pattern that reflects the daily cycle. Conversely, Thammasat's current demand manifests a weak correlation with contemporaneous weather conditions, yet a strong correlation with past or future weather conditions. This discrepancy can be attributed to distinct peak demand periods for e-scooters around midday, as opposed to the temperature's peak occurring at approximately 3 pm. Consequently, historical weather attribute data were also incorporated into the demand prediction models, with a specific focus on machine learning and deep learning models.

The demand patterns delineated earlier influenced the selection of inputs for the demand prediction models, as succinctly outlined in **Table 3.2**. Given the presence of both daily and weekly seasonal fluctuations, the temporal scope for demand prediction encompassed a span of 24 to 168 hours (equivalent to 24 times 7 days). Notably, as delineated in **Table 3.2**, the application of Box Cox transformation (# Trips BC) demonstrated a marked reduction in the volatility of hourly demand in comparison to the original scale (# Trips). Three input components—historical average of overall demand (encompassing weekly, holiday, and event patterns) —are particularly pivotal for the SARIMAX model, given its inherent limitation in accommodating the binary values of these exogenous variables. Furthermore, the temporary ban was imposed only in Minneapolis, this attribute was incorporated into the prediction models as a binary variable.

Table 3.2 Description of inputs for demand prediction models

Inputs	Thammasat (TH)	Minneapolis (MN)	Austin (TX)
# Trips	11.59 ± 11.65	12.50 ± 22.39	16.91 ± 59.98
# Trips BC	2.72 ± 1.55	1.65 ± 1.73	1.88 ± 2.56
Temperature	30.46 ± 3.31	15.96 ± 9.33	19.70 ± 9.37
Dew point	23.48 ± 2.89	9.29 ± 8.81	13.24 ± 8.84
Humidity	68.28 ± 16.05	66.89 ± 16.22	69.96 ± 20.27
Wind speed	12.15 ± 5.00	14.26 ± 7.78	13.01 ± 9.23
Wind gust	0.06 ± 2.19	7.66 ± 16.82	4.95 ± 13.46
Pressure	1010.24 ± 2.78	983.87 ± 6.31	996.93 ± 13.38
Precipitation	0.12 ± 0.46	0.16 ± 1.09	0.10 ± 1.03
HAO weekly	11.88 ± 8.78	192 ± 159.68	527.44 ± 369.44
HAO holiday	11.63 ± 7.68	157.98 ± 141.91	409.31 ± 282.48
HAO event	25.38 ± 18.14	253.53 ± 220.21	1082.10 ± 793.52
Hour of day	0 - 1	0 - 1	0 - 1
Day of week	0 - 1	0 - 1	0 - 1
Day of month	1 - 31	1 - 31	1 - 31
Holiday	0 - 1	0 - 1	0 - 1
Event	0 - 1	0 - 1	0 - 1
Ban	-	0 - 1	-

Note: ***HAO***: Historical Average of Overall demand, & ***BC***: Box Cox scale

To ensure the generalizability of the models, the whole datasets were partitioned into two segments: the training subset (in-sample) and the testing subset (out-of-sample). The initial 75% of the dataset was allocated for training the models, while the remaining 25% was reserved for evaluation purposes (refer to **Figure 3.6**). From **Figure 3.7**, volatility remains low during periods of steady demand but escalates proportionally with heightened demand levels. Similarly, numerous leisurely events, festivals, and fairs take place during the summer, resulting in increased and fluctuating demand for shared e-scooters. Consequently, 75% of the training subset was randomly designated for model training, and the remaining 25% was set aside for model assessment. This random partitioning approach was chosen due to its significantly reduced computational times in comparison to techniques like K-Folding, particularly during hyperparameter optimization. Furthermore, it allows the models to learn explanatory variables (such as events, holidays, and bans) that pertain to specific dates, which might be omitted if employing conventional time-series splitting. However, due to the relatively modest size of the Thammasat Dataset, all available data were utilized for both model training and evaluation purposes.

3.4 Demand and variance prediction

3.4.1 Demand prediction results

Aligned with the research framework depicted in **Figure 3.1**, six predictive models were employed to forecast the hourly ridership of shared e-scooters. These models comprised SARIMAX, RF, XGBoost, FCNNs, RNNs, and GRUs. The deep learning models (FCNNs,

RNNs, and GRUs) were implemented using Keras and TensorFlow, Python libraries, within the Jupyter Notebook environment. Subsequently, Random Forest and XGBoost were trained utilizing the Scikit-learn and XGBoost libraries, respectively. Lastly, SARIMAX was trained using the STATA statistical software as it facilitated the out-of-bag evaluation process.

The description of the GRU cell formulation was presented in **Section 3.2.3.4**. The architectural configuration of GRUs is illustrated in **Figure 3.9**, encompassing the input layer with GRU nodes, a single dropout layer, a cluster of hidden layers with GRU nodes, and the output layer with conventional neurons. In **Figure 3.9**, input sequences are sequentially arranged before entering the input layer, and the outputs from this layer are subject to dropout at a specific rate to enhance learning efficacy with smoother learning curves. Uniform activation functions and node counts were applied across the hidden layers. Within this architecture, the output layer may consist of one or multiple neurons, permitting training of temporal demand either spatially independently or spatially combined, respectively. Both spatially independent and spatially integrated configurations were explored, with the optimal outcome chosen. These two training methodologies each possess merits and drawbacks. Spatially independent training enables models to attain optimal learning curves unhindered, albeit at the cost of disregarding valuable information from neighboring regions. Conversely, the model featuring multiple spatial outputs capitalizes on shared correlated information across regions to enhance prediction performance, albeit necessitating careful optimization for best results.

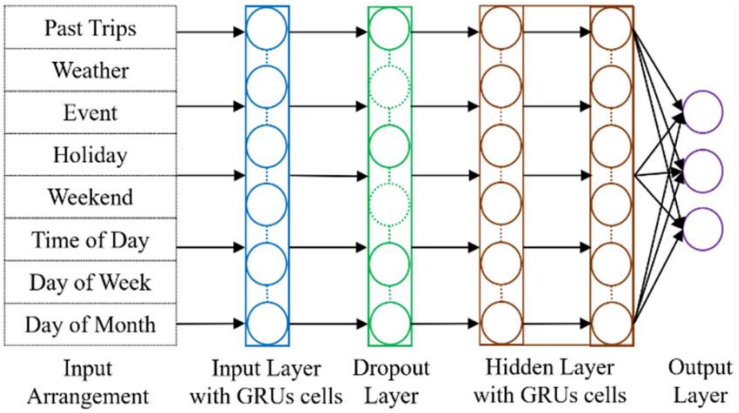

Figure 3.9 The proposed architecture of GRUs model

While deep learning models can surpass traditional probabilistic models and machine learning algorithms in performance, they also necessitate labor-intensive hyperparameter optimization. Hyperparameter Optimization (HPO) involves fine-tuning aspects such as the number of GRU nodes within the input layer, dropout rate, and the number of hidden layers, among others. Various strategies were employed during this phase, encompassing grid search, random search, and automatic optimization techniques like Bayesian Optimization, Tree-structured Parzen Estimator, and genetic algorithms, among others. Bayesian Optimization (BO) holds prominence as a sequential optimization methodology well-suited for resource-intensive problems, particularly those inherent to deep learning. BO relies on two crucial components: a surrogate function employing Gaussian Processes, and an acquisition function using Upper Confidence Bound to balance exploration and exploitation. To optimize the GRU configurations, Keras Tuner [43], a Python package for Bayesian optimization-based HPO, was

employed. This process was executed within the Keras and TensorFlow frameworks, operating through Jupyter Notebook. Default settings were employed for all Bayesian Optimization parameters, with the validation loss serving as the objective function. The initial points were set to 10, and the maximum iterations were capped at 80. Furthermore, the number of epochs was adjusted through early stopping criteria, incorporating a patience value of 10 and a maximum epoch limit of 150.

In this research, Bayesian Optimization (BO) was applied to fine-tune nine pivotal hyperparameters of the Gated Recurrent Units (GRUs), encompassing aspects such as the lookback length, input layer's activation function and number of GRU nodes, dropout rate, number of hidden layers, hidden layers' activation function and number of GRU nodes, output layer's activation function, and batch size (as outlined in **Table 3.3**). The HPO procedure was carried out in a series of sequential steps due to several factors (refer to **Figure 3.10**): prolonged training time stemming from the lookback length and number of hidden layers, challenges related to local optima, iterations failing to converge, and instances of exploding iterations where the loss function attains an infinite value. Initially, deep learning models with a solitary hidden layer were independently optimized for diverse lookback lengths (24, 48, …, 168) to identify the most optimal lookback length. Subsequently, for each designated lookback length, the BO algorithm, employing the aforementioned configuration, sought to minimize the validation loss by manipulating factors such as the number of nodes per layer, dropout rate, activation function of each layer, and batch size. Following the determination of the optimal lookback length, the configuration of deep learning models underwent re-optimization to account for a higher number of hidden layers. Within this optimization, three activation functions—ReLU, Tanh, and Sigmoid—were considered, alongside dropout rates ranging from 0.00 to 0.40 in increments of 0.01. The number of nodes per layer was confined between 10 and 500 with an interval of 10. The batch size was examined within the range of 4 to 1000. The remaining parameters of the GRUs retained their default values, encompassing the optimizer (Adam), learning rate (0.001), and the employment of Mean Squared Error (MSE) as the loss function.

The predictive efficacy of the Gated Recurrent Units (GRUs) was juxtaposed against five additional benchmark models, namely SARIMAX, RF, XGBoost, FCNNs, and RNNs. Moreover, the influence of Box Cox transformation on Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) was demonstrated by incorporating the Historical Average (HA) model. The other five models for demand prediction underwent optimization using BO for FCNNs and RNNs, and grid search for SARIMAX, RF, and XGBoost. This optimization process is outlined in **Table 3.3**. Notably, SARIMAX, RF, and XGBoost models were trained individually for each distinct region, while FCNNs and RNNs shared the same configurations as GRUs. To mitigate overfitting issues, especially prevalent in RF and XGBoost, the difference between training and validation loss was maintained at approximately 15%. A brief overview of these five baseline models is presented below:

- **SARIMAX**: As evidenced in **Figure 3.7**, a distinct daily and weekly pattern is observed. Hence, SARIMAX was structured to incorporate a daily seasonality (i.e., S=24). This involved integrating three exogenous variables within the SARIMAX model, namely the hourly average demand categorized by day of the week, public holidays, and events. To facilitate the out-of-bag evaluation, SARIMAX was trained using a statistical program,

STATA. As outlined in **Table 3.3**, a grid search was executed to optimize six parameters of SARIMAX. These encompassed the degree of differencing (*d*), deseasonalizing degree (*D*), seasonal (*P*) and non-seasonal (*p*) autoregressive lag polynomial, and seasonal (*Q*) and non-seasonal (*q*) moving average lag polynomial. Importantly, all parameters, including the exogenous variables, needed to be statistically significant at a 95% confidence level. Ultimately, the model exhibiting the smallest Root Mean Squared Error (RMSE) was selected.

- **RF**: This model was trained in a spatially independent manner for each spatial (i.e., region) demand, where historical ridership and other relevant features were flattened prior to model training. The tuning of RF involved a grid search for three key hyperparameters: the lookback length (ranging from 24 to 168), the number of trees in the forest (ranging from 10 to 500), and the maximum depth of the trees (ranging from 0 to 15).
- **XGBoost**: This model underwent a training process similar to that of RF.
- **FCNNs**: The architecture of FCNNs closely resembled that of GRUs (refer to **Figure 3.9**), employing simple neurons in each node. As a reference model, FCNNs were configured with a maximum of 2 hidden layers and up to 100 nodes per layer, adhering to the conventional architecture [3, 44]. Other hyperparameters of FCNNs were optimized using BO, as outlined in **Table 3.3**. In this context, FCNNs were trained both spatially independently and in a combined manner, with the selection of the optimal models.
- **RNNs:** The training process for RNNs resembled that of FCNNs, except that a simple RNN cell was used in each node.

Table 3.3 Description of hyperparameter optimization for demand prediction models

Model	Parameters	Value Range	Tuning
GRUs RNNs FCNNs	Lookback length (LL)	24, 48, …, 168	Bayesian Optimization
	Activation input layer	Relu, Tanh, Sigmoid	
	# Nodes input layer	10, 20, 30, …, 500	
	Dropout rate	0.0 - 0.40	
	# Hidden layer (#HL)	1 - 5	
	Activation of hidden layer	Relu, Tanh, Sigmoid	
	# Nodes in hidden layer	10, 20, 30, …, 500	
	Activation output layer	Relu, Tanh, Sigmoid	
	Batch size	4 - 1000	
XGBoost	Lookback length	24, 48, …, 168	Grid Search
	# Gradient-boosted trees	10, 15, 20, …, 500	
	Max-depth of tree	0, 1, 2, …, 10	
Random Forest	Lookback length	24, 48, 72, …, 168	Grid Search
	# Trees in the forest	10, 15, 20, …, 300	
	Max-depth of tree	0, 1, 2, …, 15	
SARIMAX (p, d, q) * (P, D, Q, 24)	p	0 - 5	Grid Search
	d	0 - 2	
	q	0 - 5	
	P	0 - 2	
	D	0 - 2	
	Q	0 - 2	

Based on our analysis, it was evident that the sigmoid activation function required nearly twice the number of epochs compared to the Tanh or ReLU functions, although it exhibited a smoother learning curve. Furthermore, the Austin dataset necessitated spatially independent training, while the Minneapolis dataset was trained with multiple spatial outputs. The outcomes of hyperparameter optimization for GRUs using the original scale, guided by BO, are depicted in **Figure 3.10**, specifically for the Downtown Census in Austin, TX. The optimal lookback length was determined to be 120, and three hidden layers demonstrated superior prediction performance. Notably, training GRUs with original data resulted in improved outcomes compared to their normalized counterparts, particularly with respect to model generalization. This observation was underpinned by the fact that the optimal architectures for normalized data frequently employed Tanh or Sigmoid activation functions, leading to effective learning of the training data and susceptibility to overfitting issues, especially when contrasted with benchmark models. These findings aligned with prior research [34], where it was demonstrated that Box Cox transformation facilitated simpler models in comparison to the original scale. This was evident in SARIMAX models, where many exogenous variables turned out to be insignificant. Additionally, the optimal GRUs model for Austin data featured two hidden layers for the original scale, as opposed to only one for data transformed with Box Cox. Consequently, Box Cox transformation proves to be advantageous for deep learning applications, as it can curtail training times, especially during hyperparameter tuning.

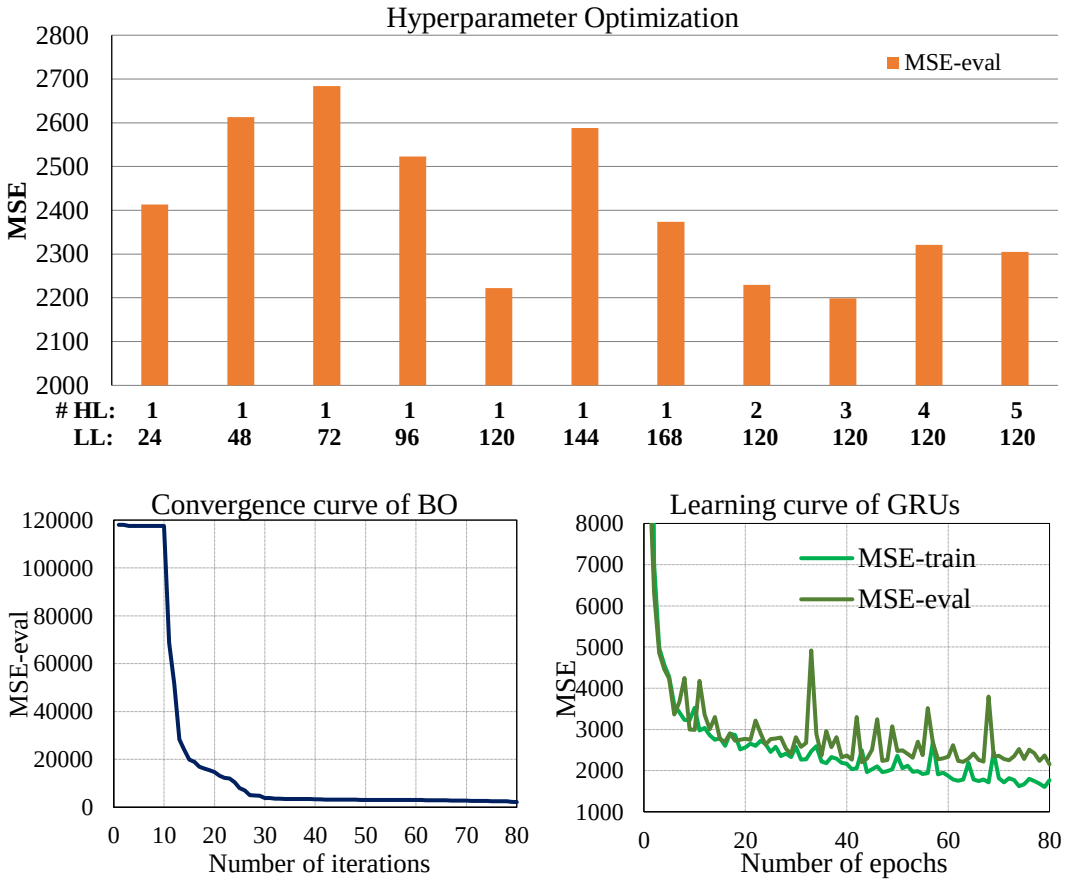

Figure 3.10 Hyperparameter optimization of GRUs by BO for Downtown Census in Austin, TX

Table 3.4 presents a comprehensive comparison of performance between Box Cox transformed data and the original or normalized counterparts (excluding the Thammasat Dataset due to a lack of testing data). For both original and normalized data, deep learning models exhibited enhancements in prediction performance, as reflected in the RMSE and MAE metrics. Notably, these improvements were significantly influenced by the number of tuned hyperparameters. Box Cox transformation showcased similar trends in the Austin and Thammasat datasets, whereas its impact differed in the Minneapolis dataset. Functioning as a generalized logarithmic transformation, Box Cox effectively compressed the atypical demand outliers towards the mean, thereby fostering model simplification and an increase in accuracy, particularly with the MAE metric. This phenomenon is evident in **Table 3.2**, which highlights that the ratio of demand's mean to standard deviation between the original and Box Cox scales stood at approximately 7 and 15, respectively. Furthermore, **Table 3.4** underscores that while the RMSE of the Historical Average (HA) in the original scale was lower than that in the Box Cox scale, the converse was true for the MAE metric. The impact of Box Cox transformation on demand volatility is the rationale behind the relatively inferior performance of deep learning models for the Minneapolis dataset compared to SARIMAX. This is attributed to the fact that Box Cox transformation rendered temporal information from neighboring regions less pertinent. Consequently, it is evident that Box Cox transformed data from Minneapolis should be trained spatially independently. In essence, Box Cox transformation is particularly advantageous when dealing with datasets characterized by high irregularity or a limited number of exogenous variables. In summary, across all datasets, Box Cox transformation yielded reductions of 0.14% and 5.36% in the RMSE and MAE metrics, respectively.

Table 3.4 Performance comparison based on RMSE and MAE

Dataset	Models	Original or Normalized Data				Box Cox Transformed Data			
		RMSE-Eval.	RMSE-Test	MAE-Eval.	MAE-Test	RMSE-Eval.	RMSE-Test	MAE-Eval.	MAE-Test
Thammasat Thailand	GRUs	5.27	-	3.41	-	5.18	-	3.37	-
	RNNs	5.52	-	3.75	-	4.91	-	3.40	-
	FCNNs	5.52	-	3.76	-	5.00	-	3.46	-
	XGBoost	5.21	-	3.64	-	5.17	-	3.46	-
	RF	5.30	-	3.72	-	5.31	-	3.56	-
	SARIMAX	5.47		3.82		5.26		3.62	
	HA	11.65	-	8.69	-	12.24	-	8.32	-
Minneapolis Minnesota	GRUs	6.96	6.34	3.58	2.89	7.18	6.92	3.67	2.99
	RNNs	7.07	6.25	3.49	2.85	7.75	6.82	3.80	2.95
	FCNNs	7.53	6.48	4.06	3.08	8.38	7.30	3.83	3.33
	XGBoost	7.44	6.44	4.04	3.37	7.16	6.04	3.66	2.73
	RF	7.34	6.39	3.85	3.21	7.35	6.20	3.76	2.81
	SARIMAX	7.79	6.24	4.08	3.07	7.72	6.03	3.85	2.64
	HA	21.21	16.83	12.97	11.08	23.80	17.39	12.28	8.55
Austin Texas	GRUs	11.24	11.28	4.15	3.70	11.20	11.01	4.00	3.54
	RNNs	11.34	11.58	4.21	3.73	11.52	11.96	4.10	3.60
	FCNNs	11.47	11.22	4.16	3.67	12.50	11.86	4.21	3.72
	XGBoost	11.29	11.83	4.29	3.89	13.06	11.75	4.29	3.70
	RF	12.18	12.00	4.31	3.92	12.54	12.21	4.26	3.77
	SARIMAX	12.30	11.13	4.58	3.83	12.60	11.23	4.40	3.54
	HA	50.49	36.44	14.82	12.53	53.95	37.30	14.34	10.90

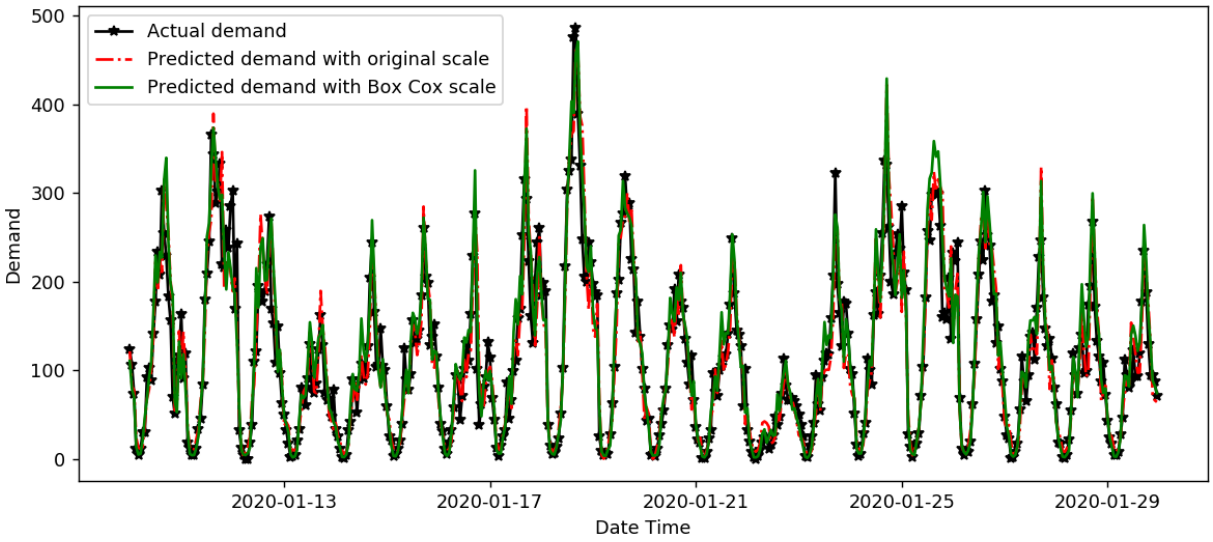

Figure 3.11 Demand prediction by GRUs with original and Box Cox scale for Downtown Census in Austin, TX

SARIMAX exhibited superior predictive accuracy on the testing dataset compared to other models due to the testing data aligning with a period of low demand. Nonetheless, this regression model demonstrated limitations during high-demand periods, such as summer, while GRUs achieved robust performance across both training and testing datasets. **Figure 3.11** visually illustrates a comparison of e-scooter demand predictions using GRUs, considering both the original and Box Cox scales, specifically for the Downtown census in Austin, Texas. Both models demonstrated adeptness in capturing the hourly demand patterns of shared e-scooters. Although variations in prediction results exist between the two models, particularly during peak demand periods, their overall predictive performance remains commendable. Notably, both models accurately predict low-demand periods during the nighttime, yet exhibit diminished performance during the afternoon and evening when demand and volatility are high, as depicted in **Figure 3.12**.

3.4.2 Variance prediction results

Capturing valuable insights from historical data plays a pivotal role in formulating effective operational strategies for dockless shared e-scooters, enabling efficient resource management and cost minimization. While advanced demand prediction models have demonstrated remarkable capabilities in forecasting future demand, inherent uncertainties remain, particularly associated with the residuals of these demand prediction models. Illustrated in **Figure 3.1**, safety stock serves as a customary approach to accommodate these uncertainties, contingent upon the fluctuations in demand. This underscores the importance of variance analysis in crafting an optimized supply level. Two crucial attributes of residuals wield significance in the design of supply levels: their distribution and heteroscedastic nature. Residuals stemming from forecasting models typically adhere to either a Normal or Student's *t*-distribution. This attribute bears relevance in the selection of the confidence level value (Z_{score}) or the cover rate (the proportion of data residing within the confines of the confidence interval). As expounded in **Section 3.4**, the concept of heteroscedasticity pertains to the temporal pattern exhibited by residuals, necessitating that inventory design aligns proportionally with this observed pattern.

Figure 3.12 illustrates the scatter plot and histogram of daily residuals for GRUs in both the original and Box Cox scales. In the case of the original scale, the distribution displayed heavier tails compared to the normal distribution, thereby suggesting the suitability of the student's *t*-distribution for these residuals. On the other hand, for GRUs using the Box Cox scale, the residuals exhibited somewhat thicker tails, albeit to a practically negligible extent. Observing the daily scatter plot, a distinct daily pattern emerged in the residuals for the original scale, whereas this pattern remained nearly constant for the Box Cox scale. To validate the presence of heteroscedasticity in the residuals, an ARCH-LM test was conducted. The outcome revealed the rejection of the null hypothesis (indicating no ARCH effects), given the *p-value* falling below the 5% threshold for both the original and Box Cox scales. It's worth noting, however, that the coefficients of the SGARCH model in the Box Cox scale were relatively minute, thereby allowing for the statistical disregard of ARCH effects [1, 34].

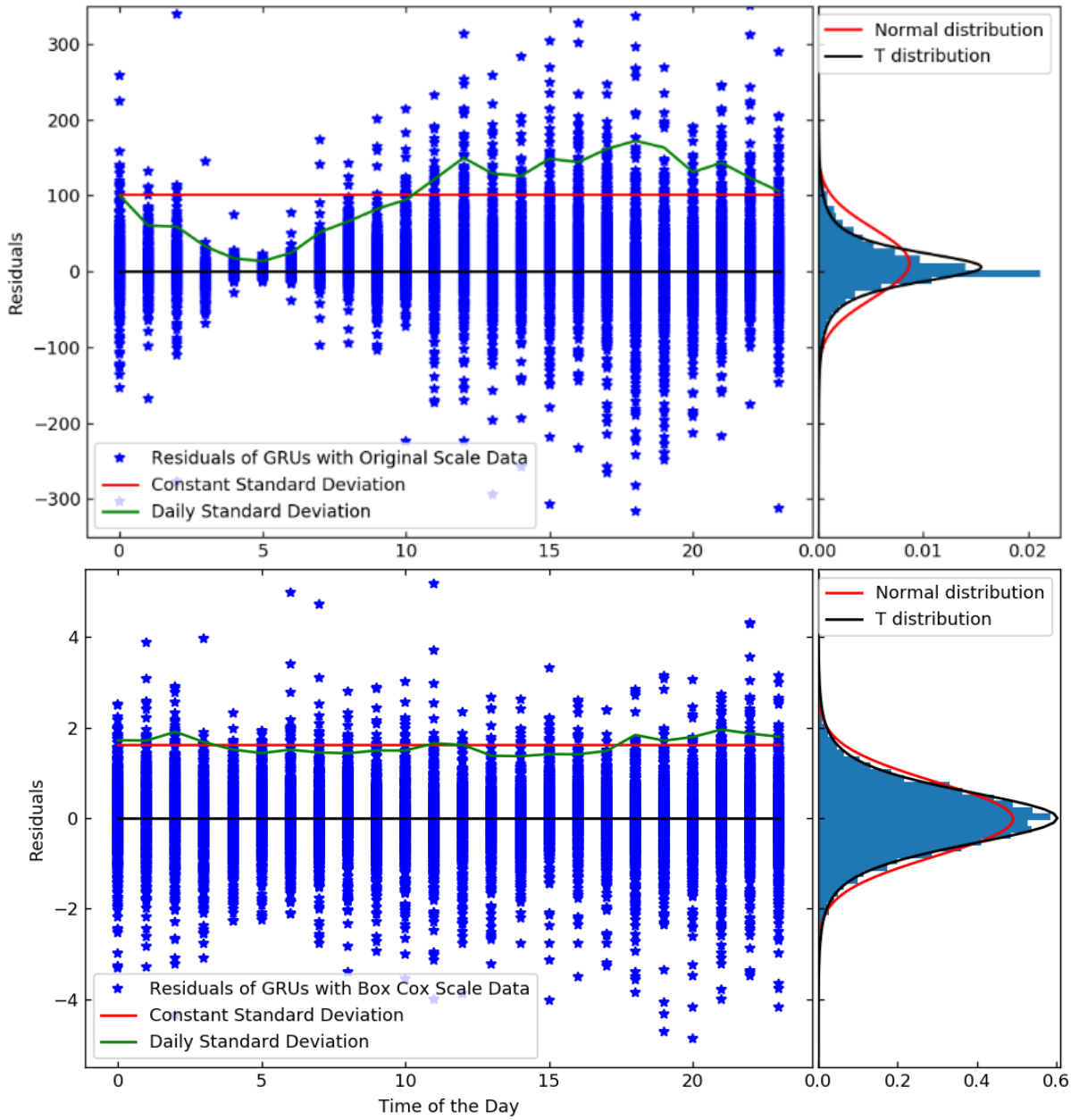

Figure 3.12 Daily scatter plot and histogram of GRUs' residuals for Downtown Census in Austin, TX: (*top*) original data and (*bottom*) Box Cox transformed data

In the context of GRUs using the original scale, as depicted in **Figure 3.12** (*top*), the 97.5% upper Confidence Interval (CI) maintained a consistent standard deviation and yielded a cover rate (Service Level Type I) of 96.79%. The subtle variation was not the primary concern; rather, it was the distribution of residuals surpassing the upper CI. In the initial half (0-11), merely 0.56% of residuals exceeded the upper CI, while in the latter half (12-23), this proportion escalated to 2.65%. This disparity indicated that while the upper CI with a constant standard deviation performed admirably in the first half, its effectiveness waned in the subsequent half. Conversely, the 97.5% upper CI considering the daily standard deviation achieved an overall cover rate of 96.8%, accompanied by outlier residuals (laying above the upper CI) of 1.65% and 1.55% for the first and second halves, respectively. With this cover rate, the percentage of served demand (Service Level Type II) amounted to 99.24% and 99.36% for the upper CI employing constant and daily standard deviation, respectively. Furthermore, the upper CI with a constant standard deviation translated to a supply ratio (the ratio of total supply to total actual demand) of 145%, whereas that of the upper CI considering daily standard deviation was 139.4%. In essence, despite sharing the same cover rate, the upper CI (or supply level) based on the daily standard deviation demonstrated lower inventory (resulting in reduced operational costs) while concurrently exhibiting a higher percentage of served demand (yielding increased trip revenue) compared to the upper CI dependent on a constant standard deviation.

Depicted in **Figure 3.12**, the residuals associated with the original scale persisted in displaying a seasonal pattern, which was corroborated by the ARCH-LM test that established the existence of ARCH effects. To delve further into the variance patterns, the SGARCH model in **Eq. 3.19** was introduced for analysis. The comparison of variance prediction models, namely Constant, Daily Seasonal, and SGARCH, is presented in **Figure 3.13**, focusing on the residuals of GRUs employing the original scale within the Downtown Census of Austin, Texas. This graphical representation elucidates the limitations of constant variance or mean squared error, given their incapacity to capture conditional variance. While the daily seasonal variance does to some extent incorporate the diurnal volatility pattern, it proves insufficiently adaptable to long-term demand fluctuations. Conversely, the predicted variance derived from the SGARCH model exhibits notable flexibility in tracking conditional variance. However, this approach does possess a key drawback: substantial errors are propagated to the subsequent seasonal step. The comparison of four supply-level models of GRUs at a 98% served demand is showcased in **Figure 3.14**. Supply levels characterized by constant variance exhibit excessive oversupply during nighttime demand but fall short of meeting afternoon demand requirements. Conversely, the supply level predicated on Box Cox variance demonstrates commendable performance, albeit with certain peak points stemming from the logarithmic inversion effect. Supply levels grounded in daily and SGARCH variance share a comparable pattern, yet the latter, underpinned by SGARCH variance, more effectively allocates uncertainty across long-term demand fluctuations.

In essence, variance analysis proved essential for the original or normalized datasets, whereas the Box Cox transformed data necessitated only constant variance consideration. Within the context of original or normalized data, three distinct variances were scrutinized: constant, daily, and forecasted variances as determined by the SGARCH model. The study encompassed a comparison of four distinct supply level models for each demand prediction model. Among these, three pertained to the demand prediction model employing the original or

normalized scale, incorporating diverse variance models (constant, daily, and SGARCH-predicted variances). The fourth supply level model corresponded to the demand prediction model utilizing the Box Cox transformed scale in conjunction with constant variance. The supply level designs underwent assessment across three distinct demand prediction models: SARIMAX, XGBoost, and GRUs. These prediction models represent established methodologies in the realms of statistical-based, machine-learning, and deep-learning modeling, respectively.

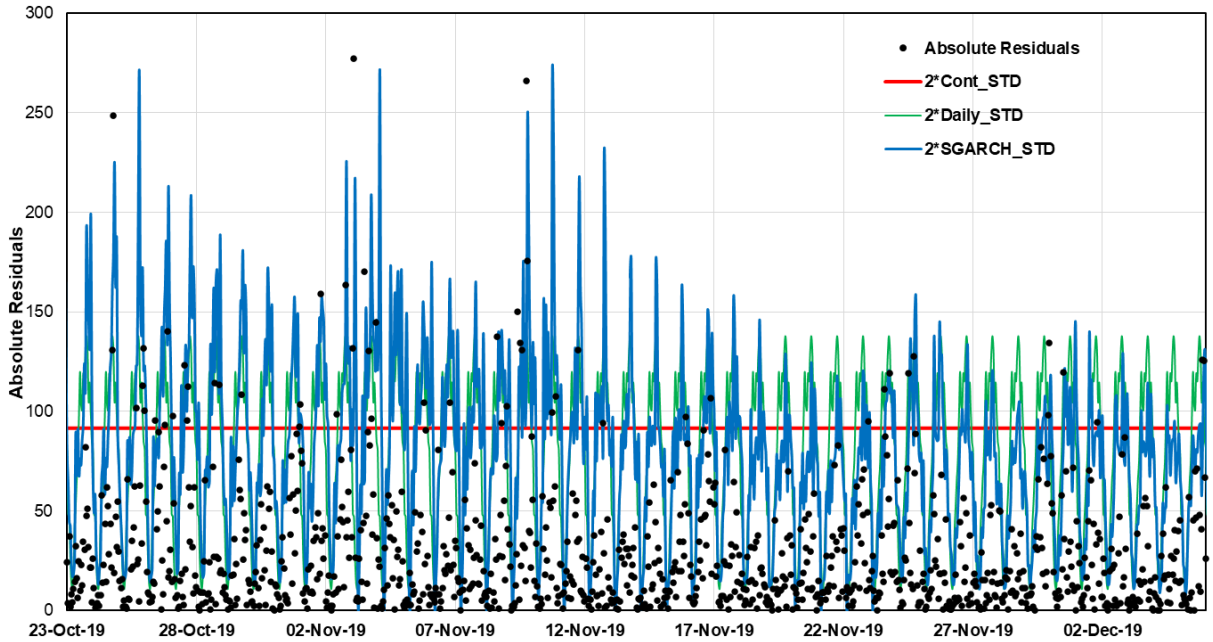

Figure 3.13 Variance prediction for residuals of GRUs with original scale data of Downtown Census in Austin, TX

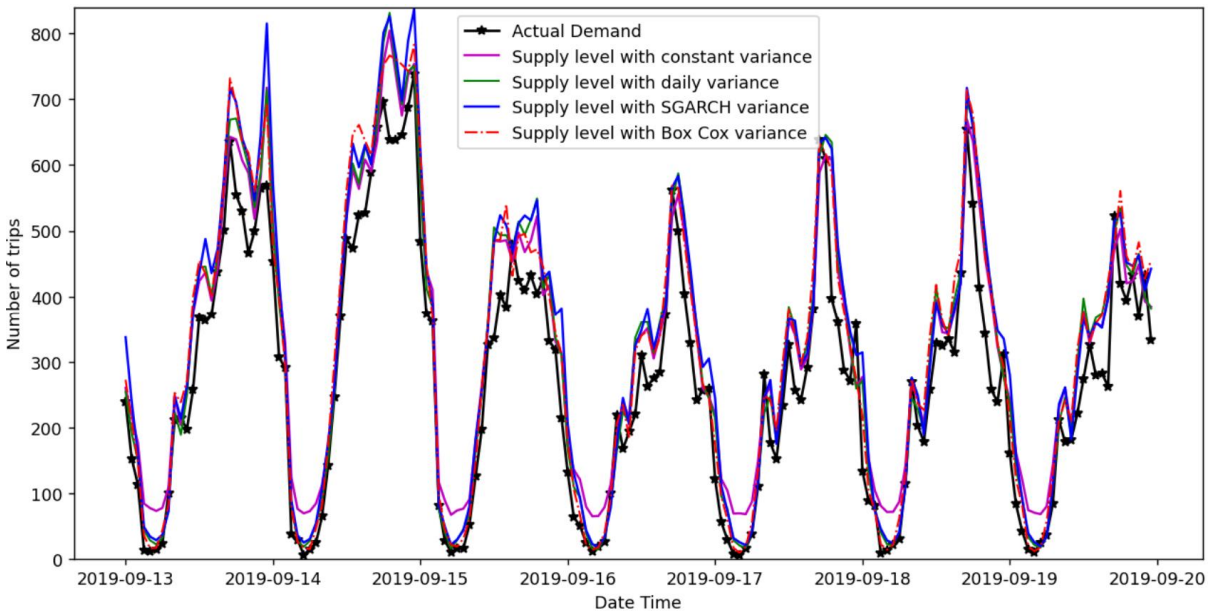

Figure 3.14 Comparison of supply level models of GRUs at 98% served demand (cover rate of around 90%) of Downtown Census in Austin, TX

3.5 Supply planning design

As indicated earlier, confidence intervals prove inadequate for guiding daily operational planning due to their failure to incorporate the magnitude of residuals, particularly in the case of heteroscedastic datasets. Furthermore, different CI models tend to yield varying inventory levels (operational cost) and anticipated served demand (trip revenue), even when maintaining the same cover rate. Hence, this study opted to evaluate supply level models based on an equal percentage of served demand (equivalent number of served or unserved demands) to facilitate a comparison of oversupplies (referred to as MO). To achieve uniform percentages of served demand, the safety stock parameter (d) was individually adjusted for each supply level model, adhering to the guidelines outlined in the flowchart presented in **Figure 3.4**. In practical applications, this parameter should be configured according to the desired service level (consistent with Z_{score}) or iteratively tuned until the supply level attains the maximum number of e-scooters.

The comparison of mean oversupply (MO) in **Table 3.5** was conducted using the training data for the Thammasat dataset, while the remaining two datasets were assessed using the testing dataset. Based on the outcomes observed in the Thammasat dataset, Box Cox transformation exhibited effectiveness in supply level determination, yielding the lowest mean oversupply across models, with the exception of XGBoost. Additionally, the SARIMAX model utilizing Box Cox transformed data demonstrated comparable MO values when compared to the worst-case scenario of GRUs with constant variance. Interestingly, the SARIMAX model even outperformed GRUs in terms of mean oversupply at higher percentages of served demand (95% and above). Overall, GRUs exhibited smaller MO values in comparison to SARIMAX, underscoring the significance of accurate demand prediction. The disparity between the least favorable and most favorable cases of GRUs' supply level models exhibited a noteworthy increase as the percentage of served demand rose, peaking at 98% served demand. To elaborate, operators aiming to achieve a 98% served demand using the supply level model with constant variance encountered an average hourly oversupply of 8 e-scooters. In contrast, adopting the supply level model with Box Cox variance allowed for a reduction in oversupply to approximately seven e-scooters per hour. By implementing this reduction strategy, operators could potentially save up to 30 e-scooters during a 3-hour rebalancing cycle for ten spatial regions, thereby eliminating the need to relocate these redundant 30 e-scooters.

In this comparative analysis of the Minneapolis dataset, the trend of the MO metric mirrored that of the accuracy performance. Specifically, SARIMAX and XGBoost demonstrated good performance with Box Cox transformation, whereas GRUs exhibited lower MO with the original scale. The variation in MO was subtle at lower percentages of served demand (or $d < 0$), but it became notably pronounced at higher percentages. This dataset also underscored the limitations of Box Cox transformation, as it led to elevated MO values at 98% served demand. This outcome could be attributed to the impact of the exponential transformation, especially when the lambda (λ) value approaches -1, which amplifies the discrepancy between the Box Cox scale and the conversion scale (as illustrated in **Figure 3.15**). Hence, it becomes imperative to cautiously restrict the maximum value of the designed supply level, denoted as $S_{(t,r)}$, for Box Cox transformed data. Despite SARIMAX achieving the lowest MO in this dataset due to its superior demand prediction on the testing dataset, it is conceivable

that GRUs would excel in both demand prediction accuracy and mean oversupply during periods of heightened demand, such as the summer season.

The application of Box Cox transformation yielded minimal MO values in the Austin dataset for served demands up to 90%. However, akin to the Minneapolis scenario, this technique encountered challenges at higher served demand percentages. Nevertheless, the performance of the predicted variance was commendable. At a served demand of 95%, GRUs exhibited a reduction of approximately one MO unit between constant variance and predicted variance models. This indicates that operators could potentially save around 30 e-scooters per hour (equivalent to 720 e-scooters in daily rebalancing operations). This gain could be substantially magnified with an increase in the number of regions or a longer rebalancing interval; for instance, a reduction of up to 50 e-scooters per hour for 50 regions or approximately

Table 3.5 Mean oversupply comparison for four supply level models

Dataset	Supply Level Model	Mean Oversupply by Percentage of Served Demand							
	Demand Model	Variance Model	70%	75%	80%	85%	90%	95%	98%
Thammasat Thailand	GRUs	Constant Variance	0.615	0.835	1.179	1.710	2.707	4.880	8.069
		Daily Variance	0.586	0.816	1.160	1.704	2.629	4.490	7.130
		SGARCH Variance	0.591	0.815	1.181	1.704	2.605	4.458	7.259
		Box Cox Variance	**0.465**	**0.695**	**1.069**	**1.631**	**2.557**	**4.330**	**7.091**
	XGBoost	Constant Variance	0.546	0.774	**1.106**	**1.667**	2.754	4.928	8.184
		Daily Variance	0.546	0.793	1.138	1.670	2.601	4.441	7.075
		SGARCH Variance	0.577	0.814	1.147	1.668	**2.566**	**4.313**	**6.909**
		Box Cox Variance	**0.527**	**0.773**	1.132	1.672	2.572	4.385	6.951
	SARIMAX	Constant Variance	0.745	1.029	1.414	2.029	3.063	5.304	8.974
		Daily Variance	0.772	1.046	1.418	2.021	2.941	4.779	7.563
		SGARCH Variance	0.721	0.992	1.401	2.030	3.022	5.022	7.831
		Box Cox Variance	**0.631**	**0.897**	**1.282**	**1.875**	**2.813**	**4.603**	**7.304**
Minneapolis Minnesota	GRUs	Constant Variance	0.582	**0.803**	**1.142**	1.980	3.562	6.894	11.845
		Daily Variance	0.597	0.821	1.155	1.676	2.700	4.916	8.202
		SGARCH Variance	0.594	0.829	1.155	**1.653**	**2.516**	**4.412**	**7.504**
		Box Cox Variance	**0.573**	0.815	1.200	1.823	2.948	5.868	11.484
	XGBoost	Constant Variance	0.538	0.785	1.185	1.849	3.428	6.892	12.078
		Daily Variance	0.589	0.855	1.242	1.891	3.024	5.379	8.844
		SGARCH Variance	0.704	0.976	1.351	1.918	2.907	4.912	**8.212**
		Box Cox Variance	**0.439**	**0.639**	**0.942**	**1.437**	**2.364**	**4.682**	8.880
	SARIMAX	Constant Variance	0.538	0.778	1.129	1.695	2.994	5.977	10.696
		Daily Variance	0.571	0.806	1.149	1.706	2.629	4.702	7.844
		SGARCH Variance	0.618	0.857	1.203	1.717	2.529	**4.210**	**6.957**
		Box Cox Variance	**0.414**	**0.602**	**0.891**	**1.371**	**2.254**	4.333	8.407
Austin Texas	GRUs	Constant Variance	0.528	0.740	1.054	1.538	2.577	5.457	11.131
		Daily Variance	0.484	0.686	1.000	1.520	2.538	5.003	9.784
		SGARCH Variance	0.492	0.709	1.030	1.537	2.485	**4.683**	**8.380**
		Box Cox Variance	**0.305**	**0.502**	**0.822**	**1.356**	**2.329**	4.699	9.094
	XGBoost	Constant Variance	0.533	0.769	1.124	1.665	2.682	5.608	11.766
		Daily Variance	0.495	0.720	1.065	1.629	2.679	5.241	10.259
		SGARCH Variance	0.540	0.786	1.144	1.681	2.642	**4.958**	**8.886**
		Box Cox Variance	**0.349**	**0.573**	**0.932**	**1.523**	**2.627**	5.081	9.521
	SARIMAX	Constant Variance	0.546	0.771	1.105	1.623	2.616	5.305	10.768
		Daily Variance	0.543	0.765	1.097	1.623	2.563	4.780	9.038
		SGARCH Variance	0.496	0.728	1.077	1.619	2.548	4.583	**7.992**
		Box Cox Variance	**0.330**	**0.530**	**0.849**	**1.382**	**2.338**	**4.469**	8.375

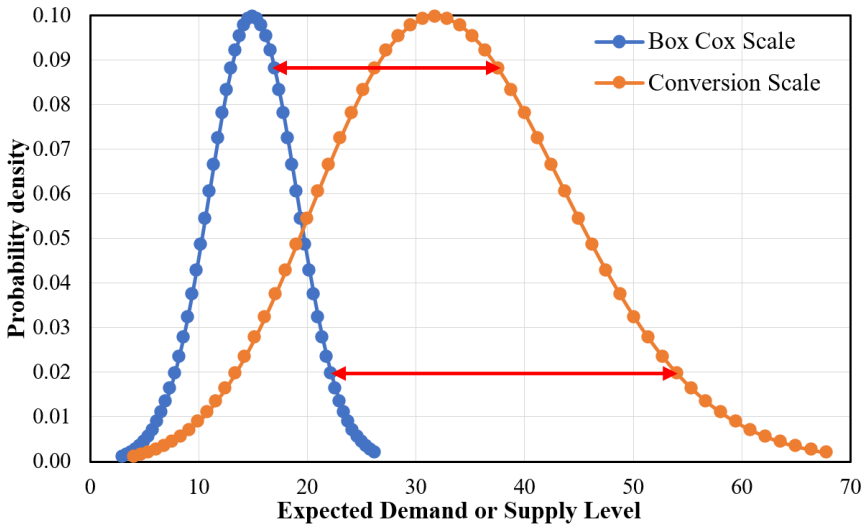

Figure 3.15 Impact of exponential conversion on supply level estimation with Box Cox transformed data

100 e-scooters for the same spatial coverage with a 2-hour rebalancing cycle. Similar to the Minneapolis case, the MO of SARIMAX was inferior to that of GRUs and XGBoost due to the seasonal demand pattern. In summation, across all datasets, incorporating conditional variance in supply level design resulted in a potential reduction of oversupply by approximately 26.22% at a 95% served demand level (equivalent to a shortage reduction of 5%).

3.6 Discussion and conclusion

This study presents a pragmatic framework aimed at devising an efficient supply planning strategy for addressing the heteroscedastic demand encountered in shared dockless e-scooter systems. The investigation involves the application of a range of prominent deep learning and machine learning models to predict hourly demand, followed by an analysis of the residuals' variance. The efficacy of this proposed methodology is evaluated across three distinct datasets pertaining to dockless shared e-scooter operations in Austin, TX, Minneapolis, MN, and Thammasat, TH.

The process entails meticulous hyperparameter tuning for both machine learning and deep learning models. The empirical findings indicate that demand prediction models, particularly those based on deep learning, exhibit remarkable performance levels. However, the residuals do not conform to the characteristics of white noise. This realization underscores the necessity of devising supply strategies tailored to the peculiarities of heteroscedastic demand. Such strategies involve the employment of variance-stabilizing transformations, exemplified by the Box Cox transformation, or the adoption of variance analysis, encompassing options such as daily seasonal variance or predicted variance through SGARCH.

While seasonal variance effectively curtails oversupply, its effectiveness diminishes when confronted with more protracted temporal residuals, notably those associated with yearly patterns. However, the introduction of a conditional variance model like SGARCH can surmount this limitation. An equally intriguing avenue is the utilization of the Box Cox transformation, a variance-stabilizing transformation. This approach not only enhances the performance of demand prediction models, particularly in terms of Mean Absolute Error (MAE), but also facilitates judicious supply-level planning, particularly at lower percentages

of served demand. It is imperative, though, to set an appropriate upper limit for supply-level planning when employing the Box Cox transformation to address higher percentages of served demand.

At a 95% served demand level, incorporating the consideration of heteroscedastic demand into supply level planning can yield a significant reduction in oversupply, quantified at 26.22%. In summation, this study underscores the insufficiency of demand prediction, even when leveraging deep learning, for short-term operational planning of shared e-scooter systems. These systems are characterized by high maintenance costs, short service life, erratic demand patterns, and stringent regulations. From a policy standpoint, operators stand to benefit from adopting our framework to mitigate demand uncertainties in their daily operations. This framework can be complemented by other strategies, such as customer incentives and a hybrid approach encompassing real-time and periodic rebalancing mechanisms.

References

[1] StataCorp, *Stata Time-Series Reference Manual*. Stata Press College Station, Texas, 2013.

[2] A. Rusyana, Nurhasanah, Marzuki and M. Flancia, "SARIMA model for forecasting foreign tourists at the Kualanamu International Airport," in *Proceedings of the 12th International Conference on Mathematics, Statistics, and Their Applications (ICMSA)*, 2016, pp. 153-158, doi: 10.1109/ICMSA.2016.7954329.

[3] B. Wang and I. Kim, "Short-term prediction for bike-sharing service using machine learning," *Transportation Research Procedia,* vol. 34, pp. 171-178, 2018, doi: https://doi.org/10.1016/j.trpro.2018.11.029.

[4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794, doi: https://doi.org/10.1145/2939672.2939785.

[5] A. Oyedele *et al.*, "Deep learning and Boosted trees for injuries prediction in power infrastructure projects," *Applied Soft Computing,* vol. 110, pp. 107587, 2021, doi: https://doi.org/10.1016/j.asoc.2021.107587.

[6] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics,* vol. 29, no. 5, pp. 1189-1232, 2001, doi: https://doi.org/10.1214/aos/1013203451.

[7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785-794.

[8] S. M. H. Moosavi *et al.*, "Understanding and Predicting the Usage of Shared Electric Scooter Services on University Campuses," *Applied Sciences,* vol. 12, no. 18, pp. 9392, 2022.

[9] S. Suman, S. Mishra and H. K. Tripathy, "A Support Vector Machine Approach for Effective Bicycle Sharing in Urban Zones," in *Cognitive Informatics and Soft Computing*, Singapore, 2021, pp. 73-83: Springer Singapore.

[10] C. Gao and Y. Chen, "Using Machine Learning Methods to Predict Demand for Bike Sharing," in *Information and Communication Technologies in Tourism 2022*, Cham, 2022, pp. 282-296: Springer International Publishing.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[12] Y. Yu, X. Si, C. Hu and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation,* vol. 31, no. 7, pp. 1235-1270, 2019, doi: https://doi.org/10.1162/neco_a_01199.

[13] C. Xu, J. Ji and P. Liu, "The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets," *Transportation Research Part C: Emerging Technologies,* vol. 95, pp. 47-60, 2018, doi: https://doi.org/10.1016/j.trc.2018.07.013.

[14] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[15] Y. Wu, H. Tan, L. Qin, B. Ran and Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," *Transportation Research Part C: Emerging Technologies,* vol. 90, pp. 166-180, 2018, doi: https://doi.org/10.1016/j.trc.2018.03.001.

[16] C. Wen *et al.*, "A novel spatiotemporal convolutional long short-term neural network for air pollution prediction," *Science of The Total Environment,* vol. 654, pp. 1091-1099, 2019, doi: https://doi.org/10.1016/j.scitotenv.2018.11.086.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems,* pp. 1-21, 2020, doi: https://doi.org/10.1109/TNNLS.2020.2978386.

[18] Z. Zhang, M. Li, X. Lin, Y. Wang and F. He, "Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies," *Transportation Research Part C: Emerging Technologies,* vol. 105, pp. 297-322, 2019, doi: https://doi.org/10.1016/j.trc.2019.05.039.

[19] X. Geng *et al.*, "Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 3656-3663.

[20] Y. Wu, "The Simulation Study of Shanghai and Shenzhen 300 Index By Garch Models," in *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, 2011, vol. 3, pp. 30-33, doi: https://doi.org/10.1109/ICIII.2011.293.

[21] A. Ti, Z. Du and W. Zhang, "Analysis on the Volatility of Sustainable Stock Index and Traditional Stock Index Based on GARCH Model," in *2019 International Conference on Economic Management and Model Engineering (ICEMME)*, 2019, pp. 47-50, doi: 10.1109/ICEMME49371.2019.00018.

[22] G. Zhang, S. Ali, X. Wang, G. Wang, Z. Pan and J. Zhang, "SPI-based drought simulation and prediction using ARMA-GARCH model," *Applied Mathematics and Computation,* vol. 355, pp. 96-107, 2019.

[23] C. Sigauke and D. Chikobvu, "Prediction of daily peak electricity demand in South Africa using volatility forecasting models," *Energy Economics,* vol. 33, no. 5, pp. 882-888, 2011.

[24] S. Kim, "Forecasting internet traffic by using seasonal GARCH models," *Journal of Communications and Networks,* vol. 13, no. 6, pp. 621-624, 2011, doi: https://doi.org/10.1109/JCN.2011.6157478.

[25] Y. Hu, J. Ni and L. Wen, "A hybrid deep learning approach by integrating LSTM-ANN networks with GARCH model for copper price volatility prediction," *Physica A: Statistical Mechanics and its Applications,* vol. 557, pp. 124907, 2020.

[26] W. Kristjanpoller and E. Hernández, "Volatility of main metals forecasted by a hybrid ANN-GARCH model with regressors," *Expert Systems with Applications,* vol. 84, pp. 290-300, 2017.

[27] S. He and K. G. Shin, "Dynamic Flow Distribution Prediction for Urban Dockless E-Scooter Sharing Reconfiguration," in *Proceedings of The Web Conference 2020*, 2020, pp. 133-143.

[28] S. He and K. G. Shin, "Distribution Prediction for Reconfiguring Urban Dockless E-Scooter Sharing Systems," *IEEE Transactions on Knowledge and Data Engineering,* vol. 34, no. 12, pp. 5722-5740, 2022, doi: https://doi.org/10.1109/TKDE.2021.3062074.

[29] S. W. Ham, J.-H. Cho, S. Park and D.-K. Kim, "Spatiotemporal Demand Prediction Model for E-Scooter Sharing Services with Latent Feature and Deep Learning," *Transportation Research Record,* vol. 2675, no. 11, pp. 34-43, 2021, doi: https://doi.org/10.1177/03611981211003896.

[30] S. Phithakkitnukooon, K. Patanukhom and M. G. Demissie, "Predicting Spatiotemporal Demand of Dockless E-Scooter Sharing Services with a Masked Fully Convolutional Network," *ISPRS International Journal of Geo-Information,* vol. 10, no. 11, p. 773, 2021.

[31] Y. Xu, X. Zhao, X. Zhang and M. Paliwal, "Real-Time Forecasting of Dockless Scooter-Sharing Demand: A Spatio-Temporal Multi-Graph Transformer Approach," 2021.

[32] S. Kim, S. Choo, G. Lee and S. Kim, "Predicting Demand for Shared E-Scooter Using Community Structure and Deep Learning Method," *Sustainability,* vol. 14, no. 5, pp. 2564, 2022.

[33] P. W. Khan, S.-J. Park, S.-J. Lee and Y.-C. Byun, "Electric Kickboard Demand Prediction in Spatiotemporal Dimension Using Clustering-Aided Bagging Regressor," *Journal of Advanced Transportation,* vol. 2022, pp. 8062932, 2022, doi: https://doi.org/10.1155/2022/8062932.

[34] N. Saum, S. Sugiura and M. Piantanakulchai, "Short-Term Demand and Volatility Prediction of Shared Micro-Mobility: a case study of e-scooter in Thammasat University," in *2020 Forum on Integrated and Sustainable Transportation Systems (FISTS)*, 2020, pp. 27-32: IEEE.

[35] I. K. Yeo and R. A. Johnson, "A new family of power transformations to improve normality or symmetry," *Biometrika,* vol. 87, no. 4, pp. 954-959, 2000.

[36] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001, doi: https://doi.org/10.1023/A:1010933404324.

[37] S. Kumar, L. Hussain, S. Banarjee and M. Reza, "Energy Load Forecasting using Deep Learning Approach-LSTM and GRU in Spark Cluster," in *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 2018, pp. 1-4, doi: https://doi.org/10.1109/EAIT.2018.8470406.

[38] J. R. Trapero, M. Cardós and N. Kourentzes, "Empirical safety stock estimation based on kernel and GARCH models," *Omega,* vol. 84, pp. 199-211, 2019, doi: https://doi.org/10.1016/j.omega.2018.05.004.

[39] P. L. King, "Crack the code: Understanding safety stock and mastering its equations," *APICS Magazine,* vol. 21, no. 2011, pp. 33-36, 2011.

[40] E. D. O'Mahony, "Smarter tools for (Citi) bike sharing," Ph.D. dissertation, Cornell University., 2015.

[41] Y.-H. Seo, "A Dynamic Rebalancing Strategy in Public Bicycle Sharing Systems Based on Real-Time Dynamic Programming and Reinforcement Learning," Ph.D. dissertation, Seoul National University, South Korea, 2020.

[42] N. Saum and M. Piantanakulchai, "A Review on an Emerging New Mode of Transport: The Shared Dockless Electric Scooter," in *Proceedings of the Eastern Asia Society for Transportation Studies*, Srilanka, vol. 12, 2019.

[43] T. O'Malley *et al.*, "Keras documentation: Keras Tuner," Available: https://github.com/keras-team/keras-tuner

[44] X. Liu, A. Gherbi, W. Li and M. Cheriet, "Multi features and multi-time steps LSTM based methodology for bike sharing availability prediction," *Procedia Computer Science,* vol. 155, pp. 394-401, 2019.

CHAPTER 4
4. HYPERPARAMETER OPTIMIZATION BY ITERATIVE DECISION TREE (IDT)

4.1 Introduction

There has been considerable focus from both the academic and corporate sectors on the implementation of machine learning (ML) and deep learning (DL) techniques, influenced by various factors [1], [2], [3], and [4]. Initially, the utilization of conventional statistical models becomes problematic when dealing with the ever-expanding volumes of high-dimensional data. Additionally, the advanced algorithms derived from ML and DL have demonstrated impressive predictive capabilities, significantly influencing the demanding business landscape. Lastly, the recent advent of high-performance computing resources enables the rapid training of intricate models encompassing millions of trainable parameters, especially when executed in parallel.

However, these ML and DL models need a lot of computational work, particularly when it comes to fine-tuning the hyperparameters (outer variables) and optimizing the trainable or ordinary parameters (inner variables). Trainable or ordinary parameters denote the weight matrix or bias vector associated with a specific model, and they are automatically optimized or learned during the model training phase. On the other hand, hyperparameters relate to parameter sets that users frequently set. The hyperparameters for straightforward artificial neural networks (ANNs) can include batch size, learning rate, activation function, number of layers, and neurons per layer, among others. The training dataset is used to optimize ordinary parameters, which are primarily continuous variables (inner optimization). Conversely, hyperparameters can take on continuous, integer, binary, or categorical values, resulting in objective functions that are typically non-differentiable. And these objective functions are optimized using the validation dataset (known as outer optimization). Additionally, certain studies have aimed to optimize both the ordinary parameters and hyperparameters expeditiously, either by treating them as a unified objective function [5], [6], [7] or by considering multiple objective functions [8], [9], [10]. Hyperparameter Optimization (HPO) can be addressed through the utilization of meta-learning, employing either a single learning algorithm (homogeneous meta-learning) or multiple learning algorithms (heterogeneous meta-learning). This approach enables the simultaneous optimization of both the configuration of the learners and the learning algorithms themselves. A thorough examination of heterogeneous meta-learning can be found in [1], while popular frameworks such as Auto-WEKA, Hyperopt-sklearn, Auto-sklearn, Auto-Net, etc., have been developed. In conclusion, there has been a lot of prior research on automatic machine learning (AutoML) to reduce human intervention and bias, making it more approachable for laypeople.

This study exclusively centers on Hyperparameter Optimization (HPO), assuming the objective function to be black-box, computationally intensive, and non-differentiable. As discussed in [1] and [4], the origins of the HPO problem can be traced back to the early 1990s when researchers began comparing the predictive performance of a single model with varying configurations. The primary objective of HPO is to identify the optimal combination of hyperparameters that yield the highest or lowest prediction performance on the validation dataset, depending on whether it involves maximization or minimization. The maximization variant of the HPO problem is typically formulated as follows:

$$x^* = \underset{x}{\operatorname{argmax}}\{f(x): x \in X\} \tag{4.1}$$

The objective function $f(x)$ in HPO represents the quantity to be maximized, which could be metrics such as accuracy or the negative Mean Squared Error (MSE). However, in HPO, this function lacks an explicit expression as its value corresponds to the evaluated performance of the model on the evaluation dataset or a specific objective function, such as a weighted average MSE computed from the training and validation datasets. The optimal set of hyperparameters x^* represents the combination that yields the highest value of the objective function, while x represents a potential hyperparameter combination drawn from the search space X.

A significant body of literature on Hyperparameter Optimization (HPO) exists, which can be classified into distinct categories: manual search, model-free approach, model-based approach, population-based approach, and optimization-based approach.

Manual Search (MS), also referred to as “Trial and Error,” “Babysitting,” or “Grad Student Descent,” is an HPO technique that relies on human expertise to determine the adjustments to be made to hyperparameters in each iteration. It involves a series of trial-and-error experiments, guided by the practitioner's knowledge and historical experience until either time constraints are reached or a stopping criterion is met [11]. As a result, MS often relies on "rule of thumb" or default settings, relying on prior knowledge and personal preferences. This approach is time-consuming and computationally expensive [12], often leading to locally optimal results [13], [14], and limited reproducibility [1]. However, MS provides valuable insights into the impact of individual hyperparameters and does not require significant computational resources [15]. Therefore, MS is well-suited for addressing HPO challenges associated with simple learning algorithms with few hyperparameters.

Model-Free Approach encompasses those automatic HPO techniques that rely on trial-and-error or random-walk exploration of the hyperparameter space. This approach primarily includes two main techniques: Grid Search (GS) and Random Search (RS). GS is a fundamental method in HPO that involves evaluating all possible combinations of hyperparameters. However, GS is highly sensitive to the number of hyperparameters, as the number of potential configurations increases exponentially. To mitigate this challenge, GS is often conducted on a coarse grid, ex., step sizes of 100 or powers of 2. Alternatively, certain hyperparameters may be varied while keeping others fixed to estimate the model's performance [16]. RS involves evaluating learning algorithm configurations randomly sampled within each hyperparameter's lower and upper bounds until the allocated budget is depleted. Given that each hyperparameter affects the objective function differently, RS is theoretically more effective than GS when working with a tight budget or in high-dimensional spaces [17]. Floria and Andonie [18] introduced an extension to Random Search (RS) known as Weighted Random Search (WRS), where each hyperparameter is assigned a distinct probability of change. Additionally, the Multi-fidelity approach has been applied to RS for HPO. This approach allocates low-fidelity (early stopping) and high-fidelity evaluations to underperforming and potentially promising configurations. Bandit-based algorithms, such as successive halving [19], [20], [3], and Hyperband [21], divide the total budget (e.g., 100 epochs) into several rungs (e.g., 10 epochs per rung). All configurations are compared within each rung, and half of the configurations with unfavorable performance are eliminated.

One advantage of these model-free approaches is that the configurations can be trained independently, making it straightforward to implement model parallelism [1], [3], [4].

Model-Based Approach, also known as sequential model-based optimization (SMBO), utilizes a surrogate regression model $\hat{f}(X)$ to understand the impact of hyperparameters X on the given black-box function $f(x)$. The surrogate function is trained using a limited number of initial random configurations, and subsequently, an acquisition function is employed to propose a new candidate configuration. Subsequently, the performance of this new configuration is evaluated, and the surrogate function is updated accordingly. This sequential process continues iteratively until the stopping criteria are met. Within the Model-Based Approach, numerous algorithms have been proposed, each utilizing distinct surrogate models and strategies for selecting new hyperparameters. These algorithms include Bayesian Optimization (BO) [22], [23], [24], Sequential Model-based Algorithm Configuration (SMAC) employing random forest [25], Tree-structured Parzen Estimator (TPE) [14], Nelder-Mead [26], and several others. In order to reduce the training time of sequential HPO algorithms, several multi-fidelity approaches have been combined with Bayesian Optimization (BO). Examples of such combinations include Freeze-thaw BO [27], BO with Bayesian Optimal Stopping (BO-BOS) [28], and BO with HyperBand (BOBH) [29]. Most algorithms within this approach suggest only a single new candidate per iteration, which makes parallelization challenging. However, there have been efforts to enable parallel computing in Sequential Model-Based Optimization (SMBO) approaches, including BOBH [29], BO with Pure Exploration (GP-UCB-PE) [30], BO with multi-points Expected Improvement (BO-q-EI) [31], and Batch BO with parallel knowledge gradient [32].

Population-Based or Nature-Inspired Approach refers to optimization techniques that draw inspiration from biological evolution, involving concepts such as reproduction, mutation, selection, and the interaction of agents [33]. Within this approach, numerous algorithms have been proposed to solve optimization problems across various fields, and recently, some popular algorithms have been applied to Hyperparameter Optimization (HPO). For instance, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) has been utilized for the parallel optimization of Convolutional Neural Networks (CNNs) architectures [34]. The Genetic Algorithm (GA), known as one of the most popular evolutionary algorithms, has been employed for parallel search in various configurations, including CNNs [13], Deep Belief Networks (DBN) [35], Lenet-5 CNNs, and convolutional autoencoder [36]. Additionally, other population-based algorithms have been applied in Hyperparameter Optimization (HPO), such as Particle Swarm Optimization (PSO) [11], Univariate Dynamic Encoding Algorithm for Searches (uDEAS) [37], Cuckoo Search [38], Differential Evolution (DE) [39], Simulated Annealing (SA) [40], and Harmony Search [41]. These nature-inspired algorithms have also been extended to enhance their searching performance, for example, the combination of GA and Tabu Search (Tabu_Genetic Algorithm) [42], GA with Local Search (Memetic Algorithm) [43], and the Statistically-driven Coral Reef Optimization algorithm with Hybridisation (HSCRO) [44].

In the optimization-based or gradient-based approach, several studies have made efforts to estimate the gradients of validation performance with respect to all hyperparameters. For example, Maclaurin *et al.* [5] employed backpropagation of stochastic gradient descent with momentum to compute the exact gradients of continuous hyperparameters in a neural network architecture. Brock *et al.* [6] introduced One-Shot Model Architecture Search through HyperNetworks (SMASH) to approximate architecture weights using HyperNet. Likewise,

Lorraine and Duvenaud [7] extended SMASH, known as Hyper-Training, by applying the chain rule to jointly optimize deep learning's weights and hyperparameters through stochastic optimization. Furthermore, the bilevel optimization framework has been utilized to simultaneously optimize the validation loss (outer objective) and training loss (inner objective) for tasks such as data augmentation strategy [9] and the integration of gradient-based HPO and meta-learning [10].

In summary, extensive research has been conducted on HPO algorithms, with a focus on assisting automated machine learning. Two prominent approaches in this field are the sequential and population-based approaches. The sequential-based approach is well-suited for scenarios involving expensive objective functions. However, it has a limitation in that it suggests only one new candidate in each iteration, making it challenging to implement in parallel computing. On the other hand, the population-based approach is more suitable for situations with inexpensive objective functions, as it can suggest dozens to hundreds of new candidates per generation, enabling easy parallel computation. A significant disparity exists between the sequential-based and population-based approaches in terms of the number of candidates suggested per iteration. Additionally, reproducibility, particularly for deep learning models, is a concern, as many HPO algorithms tend to retrain repetitive candidates, leading to variations in results. This study proposes a novel HPO algorithm called Iterative Decision Tree (IDT) to address these limitations. IDT is a sequential-based algorithm that utilizes Decision Tree (DT) regression as the surrogate function. In other words, a DT regressor is employed to partition the search space based on the evaluated points, ensuring that all possible candidates within each region yield the same expected result. Unlike traditional approaches, IDT does not rely on an acquisition function. Instead, it suggests new candidates by selecting extreme or random points from the best-performed region (exploitation). To enhance exploration capabilities, the algorithm suggests new candidates from a few up to a dozen of the best-performed regions, thereby increasing the diversity of the search. By offering flexibility in selecting the number of new candidates per iteration and emphasizing reproducibility, IDT aims to overcome the limitations of existing HPO algorithms.

The following is a summary of the study's contributions:

- Introduce a novel algorithm for hyperparameter optimization (HPO) called Iterative Decision Tree (IDT), which utilizes Decision Tree regression as the surrogate function. This approach addresses certain limitations observed in existing algorithms. The proposed IDT algorithm consists of two variants: IDT-E, which selects extreme points as new candidates, and IDT-R, which employs random selections.
- Conduct a comparative analysis of the proposed algorithms (IDT-E and IDT-R) against several widely-used state-of-the-art algorithms, namely Grid Search, Random Search, Bayesian Optimization, Random Forest, Tree-structured Parzen Estimator, and Genetic Algorithm.
- Assess the efficacy of the IDT algorithm by evaluating its performance on diverse optimization problems, including benchmark nonconvex functions and hyperparameters of Support Vector Machine (SVM), Random Forest (RF), Auto-Encoder (AE), and Convolutional Neural Networks (CNNs). The evaluation of hyperparameter optimization involves employing benchmark datasets such as digits classification, car evaluation classification, MNIST, and CIFAR-10 for the respective models.

4.2 Literature review

Even machine learning (ML) and deep learning (DL) models could outperform the statistical approaches, but they need to be traded off between accuracy against training cost and model complexity. Therefore, hyperparameter optimization (HPO) or hyperparameter tuning is necessary to control DL models' reproducibility, accuracy, and overfitting. As a result from Chapter 3, some configurations or hyperparameters of machine learning and deep learning models could perform worse than statistical models. Under computational time constraints, the HPO approaches presented above were employed to optimize only some important hyperparameters, while other hyperparameters might be fixed or use default values.

Most RNNs were trained with default parameters or standard architecture to reduce the training time during hyperparameter tuning. LSTM-based Sequence to Sequence (S2S) architecture was used to predict the household loading based on 60 sequence length and Adam optimizer, and tuning two parameters, number of hidden layers and nodes per layer [45]. The combination of feedforward NNs and LSTM NNs was employed to predict the $PM_{2.5}$ Concentration in Jingjinji area, China [46]. The authors constructed the model by combining 2 FNN layers with one or two LSTM NNs layers with a sequence length of 48, 3 neighborhoods, 100 epochs, 256 batch size, 0.1 dropout rate, and RMSprop optimizer. The architecture, 1-Shared Feedforward NNs + 2-Bi-directional RNNs + Soft Attention Mechanism, was proposed to predict bike sharing (Citi Bike) in New York [47]. The authors trained this architecture to predict the real value using Relu activation and 50 epochs. RNNs were used for intrusion detection with only two tuned parameters, learning rate and number of nodes per layer [48]. To deal with the volatility of residential's loading, LSTM NNs were used to predict the loading of household appliances by tuning only one parameter, the sequence length [49]. The combination of LSTM NNs and Gaussian Mixture Model (GMM) was proposed to predict the health status of aircraft turbofan engines [50]. This model's configuration comprises one LSTM layer with 64 nodes, 10% dropout rate, batch size of 10, Adam optimizer with learning rate (0.0001) and gradient clipping (1.0), and 5-component GMM.

C. Xu, Ji, and Liu (2018) used LSTM NNs to predict dockless shared bike demand in Nanjing (Jiangsu), China. Grid search on four parameters (number of epochs, batch size, number of nodes, and dropout rate) of LSTM NNs was examined [51]. Wang and Kim (2018) employed Random Forest, LSTM NNs, and GRU to predict station-level bike availability in Suzhou (China), and these models yielded almost the same performance. This study set the parameters for GRU NNs and LSTM NNs as: 2 hidden layers, 100 nodes in each hidden layer, and 25 epochs [52]. Kumar, Hussain, Banarjee, and Reza (2018) employed RNNs, LSTM NNs, and GRU to predict electricity load by tuning only two parameters, number of hidden layers and number of nodes per layer. Their results showed that the performance of these three models could be ranked from worst to best as RNNs, LSTM NNs, and GRU, respectively [53]. Huang and Kuo (2018) combine 1-Dimensional Convolutional Neural Networks and LSTM NNs to predict the hourly $PM_{2.5}$ Concentration in Beijing [54]. This architecture comprises three layers of CNN1D with SELU activation connecting to one-layer LSTM NNs and a dense output layer with sigmoid activation. The data were transformed with normalization before inputting into the architecture, while the best model was selected using early stoppage criteria. Random Search algorithm was proposed to tune the configuration of GRU in detecting the electricity

cyber-attack [55]. This algorithm was used to find the optimal values of 4 parameters such as number of layers, number of nodes per layer, activation function, and optimizer. On the other hand, Manual Search was performed to find optimal hyperparameters (ex., hidden layers, number of nodes, dropout rate, learning rate, number of epochs, batch size and lookback length) of Bi-LSTM NNs in fog-cloud-based intrusion detection [56].

This architecture was also the baseline for multistep shared bike availability [57]. Similarly, LSTM NNs were used to predict City Bike demand in New York by fixing the number of hidden layers to 2 with 1000 nodes each [58]. To improve the performance of LSTM NNs, the authors added demand data from the nearby regions, called Neighborhood-Augmented LSTM NNs, to predict taxi-passenger demand in Porto, Portugal [59]. The authors performed grid search for many parameters, such as number of neighborhoods, window size, number of hidden layers, number of nodes per layer, batch size, epochs, and dropout rate. Uddin, Bapery, and Arif (2019) used GRU model to predict depression statements on Bangla social media. This study used hyperparameter tuning for four parameters: number of GRU layers, layer size, batch size, and number of epochs [60].

Grid Search was employed to tune the parameters (Transformation, number of hidden layers, number of nodes per layer, dropout rate, epoch, learning rate, optimizer) of LST NNs for Solar Irradiance Forecasting [61]. LSTM NNs were used to predict the time series datasets on Kaggle by randomly tuning two parameters, including the number of hidden layers and nodes per layer [62]. The combination of CNN1D with ANNs was used to predict the daily rainfall in Maharashtra [63]. To ascertain the best configuration of the model, the authors try some random searches for several parameters, including activation function, epoch, number of hidden layers, number of nodes per layer, batch size, learning rate, dropout rate, number of filters, kernel size, pooling size, and loss function, while fixing other three parameters such as data normalization, five lookback lags, and Adam optimizer. LSTM architecture with 4 dropout layers and 4 hidden LSTM layers was employed to predict daily open prices of GOOGL and NKE, while 4 cases of epochs (12, 25, 50, 100) were examined [64]. LSTM NNs and GRUs were employed to forecast wind power, while several hyperparameters were optimized by manual search (ex., number of hidden layers, and number of nodes per layer) and grid search (ex., batch size, epochs, optimizer, activation function, and kernel initializer) [65]. Similarly, grid search was also employed to optimize the hyperparameters (ex., number of nodes per layer, batch size, dropout rate, activation function, lookback length, and epochs) of LSTM NNs and GRUs in pore pressure prediction [66]. Several recent studies employed grid search to optimize, RNN models for waste disposal rate prediction [67, 68], RNN models for high-speed train vibration prediction [69], graph convolutional recurrent neural networks (GCRNNs) for water demand forecasting [70], and RNN-based hybrid and ensembles for stock market prediction [71].

Yahyaoui (2019) employed three optimizations, Bayesian optimization with Gaussian Process (BO-GP), Tree-structured Parzen Estimator (TPE), and Covariance Matrix Adaptation Evolutionary Strategy (CMAES), to tune the configuration of LSTM NNs in financial time series forecasting. These three algorithms were employed to find the best parameter of LSTM NNs' parameters: Sequence Length, Number of Hidden Layers, Number of Nodes per layer, Dropout Rate, Learning Rate, and Activation Function of Hidden Layers. His result shows that TPE could achieve the overall highest performance for in-sample and out-of-sample metrics,

speed, and low trial-to-trial variability [72]. Wu et al. (2019) employed Bayesian Optimization (BO) algorithm to find the optimal parameters for RF, CNNs, and RNNs. This BO algorithm cannot tune sequential decision parameters such as the number of layers, number of nodes per layer, and filter size. In this case, they employed BO algorithm to optimize only the learning rate and batch size for RNNs [22]. TPE was used to tune LSTM NNs in household appliances' load consumption by tuning four parameters, including sequence length, number of hidden layers, number of nodes per layer, and optimizer optimization [73]. In addition, TPE optimization was applied for hyperparameter tuning of Multi-Attention Recurrent Neural Networks and compared with the random and manual search [74]. In this research, the authors tuned several parameters such as activation function, attention length, number of nodes per layer, dropout rate, and learning rate, while other parameters were fixed as sequence length (24), batch size (128), epoch (10), Adam optimizer, and Normalization.

Swarm optimizations were also popular for hyperparameter tuning for deep learning. For instance, Rashid, Aziz, and Hasan (2019) employed Particle Swarm Optimization (PSO) algorithm for tuning the parameters of RNNs in machine failure prediction. However, PSO is feasible for only the linear parameters, so this algorithm could cope with only the learning rate [75]. Harmony Search Algorithm was employed to tune the CNNs models to predict the MNIST and Cifar-10 datasets [41]. Four critical parameters of CNNs were tuned by this algorithm, including kernel size, pooling size, number of channels, and strides. [38]. Five public datasets were used to evaluate the efficiency of the Cuckoo Search Algorithm in tuning the parameters of LSTM NNs, including the number of hidden layers, number of nodes per layer, and optimizers [38]. On the other hand, Bouktif, Fiaz, Ouni, and Serhani (2020) combined PSO and Genetic Algorithm (GA) to find the optimal parameters for LSTM NNs for electric load forecasting. Their methodology was used to find the optimal parameter one by one, including the number of sequences, sequence length, starting point, number of nodes, batch size, activation function, and optimizer [76].

Differential evolution was used to tune two variables (number of nodes per layer and batch size) of LSTM NNs for motion classification [39]. The sequence length is fixed to be 25 and 8 for EEG and BVP, respectively, while the number of epochs and learning rate are 10 and 0.0025. The authors compare the performance with other algorithms at three different levels of interactions, such as 50, 100, and 300. Yoo (2019) proposed an automatic parameter optimization for deep neural networks using Univariate Dynamic Encoding Algorithms for Searches (uDEAS). As a result, this algorithm could converge to the optimal solution using 402 and 802 searches compared to 2^{18} and 2^{24} possibilities of grid search for three parameters, learning rate, number of hidden layers, and batch size [37]. Lastly, two deep learning models (RNNs and CNNs) and three datasets (human intension recognition EEG, activity recognition by wearable sensors IMU, and activity recognition by pervasive sensors RFID) were employed to evaluate the performance of Orthogonal Array Tuning Method (OATM) [77]. For RNNs, the authors tuned several parameters, such as the learning rate, the regularization coefficient, the number of hidden layers, and the number of nodes in each hidden layer. On the other hand, some parameters of CNNs were tuned, including the learning rate, the filter size, the number of convolutional and pooling layers, and the number of nodes in the second fully connected layer.

In summary, this study also evaluates the effectiveness of the proposed HPO algorithm, Iterative Decision Tree (IDT), in predicting the spatiotemporal demand of shared e-scooters.

Due to time limitations, we selected only two models, Random Forest (RF) and Gated Recurrent Units (GRUs), as the case study. The evaluations were performed on three datasets, as mentioned in the previous chapter, namely Thammasat (TH), Minneapolis (MN), and Austin (TC). Based on the literature reviewed above, it is commonly observed that several hyperparameters of recurrent neural network architectures need to be optimized to achieve desirable prediction results. These hyperparameters include the lookback length, number of layers, number of nodes per layer, activation function, dropout rate, learning rate, and batch size. Therefore, in this study, we also optimized these hyperparameters for GRUs to predict the spatiotemporal demand of shared e-scooters. In this section, RF was employed to predict the spatiotemporal demand of shared e-scooters while optimizing several hyperparameters, including the lookback length, sampling rate, number of trees in the forest, and maximum depth of the tree.

Table 4.1 Summary of hyperparameter tuning methods of deep learning models

#	Authors	Year	Deep Learning Models	Hyperparameter Tuning Method	Hyperparameters	
					Fixed	Varied
1	Marino et al.	2016	LSTM NNs Sequence2-Sequence	Grid Search	60 Lookback Length, Adam	No. HLayers, No. Nodes/layer
2	Fan et al.	2017	FNN + LSTM NNs	No	48 Lookback Length, 2 FNNs+1/2 LSTM NNs, 3 Neighborhoods, 100 Epoch 256 Batch Size, 10% Dropout Rate, RMSProp	-
3	P. Chen et al.	2017	SFNNs+ Bi-RNNs + SAM	No	Real-Value, 2 HLayers Relu Activation, 12 Nodes/layer, 50 Epochs	-
4	Yin et al.	2017	RNNs	Grid Search	Normalization, 100 Epochs	Learning Rate & No. Nodes/layer
5	Kong et al.	2017	LSTM NNs	Grid Search	Linear Activation, Adam, 2 HLayers + 1Dense Layer, 512 Nodes/layer	Lookback Length
6	Kong et al.	2017	LSTM NNs	Tree-structured Parzen Estimator	-	Lookback Length, No. HLayers, No. Nodes/layer, Optimizer
7	C. Xu et al.	2018	LSTM NNs	Grid Search	5 Neighborhoods, Adam, Standardization	Batch Size, Dropout Rate, No. of Nodes & Epoch
8	B. Wang & Kim	2018	LSTM NNs & GRU	Grid Search	2 HLayers, 25 Epochs, 100 Nodes/layer	Lookback Length
9	Kumar et al.	2018	RNNs, LSTM NNs, GRU	Grid Search	24 Lookback Length	No. of HLayers & No. of Nodes/layer
10	Huang & Kuo	2018	CNN1D + LSTM NNs	No	Normalization, SELU/ Sigmoid, 24 Lookback Length, 3 CNN1D layers + 1 LSTM NNs layer, Early Stoppage	-
11	Nabil et al.	2018	GRU	Random Search	10 Epochs, 350 Batch Size, 20% Dropout Rate	No. Layers, No. Nodes/layer, Optimizer & Activation
12	Lee et al.	2018	CNN	Harmony Search Algorithm	2 Conv layers + 2 ANN layer	Kernel Size, Pooling Size, Stride, Padding
13	Nakisa et al.	2018	LSTM NNs	Differential Evolution	25/8 Lookback Length 10 Epochs 0.0025 Learning Rate	No. Nodes/layer, Batch Size
14	Liu et al.	2019	LSTM NNs	No	20 Lookback Length, 2 HLayers, 100 Nodes/layer	-
15	Y. Pan et al.	2019	LSTM NNs	No	24 Lookback Length, 2 HLayers, 1000 Nodes/layer	-

#	Authors	Year	Deep Learning Models	Hyperparameter Tuning Method	Hyperparameters	
					Fixed	Varied
16	Le Quy et al.	2019	LSTM NNs	Grid Search	Normalization, Adam, Tanh Activation	Lookback Length, Batch Size, No. of Neighborhoods, Epochs, HLayers, Nodes per Layer & Dropout Rate
17	Uddin et al.	2019	GRU	Grid Search	0.0001 Learning Rate	No. of HLayers, Epochs, Nodes/layer, Batch Size
18	Husein & Chung	2019	LSTM NNs	Grid Search	-	Transformation, No. HLayers, No. Nodes/layer, Dropout Rate, Epoch, Learning Rate, Optimizer
19	Peter & Matskevichus	2019	LSTM NNs	Random Search	-	No. Layers, No. Nodes/layer
20	Yahyaoui	2019	LSTM NNs	BO-GP, TPE Covariance Matrix Adaptation Evolutionary Strategy	Adam, Early Stoppage (1000), 32 Batch Size, Linear Activation for Output Layer	Lookback Length, No. HLayers, No. Nodes/layer, Dropout Rate, Learning Rate, Activation Function of HLayers
21	J. Wu et al.	2019	Random Forest, CNNs, RNNs, Cascade Forest	Bayesian Optimization	1 HLayers, 28 Lookback Length, 128 Nodes/layer	Batch Size, Learning Rate
22	Mashlakov et al.	2019	Multi-Attention RNNs	Tree-structured Parzen Estimator	24 Lookback Length, 128 Batch Size, 10 Epochs, Adam, Normalization	Activation Function, No. layer of Attention, No. Nodes/layer, Dropout Rate, Learning Rate
23	Rashid et al.	2019	LSTM NNs	Particle Swarm Optimization	100/500/1000 Epochs 2/4/6 Nodes/layer	Learning Rate
24	Srivastava et al.	2019	LSTM NNs	Cuckoo Search Algorithm	400 Epochs	No. HLayers, No. of Node/layer Optimizer
25	Yoo	2019	Autoencoder CNN	Univariate Dynamic Encoding Algorithm	1 HLayer, 20 Epochs 2 CNN Layers, 20 Epochs, 3 Filter Size	Learning Rate, No. Nodes, Batch Size, Conv L1 & L2
26	X. Zhang et al.	2019	CNNs RNNs	Orthogonal Array Tuning Method	Adam	Learning Rate, Filter Size, No. Conv. &Pool Layers, No. Nodes ANN layers Learning Rate, No. Hlayers+2 LSTM, No. Nodes/layer, Regularization Coef.
27	M. I. Khan & Maity	2020	CNN1D + ANNs	Random Search	5 Lookback Length, Adam, Normalization	Activation Function, Epoch, No. HLayers, No. Nodes/layer, Batch Size, Learning Rate, Dropout Rate, No. of Filter, Kernel Size, Pooling Size, Loss Function
28	Boukif et al.	2020	LSTM NNs + ANNs	Genetic Algorithm + Particle Swarm Algorithm	1 LSTM + 3 ANN layers, Early Stoppage, Dropout Rate, No. Nodes/ANNs layer	No. of Sequence, Lookback Length, Starting Points, Activation Function, No. LSTM, Batch Size, Optimizer
29	Moghar & Hamiche	2020	LSTM NNs	Grid Search	5 LSTM Layers + 4 Dropout Layers, 50 Lookback Length, 96 Nodes/layer	Epoch
30	Q. Wu et al.	2020	LSTM NNs + Gaussian Mixture Model	No	64 Nodes, 10 Batch Size, Adam Optimizer, 0.0001 Learning Rate, 1.0 Gradient Clipping, 10% Dropout Rate, Data Augmentation	-
31	Ham et al.	2021	ERD or CNNs +RNNs	Grid Search	5 Lookback Length, 5 RNNs layers	Learning rate, Activation Function GRU or LSTM cell, Input features
32	Kisvari et al.	2021	LSTM NNs & GRUs	Grid Search Manual Search	-	Batch Size, Epochs, Optimizer, Activation Function, Kernel Initializer, No. Nodes/layer, No. HLayers

#	Authors	Year	Deep Learning Models	Hyperparameter Tuning Method	Hyperparameters	
					Fixed	Varied
33	Wei et al.	2021	LSTM NNs & GRUs	Grid Search	-	No. Nodes/layer, Batch Size, Dropout Rate, Epochs, Activation Function, Lookback Length
34	Y. Huang et al.	2021	S*pAtt*RNN	No	0.01 Learning Rate, 30 Epochs, 128 LSTM nodes, 10 Spatio-Attention Outputs	-
35	Vu et al.	2021	LSTM NNs & RNNs	Grid Search	1 HLayer with 128 Nodes 2 HLayer with 600 Nodes	Lookback Length
36	Vu et al.	2022	LSTM NNs & RNNs	No	1 Input Layer + 1 HLayer, 128 Nodes/layer	-
37	Siłka et al.	2022	LSTM NNs	Grid Search	8 LSTM Layers + 2 ANN Layers, ReLU output Activation Function	Optimizer, Lookback Length, Error Margin
38	Zanfei et al.	2022	GCRNNs	Grid Search	3 GCN Layers + 2 LSTM Layers 100 Epochs, 24 Batch Size	No. Nodes and Edges
39	Syed et al.	2023	RNNs & Bi-LSTM NNs	Manual Search	10% Dropout Rate, Adam	No. HLayers, No. Nodes/layer, Learning Rate, Epochs, Batch Size, Lookback Length
40	Song & Choi	2023	CNNs + LSTM NNs GRUs-CNNs Ensembles	Grid Search	10 Early Stopping Patience, 1D-Conv with 32 Filters, 3 Size & 1 Stride	Lookback Length, Dropout Rate, Epochs, Learning Rate, No. HLayers, No. Nodes/layer, Batch Size

4.3 Methodology

Decision Tree (DT) is a powerful non-parametric supervised learning technique that builds the classification or regression models as a tree structure. As shown in **Figure 4.1**, DT composes of three main elements: root node (parent node), decision node (child node or interior node), and terminal node (leaf node). The root node is the initial node covering the entire sample and may get split into decision nodes or leaf nodes. The decision node represents the decision rule using data features to get two or more branches, while the leaf node represents the outcome. There are several popular algorithms proposed for a decision tree from a dataset, such as Iterative Dichotomiser 3 (ID3), ID3's extensions (C4.5 and C5.0), and Classification and Regression Tree (CART).

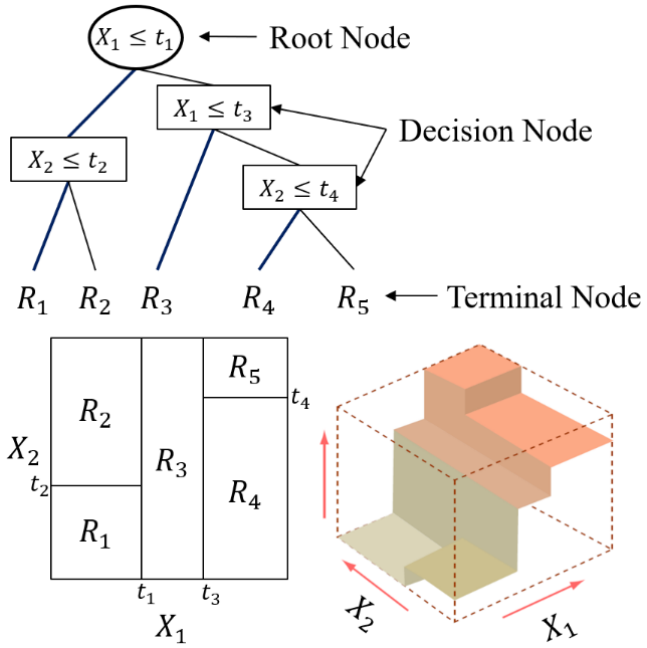

Figure 4.1 Example of DT regressor with CART algorithm: *top* tree corresponding to the partition of the *bottom left* panel and the perspective plot of the prediction surface is on the *bottom right* panel [78]

The decision tree regressor (or regression tree) was built based on the CART algorithm using a Python module, *DecisionTreeRegressor*, of scikit-learn [79]. The detailed formulation of this algorithm can be found in their documentation or in [78]. The formulation of DT with MSE as the loss function for a given training vectors $x_i \in R^n, i = 1, \ldots,$ Iand the label vector $y \in R^l$spliting on node m with feature j and threshold t_m (i.e., splitting candidate $\theta = (j, t_m)$) is as follows:

$$Q_m^{left}(\theta) = \{(x,y)|x_j \le t_m\} \tag{4.2}$$

$$Q_m^{right}(\theta) = \{(x,y)|x_j > t_m\} \tag{4.3}$$

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H\left(Q_m^{left}(\theta)\right) + \frac{n_m^{right}}{n_m} H\left(Q_m^{right}(\theta)\right) \tag{4.4}$$

$$H(Q_m) = \frac{1}{n_m} \sum_{y \in Q_m} (y - \bar{y}_m)^2 \tag{4.5}$$

Where $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ are the subsets of data partitioned on node m to the left and right nodes, respectively. The quality of a candidate split $G(Q_m, \theta)$ is minimized and grown until the maximum allowable depth $\theta^* = \text{argmin}_\theta G(Q_m, \theta)$. The split quality is simply the weighted average of loss function $H(Q_m)$ as MSE of that node (i.e., the prediction of the node $\bar{y}_m$ is the average value). In case of a classification problem with targe class values of (0, 1,…, K-1), the criterion or loss function could be the Gini impurity $H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$ or Entropy $H(Q_m) = -\sum_k p_{mk} \log(p_{mk})$, where $p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$.

Our proposed method, the Iterative Decision Tree (IDT), is a sequential model-based optimization (SMBO) technique that uses decision tree regression as the surrogate function. DT regressor is suitable for learning the correlation between hyperparameters and the black-box

objective function, often referred to the loss function in machine learning. It provides short training time, the ability to handle categorical and conditional variables, small datasets with multiple features, and nonlinear relationships. From the parameter setting of CART, as mentioned above, DT is employed to partition the overall search space based on the evaluated points. In this case, the DT regressor works as the surrogate function so that the new candidates, as extreme or random points, can be drawn from the promising regions (terminal nodes). In other words, the DT regressor is used to predict the expected objective value of all the combinations of hyperparameters. After that, the new candidates were selected from the combinations given the highest expected objective (i.e., the highest promising region). For other promising regions following the sorted evaluated points, the new candidates were selected from each region independently.

Figure 4.2 shows the Iterative Decision Tree algorithms with the new candidates as extreme points (IDT-E) or random points (IDT-R). IDT starts with some initial candidates (N) as grid or random, then the objective values of these candidates are evaluated. The decision tree regressor is trained with the input as the initial candidates and the output as objective values. In this algorithm, DT is trained using the CART algorithm with MSE as the loss function while the tree is grown until the terminal node is pure (i.e., the terminal nodes have only one sample or many samples with the same objective values). In each iteration, the T best performed leaves or terminal nodes are selected, while the new candidates are chosen as R random points or all extreme points from each leaf. The repetitive candidates are removed before evaluating the objective function. This iterative loop (training DT, selecting T best performed leaves, selecting new candidates, removing the redundant candidates, and evaluating the new candidates) are performed until reaching the maximum iterations, or there are no new candidates. If the termination criteria are met, the best solution is chosen, and the algorithm is ended. It is worth noting that IDT algorithm can not secure the global optimal solution but a near-global solution. Therefore, IDT is suitable for HPO, where the objective functions are not too expensive or too cheap.

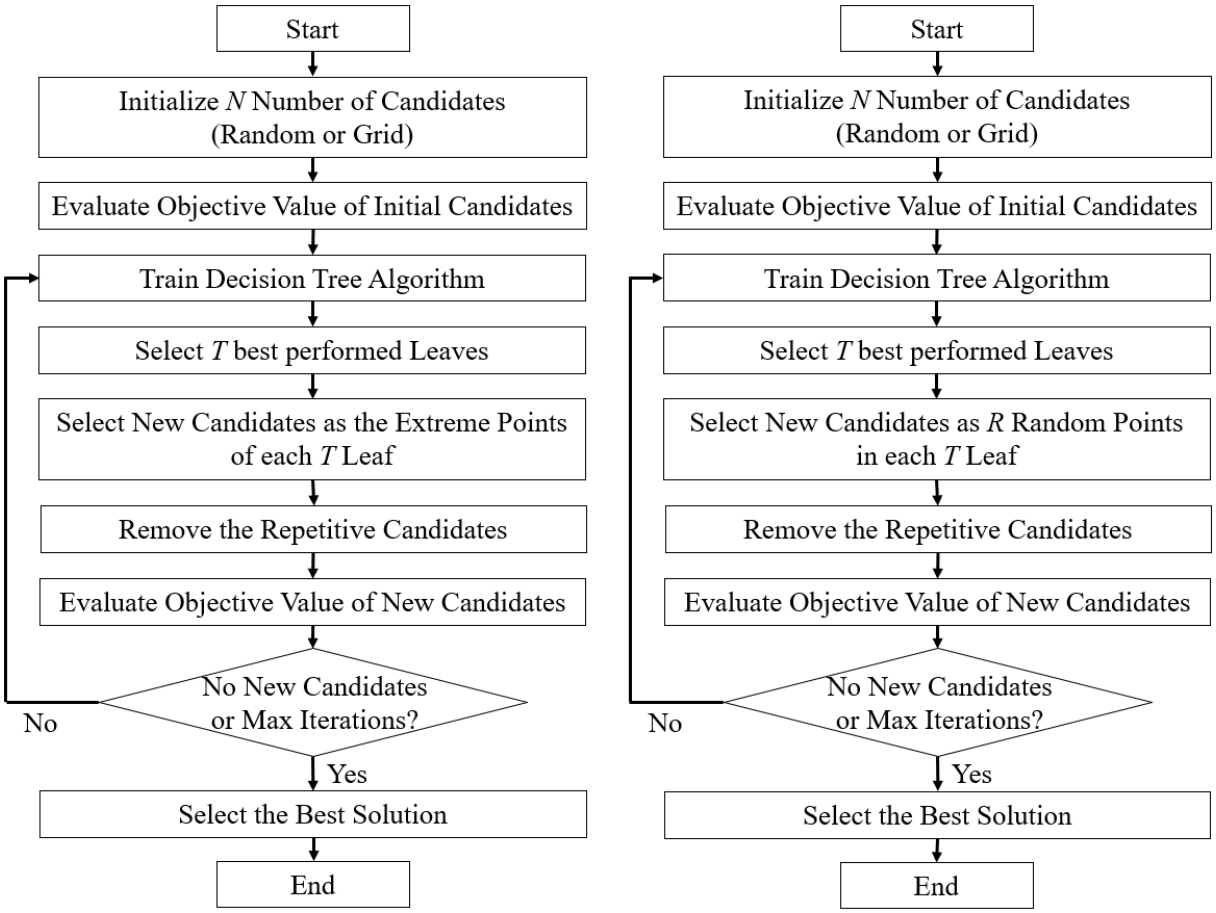

Figure 4.2 Iterative Decision Tree with new candidates as extreme points (*left*: IDT-E) and as random points (*right*: IDT-R)

To better understand the concept of our proposed methodology, two examples were provided using Equations (4.6) and (4.7). As shown in **Figure 4.3**, IDT started with six initial random points, and four new candidates were suggested from the extreme points of two top leaves. After evaluating these four new candidates, DT was retrained and suggested three new candidates (one repetitive candidate). This sequential loop was performed until reaching the stopping criteria. From one to another iteration, the newly recommended candidates were closer and closer to the global optimal location. After the fourth iteration, we got the optimal result as 0.7626 (objective of -6.00522), while the actual global result is 0.7573 (objective of -6.02074). We will get closer to the actual global result if we do several more iterations.

$$f(x) = (6x-2)^2 \sin(12x-4) \qquad x \in [0,1] \tag{4.6}$$

$$f(x,y) = -x \sin\left(\sqrt{|x|}\right) - y \sin\left(\sqrt{|y|}\right) \qquad x,y \in [-500,500] \tag{4.7}$$

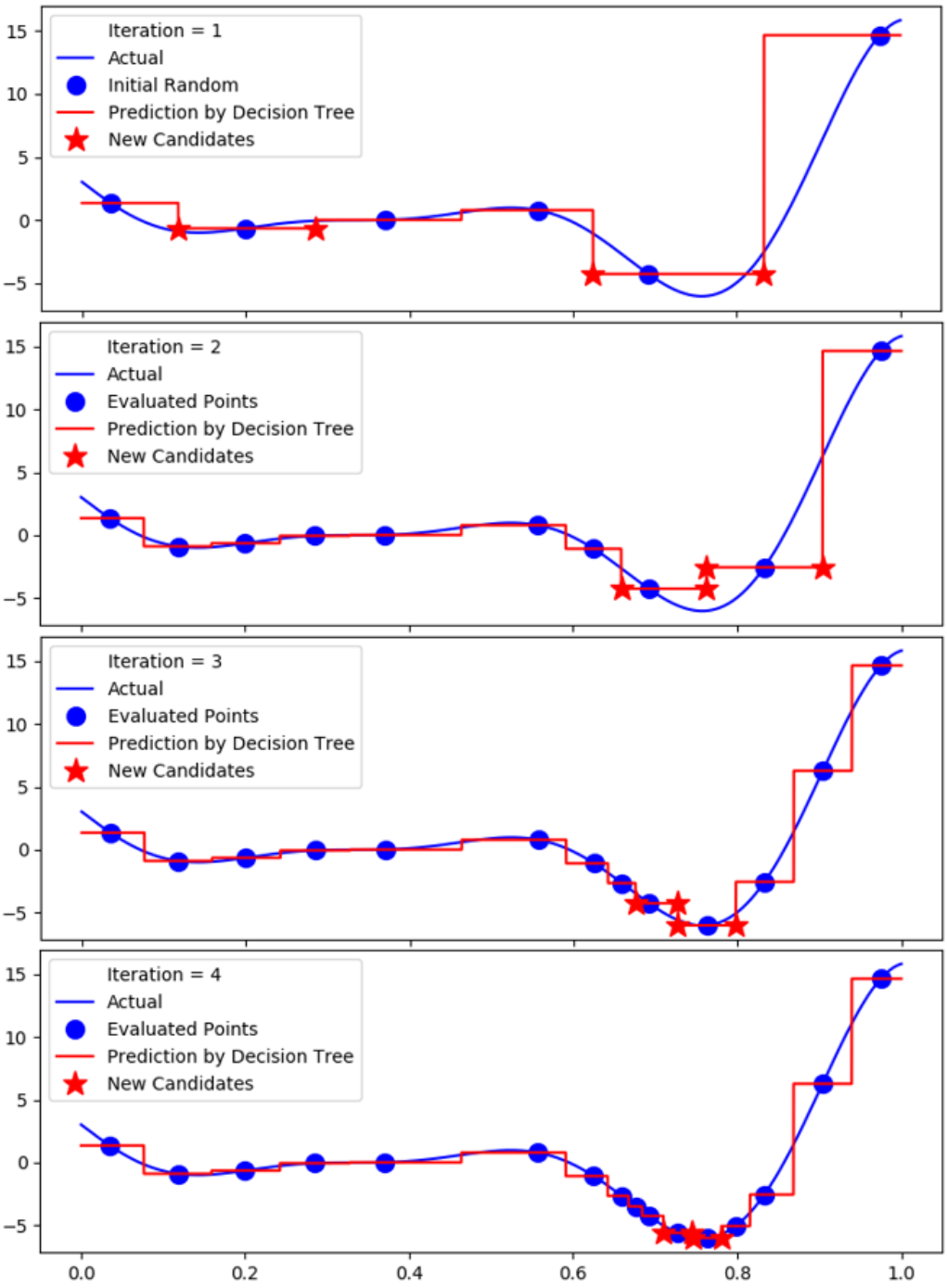

Figure 4.3 Optimization procedure for (4.6) by Iterative Decision Tree with Extreme points (IDT-E) with the parameters of 2 best-performed leaves and eight initial random points

Figure 4.4 shows the five iterations of the optimization procedure of the Schwefel function (4.7) by IDT-R with the parameters of 5 best-performed leaves, two random points in each leaf, and 100 initial random points. In this example, 100 random initial points were selected to train the DT regressor. The CART algorithm split the overall search space into 100 regions (representing 100 terminal nodes). 2 points were randomly drawn from 5 best-performed leaves scattering in many local optimal regions. Similarly, the new candidates were suggested in several regions, which represent the exploration capability of IDT-R. In the third iteration, IDT-R searched in only two regions, while IDT-R searched only in the global optimal region in the

4th and 5th iterations, i.e., exploitation. In just five iterations with total searches of 150, the new candidates were almost on the top of the global optimal solution.

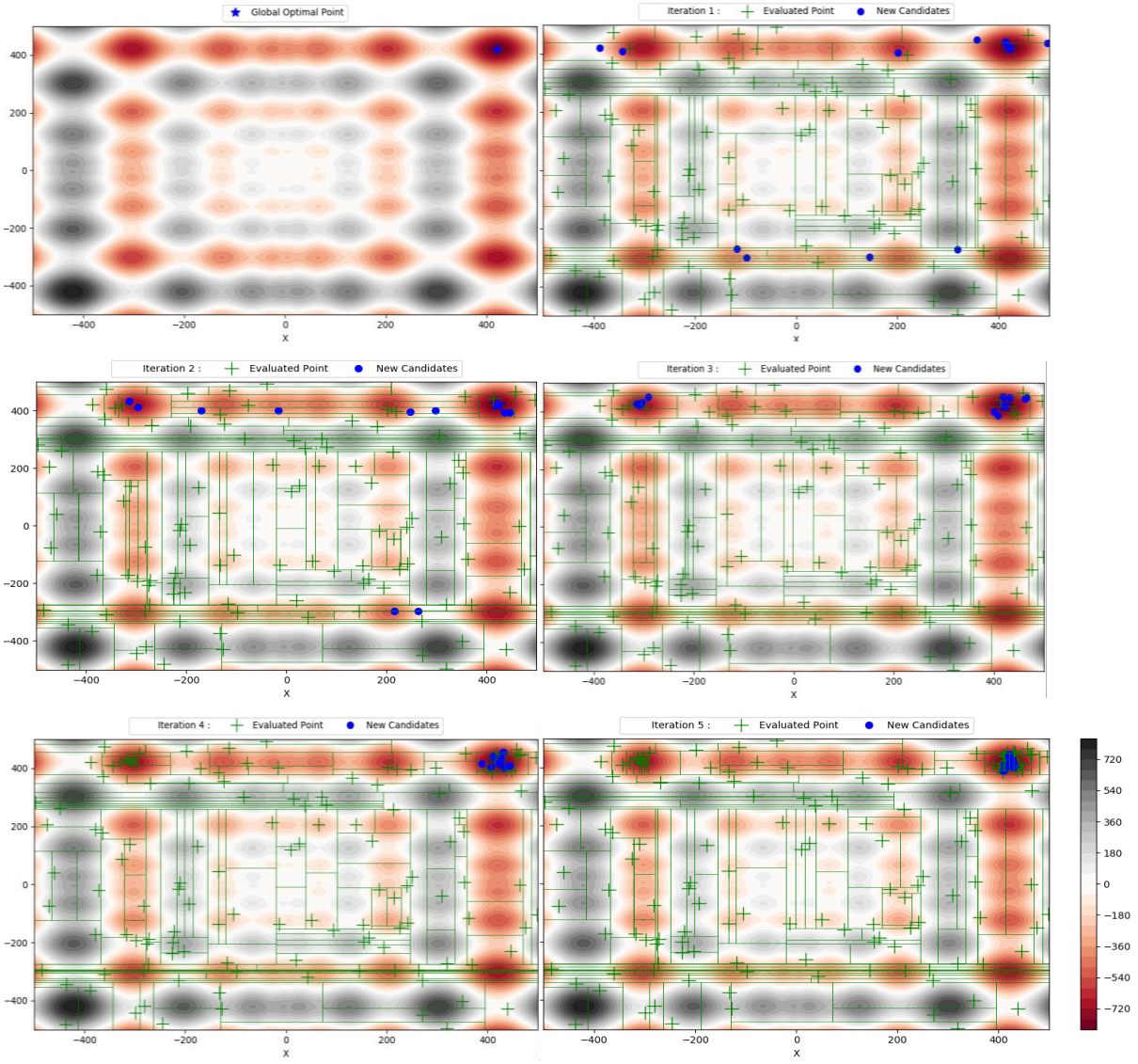

Figure 4.4 Optimization procedure for Schwefel function (4.7) by Iterative Decision Tree with Random points (IDT-R) with the parameters of 5 best-performed leaves, two random points in each leaf, and 100 initial random points

To reveal the effectiveness of the proposed approach, we will compare them with several benchmark algorithms, including Grid Search or GS, Random Search or RS, Bayesian Optimization based on Gaussian Processes with Lower Confidence Bound or GP-LCB, Bayesian Optimization based on Random Forest regressor with Lower Confidence Bound or RF-LCB [80], Tree-Structured Parzen Estimator or TPE [81], and Genetic Algorithm or GA [82], see **Table 4.2**. The comparison was examined both on the benchmark problems and the demand prediction of shared e-scooter. The benchmark problem included the optimization of nonconvex functions (Cross-in-tray, Eggholder and Styblinski-Tang functions), machine learning models (Support Vector Machine or SVM and Random Forest or RF), and deep learning models (Autoencoder or AE and Convolutional Neural Networks or CNNs). The HPO

of these models was examined using the benchmark datasets such as hand-written digits dataset [79], car evaluation dataset [83], MNIST [84], and CIFAR-10 [85]. For demand prediction of shared e-scooter, two models (RF and GRUs) were examined.

Table 4.2 Parameter settings for hyperparameter optimization algorithms

Algorithm	Parameter	Value Range	
		SVM, RF, AE, CNNs, GRUs	Nonconvex Function
GP-LCB	Number of initial random points	50 - 150	100 - 300
	Kappa	0.0 - 2.0	0.0 - 2.0
RF-LCB	Number of initial random points	50 - 150	100 - 300
	Kappa	0.0 - 2.0	0.0 - 2.0
TPE	Number of initial random points	50 - 150	100 - 300
	Number of candidates for EI	20 - 50	20 - 50
GA	Population size	5 - 30	10 - 50
	Mutation probability	0.01 - 0.5	0.01 - 0.5
	Elite ratio	0.0 - 0.1	0.0 - 0.1
	Crossover probability	0.2 - 0.7	0.2 - 0.7
	Parent portion	0.1 - 0.5	0.1 - 0.5
IDT-E & IDT-R	Number of initial random points	50 - 150	100 - 300
	Number of best-performed leaves	2 - 5	2 - 10
	Number of randoms in each leaf	1 - 3	2 - 5

The proposed algorithms (IDT-E and IDT-R) seek near-global optimization under constrained computational resources and offer the feature importance metric to better understand the effect of hyperparameters on the objective function. Additionally, these important metrics can be used to create an effective search space, for example, by designating a coarse grid to less significant hyperparameters and a wider range to those that are more important. Impurity-based feature importance, also known as Gini importance, is calculated as the normalized total decrease of the impurity criterion (for example, R2-score) [79]. In contrast to binary or categorical features, this metric tends to be quite biased and favors high cardinality features. Therefore, the relation between the factors and the response is primarily interpreted using the non-biased permutation-based feature importance. The difference in impurity score between the original data and the randomly reordered (permuted) data for each feature is used to calculate the permutation-based feature importance. Both feature importance based on

impurities and feature importance based on permutations is compared in this study, with 100 permutations.

4.4 Numerical results

4.4.1 Optimization result of nonconvex functions

Since the global optimal values of nonconvex functions are known, optimizing these functions has frequently been used to assess the efficiency of optimization techniques. Problems involving the optimization of nonconvex functions occur when any of the constraints or the objective functions are nonconvex. Several viable regions, locally optimal points, flat regions, and saddle points are characteristics of nonconvex functions. Most optimization algorithms can only offer a near-global optimal solution when working within a constrained computing budget since the global optimal point of a loss function is unknown in the real world. Three benchmark nonconvex functions were tentatively chosen based on the difficulty of the problems, which ranged from easy to challenging. Although the mathematical expression for the cross-in-tray function is complex, the fact that it only has two parameters, and four global solutions makes it a simple task. Conversely, the Eggholder function is a difficult problem because there is only one global solution, and the range of values for its two parameters is quite large. As it only has one global solution with *n* parameters, the Styblinski-Tang function was chosen to make the problem more complex. The ranges of objective value for these three functions also vary. The expression of Cross-in-tray function (**Eq. 4.8**), Eggholder function (**Eq. 4.9**), and Styblinski-Tang function (**Eq. 4.10**) are:

$$f(x,y) = -0.0001\left[\left|\sin x \sin y \exp\left(\left|100 - \frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right| + 1\right]^{0.1} \tag{4.8}$$

$$f(x,y) = -(y+47)\sin\sqrt{\left|\frac{x}{2}+(y+47)\right|} - x\sin\sqrt{|x-(y+47)|} \tag{4.9}$$

$$f(x_i) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i \tag{4.10}$$

Where the search domain of (**Eq. 4.8**), (**Eq. 4.9**), and (**Eq. 4.10**) is between $[-10, 10]$, $[-512, 512]$, and $[-5, 5]$, respectively. The Styblinski-Tang function's *n* value was set to 5, meaning that five parameters were optimized $(x_1, x_2, x_3, x_4, x_5)$.

The global minimum values of the benchmark nonconvex functions Cross-in-tray, Eggholder, and Styblinski-Tang are, respectively, -2.06261, -959.6407, and -195.8308. Except for the cross-in-tray function, which has four symmetric optimal points, all functions have a single global optimal point. Due to limited computational resources, the HPO algorithms cannot guarantee the global optimal solution for our parameters, just a close to global one. The average of each nonconvex function calculation trial for the 400 total searches represents the global convergence curve in **Figure 4.5**. IDT-E and IDT-R produce comparable outcomes when starting with an initial set of 200 random points on average, but IDT-R reaches the optimal region more quickly. Because IDT-R only generates roughly 15 new candidates per iteration compared to IDT-E's 20 (2x2x5) candidates for Cross-in-tray and Eggholder and 50 (5x2x5)

candidates for Styblinski-Tang. In other words, when solving problems with several decision variables, IDT-E engages in more exploration than exploitation. After a few iterations (inadequate exploitation) for the Styblinski-Tang function, IDT-E meets the stopping condition (maximum number of examined points). As a result, RS is just marginally superior.

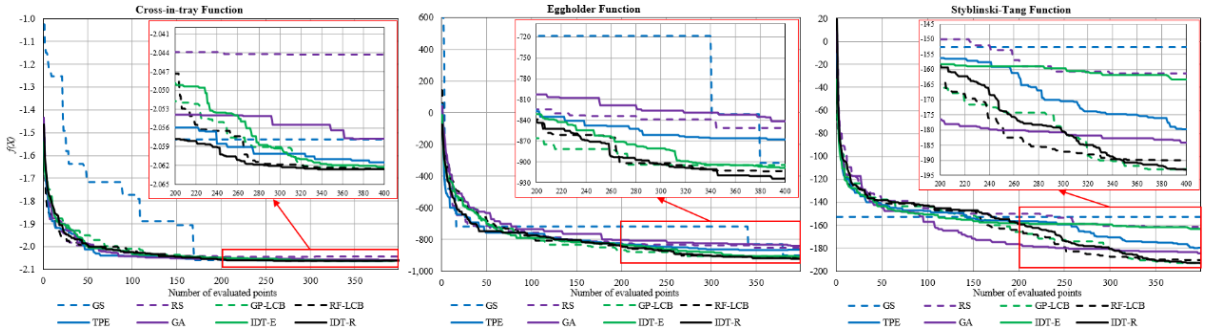

Figure 4.5 Global convergence curve (average value) of HPO algorithms for the nonconvex functions: Cross-in-tray, Eggholder, and Styblinski-Tang

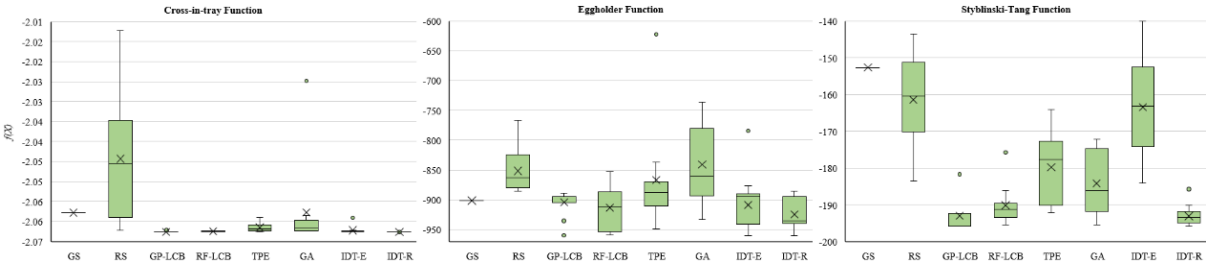

Figure 4.6 Box plot of best results of HPO algorithms for the nonconvex functions: Cross-in-tray, Eggholder, and Styblinski-Tang

The best outcome of HPO algorithms for nonconvex functions is shown in a box plot in **Figure 4.6**. The majority of HPO algorithms might easily reach one of the optimal solutions for the Cross-in-Tray function. Four algorithms—GP-LCB, RF-LCB, IDT-E, and IDT-R—achieved the best result in this case, with IDT-R having the greatest average value and the lowest variance (i.e., high stability). With regard to the Eggholder function, the best results from all algorithms had a comparatively high standard deviation, presumably due to local optima or insufficient exploitation. GA performed similarly to RS, although with a larger variance. Due to the small population size in some trials, GA may have achieved favorable outcomes but also fell into local optima. IDT-R performed better than other algorithms overall in terms of mean and standard deviation. Although IDT-R had the best performance, the Styblinski-Tang function was similar to the first two functions in terms of how GP-LCB, RF-LCB, and IDT-R performed. The following algorithms have the shortest average computation times: GS (0.01s), RS (0.06s), GA (0.56s), IDT-E (0.66s), IDT-R (1.01s), TPE (2.74s), RF-LBC (59.72s), and GP-LCB (817.67s).

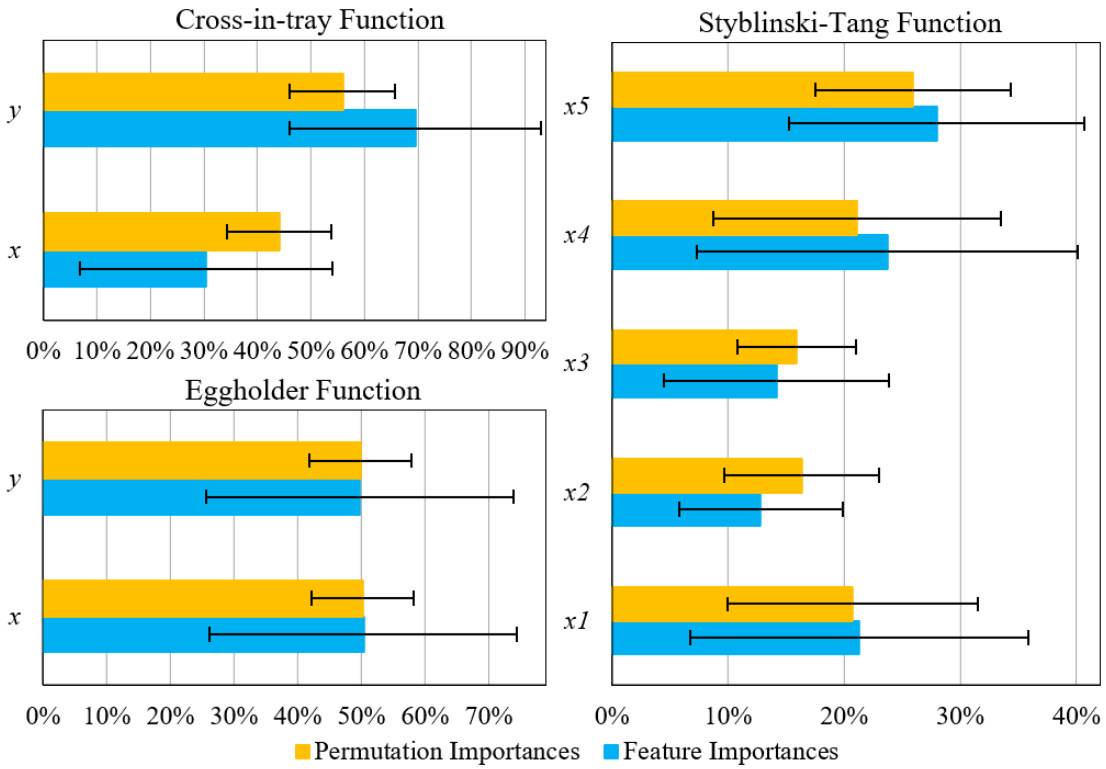

Figure 4.7 Mean and STD of feature importance metrics by IDT-R for each parameter of the nonconvex functions: Cross-in-tray, Eggholder, and Styblinski-Tang

The feature importances of the functions with 2 and 5 were 50% and 20%, respectively, based on a large number of random points in the search space. The importances of the features based on impurity and permutation were equal in this instance. **Figure 4.7** illustrates how permutation importances are much closer to the ideal value and have a lower variation than feature importances (i.e., impurity-based feature importances). As a result, the permutation importances should be used as the foundation for this metric's interpretation.

4.4.2 HPO result of SVM for hand-written digits dataset

Suitable for both classification and regression, Support Vector Machine (SVM) is a reliable supervised learning technique. SVM is based on the idea of decision planes, which use decision boundaries or hyperplanes to best divide the data into various categories. SVM can handle various classification or regression issues because of the flexibility and simplicity of the model, including high dimensional spaces, small datasets with a greater number of features, and handling both linear and nonlinear data [79]. Detail SVM formulas may be found in [78]. The C-Support Vector Classification Python module from the Scikit-learn packages was used to train this model [79]. The kernel coefficient ($gamma\ \epsilon\ [0.001, 1]$) and the regularization parameter ($C\ \epsilon\ [0.001, 10]$) were the two hyperparameters of SVM that were optimized in this study. Other hyperparameters were set to their default values, with the decision function for multi-class classification being one-vs-one and the kernel function being the Radial Basis Function (RBF). The hand-written digits data from Scikit-learn was used to train the SVM in this study, see **Figure 4.8**. The images in this dataset are 8x8 grayscale images labeled 0 - 9.

Model training (50%) and model evaluation (50%) were randomly selected from the total sample 1797.

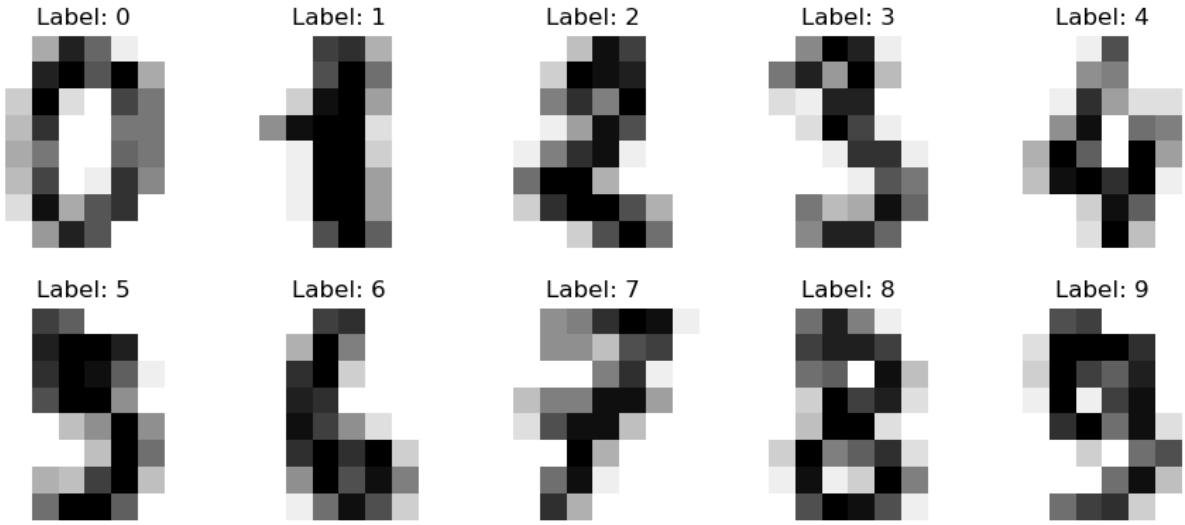

Figure 4.8 Hand-written digits dataset [79]

The digit classification numerical results of SVM's HPO are shown in **Figure 4.9**, along with a box plot of the best solutions and a global convergence curve by computation time for the 200 evaluation points that were included. TPE performs well initially because it converges faster than other algorithms but finds it difficult to depart the local optima. Because they mostly depend on random searches, other HPO algorithms perform similarly for the first 100 searches (about 10 seconds). Most of these algorithms are stuck in local optima that resemble TPE after this stage, except IDT-R, which consistently ascends to the global optimal point. Similar to previous problems, the population-based algorithm (GA) performs relatively poorly, even worse than RS, because it is prone to falling into local optimal points because of the small population size and the insufficient number of generations.

The performance of IDT-E was comparable to that of other sequential-based algorithms like TPE, GP-LCB, and RF-LCB because just two hyperparameters were optimized. SVM's benchmark performance on this dataset was 96.89% [79], while 97.44% accuracy was attained through the hyperparameter SVM optimization. IDT-R, on average, completed calculations in 18.3 seconds with a 97.42 percent accuracy. TPE performed at 97.36 percent with a training time that was marginally faster (16.3s). The longest computation time was needed for GP-LCB (104s), followed by RF-LCB (42.6s), and GA (32.5s). For roughly 21s, training was required for GS, RS, and IDT-E. Because they were trained with fewer evaluated points than the maximum number of searches (200), IDT-E and IDT-R had faster training times than other model-based approaches. The regularization parameter ($C = 3$) and kernel coefficient ($gamma = 0.2$) were the SVM's global optimal hyperparameters for the digit classification dataset.

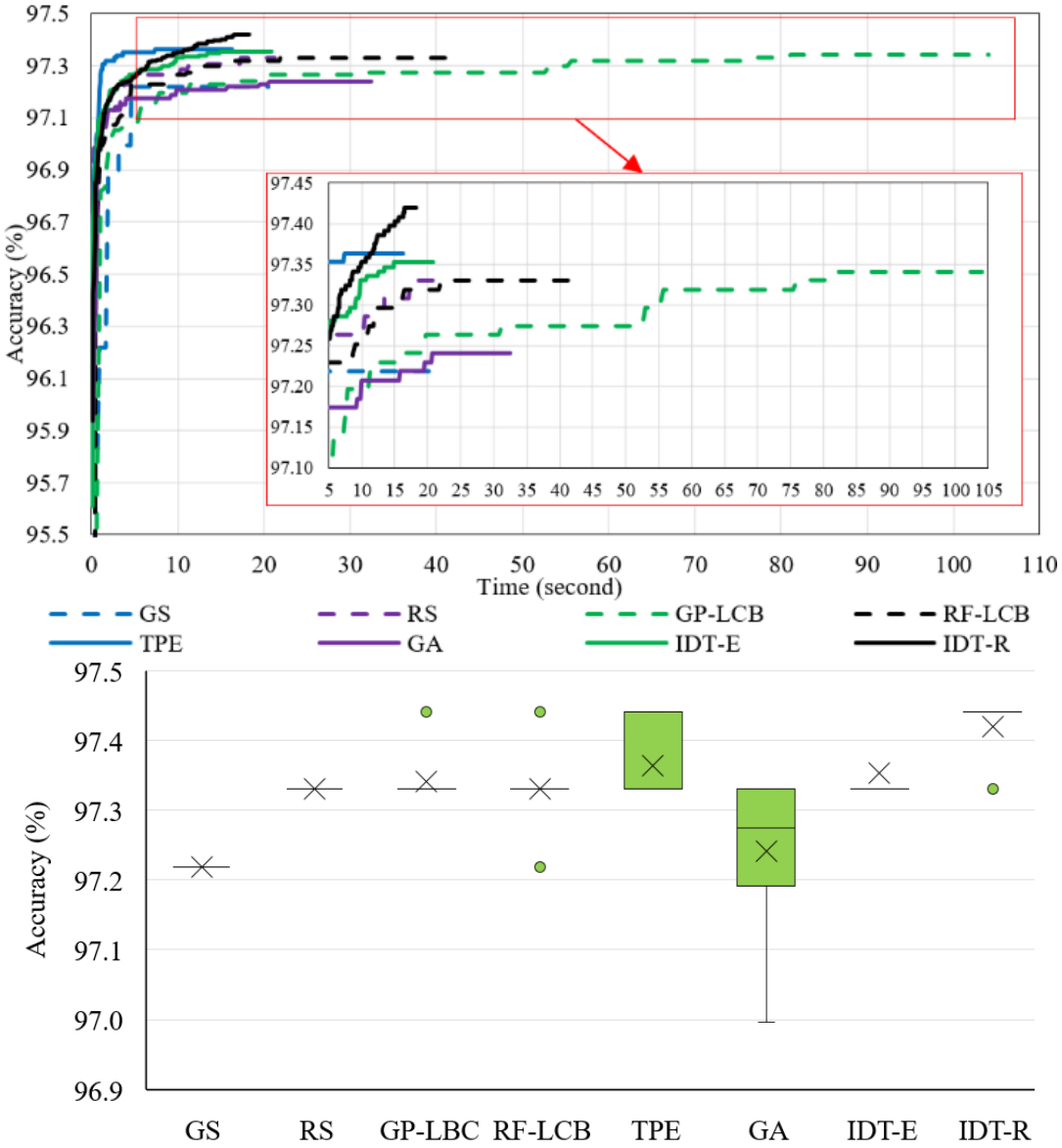

Figure 4.9 Numerical results of SVM's HPO for digit classification: (*top*) global convergence curve and (*bottom*) Box plot of best results of HPO algorithms

Based on a large random sample, the hyperparameters, *C* and *gamma*, of the SVM have feature importances of 80% and 20%, respectively. The average permutation importance and feature importance of gamma from 20 trials of IDT-R with 200 evaluated points was 43.29 and 42.80 percent, respectively. Due to the relatively small sample size (200 samples) in comparison to the prior section (400 samples), there is a notable deviation from the ideal values. The interpretation of the IDT-R algorithm's feature importances should be based on a large enough sample size.

4.4.3 HPO result of RF for car evaluation dataset

As explained in Chapter 3, Breiman's Random Forest (2011) is a potent machine-learning technique that can handle highly dimensional data with minimal training data. RF is the average (for regression) or majority vote (for classification) of hundreds of predictions made by a decision tree using randomly chosen inputs or feature combinations. RF can handle the

overfitting of traditional decision trees for classification and regression issues as the bootstrap aggregation (or bagging) of weak tree learners. The Random Forest Classifier Python module from the Scikit-learn packages was used to train the random forest in this study [79]. Three hyperparameters of RF, including the number of trees in the forest ($n\_estimators \epsilon [1,\ 400]$), the maximum depth of the tree ($\max\_depth \epsilon [1, 20]$), and the criterion function (Gini impurity or entropy) was optimized in this section. RF's trees were constructed using bootstrap samples, with a minimum of 2 samples needed for internal node splitting and 1 sample in each leaf node. The square root of the total number of features was used to determine how many features should be used for the optimal split.

Table 4.3 Car evaluation dataset [83]

Items	Attributes
Evaluation classes	Unacceptable, Acceptable, Good, Very Good
Buying price	Low, Medium, High, Very High
Maintenance price	Low, Medium, High, Very High
Number of doors	Two, Three, Four, Five or more
Capacity as # persons	Two, Four, More
Size of luggage boot	Small, Medium, Big
Estimated safety of car	Low, Medium, High

With the abovementioned settings, a random forest classifier was trained on a benchmark dataset, car evaluation, as in **Table 4.3** from the UCI machine learning repository [83]. Seven elements and 1728 instances make up this dataset. Unacceptable, acceptable, good, and very good are the four evaluation categories. The cost of purchasing and maintaining the car, the number of doors, the number of people it can carry, the size of the luggage boot, and the projected level of safety are the six categorical explanatory variables. The complete dataset was, by default, randomly divided into two portions for model training (70%) and model evaluation (30%).

Figure 4.10 depicts the numerical outcomes of hyperparameter tuning for car evaluation by random forest classifier as the box plot of the best outcome for each HPO algorithm as well as the global convergence curve by computational time. The performance of RF varies from 67.82 percent to 97.11 percent (the global optimal result) in this search boundary. As can be seen, the majority of HPO algorithms, including TPE, GA, and RF-LCB, were stymied in two local optima at 96.72 and 96.92 percent. All algorithms but GP-LCB and IDT-R fared better on average than IDT-E. Although the performance of these two algorithms was comparable, IDT-R found the ideal solution after about 150 searches (40s), whereas GP-LCB took 195 searches to do so (170s). The optimum RF hyperparameters for the car evaluation dataset, for an accuracy of 97.11 percent, were 77 for the number of trees in the forest, 11 for the maximum depth of the tree, and the Gini impurity criteria.

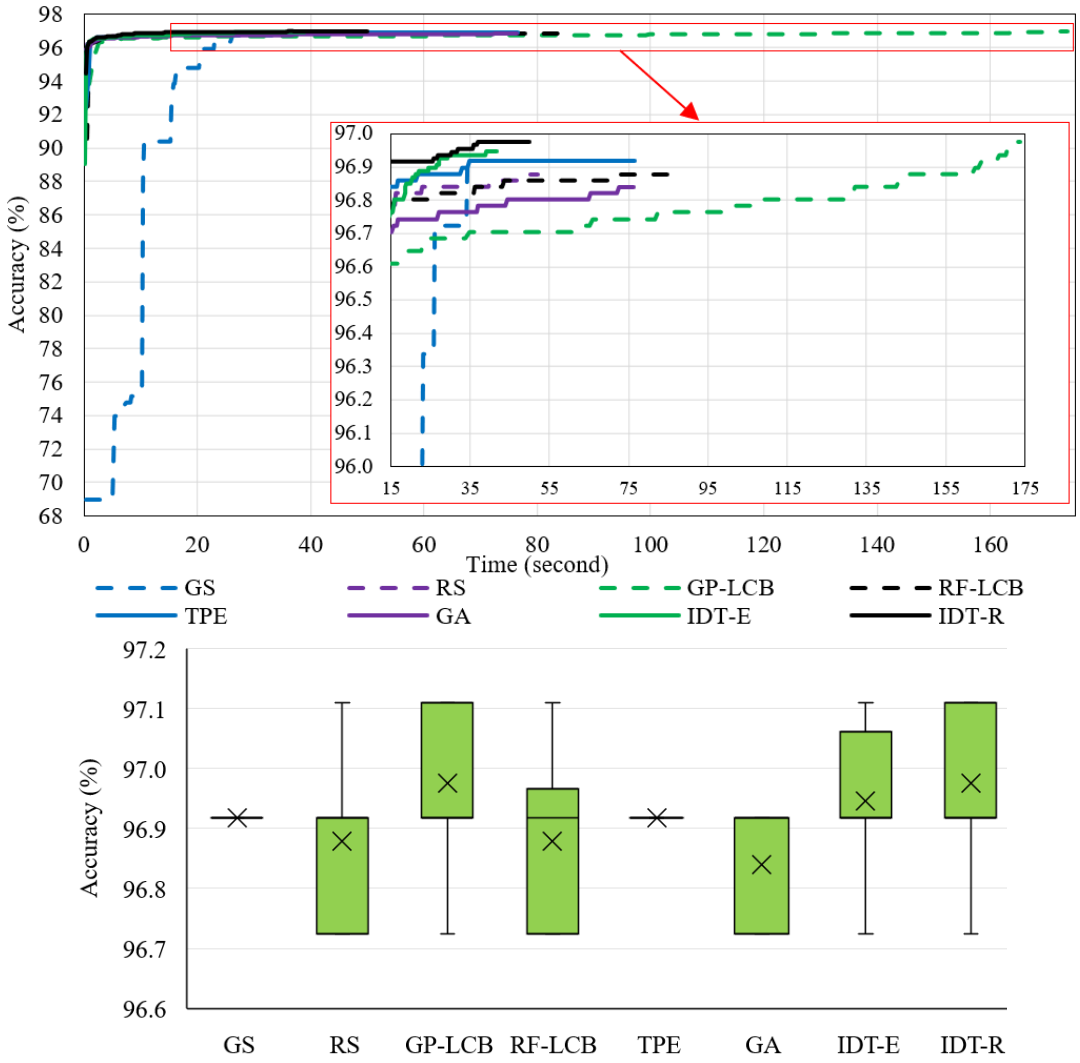

Figure 4.10 Numerical results of RF's HPO for car evaluation dataset: (*top*) global convergence curve and (*bottom*) Box plot of best results of HPO algorithms

These HPO algorithms are ranked according to their computational times as follows: IDT-E (41.6s), IDT-R (50s), GS (50.8s), RS (52s), GA (76.4s), TPE (76.5s), RF-LCB (86.25s), and GP-LCB (173.7s). Because they did not require iteratively updating the surrogate function like the sequential-based approaches or training the repetitive candidates (i.e., they evaluated less than 200 points in some trials), IDT-E and IDT-R had shorter training times than algorithms, similar to HPO of SVM. The three hyperparameters (number of trees, maximum depth, and criterion) had real feature importances for this problem setting of 0.65 percent, 99.15 percent, and 0.2 percent, respectively. These three hyperparameters had average impurity-based feature importances from 20 HPO trials of 1.03 percent, 98.83 percent, and 0.14 percent, respectively. The permutation importances were 1.09 percent, 98.72 percent, and 0.18 percent. As a result, the performance of RF is substantially influenced by the hyperparameter, the maximum depth of the tree.

4.4.4 HPO result of AE for MNIST dataset

The term "autoencoder" refers to a collection of unsupervised neural networks specifically employed to extract the significant characteristics of data, possibly for dimensionality reduction, anomaly detection, data denoising, information retrieval, image inpainting, and other similar tasks. The input layer connects to one or more successively smaller layers (Encoder), followed by successively bigger layers (Decoder), which connect to the output layer in AE. The original data must pass through the middle layer, which acts as a bottleneck and stores the compressed knowledge representation, in order to minimize the difference between the input and output layers. Many different autoencoder models exist, including fully connected, convolutional, sequence-to-sequence, and variational autoencoders.

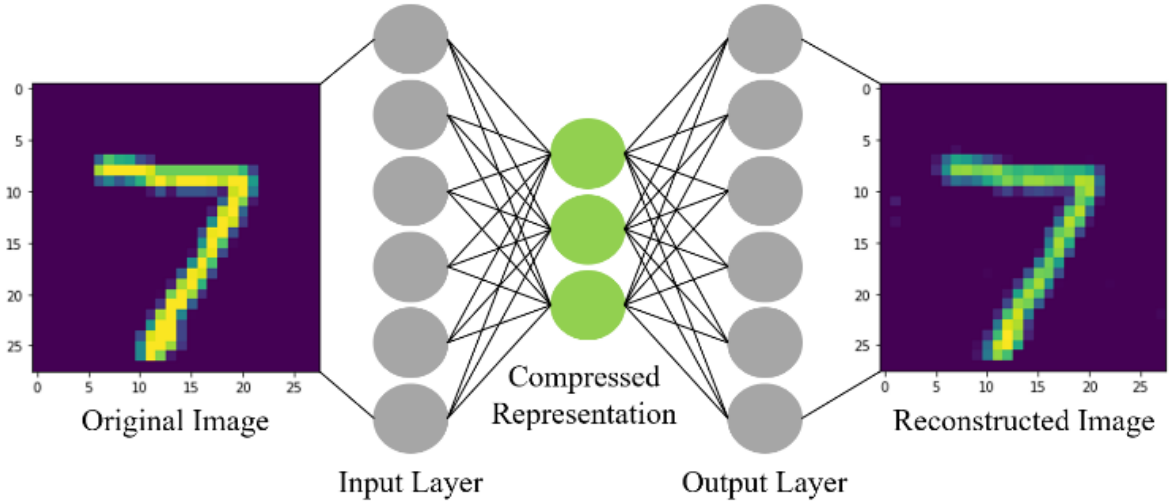

Figure 4.11 Architecture of Autoencoder as dimensionality reduction for MNIST dataset

This research uses an easy, fully-connected architecture (see Figure 5.7) as the autoencoder, similar to [37]. The input layer connected to the compacted layer required the flattening of the original images. The output layer, which could be used to recreate the original image, received the representative properties retrieved by the bottleneck layer before being transferred to it. This fully connected autoencoder was trained using the MNIST dataset [84] with the MSE as the loss function. 70,000 handwritten digits from 0 to 9 in 28x28 pixel images make up this dataset. 10,000 data were utilized to evaluate the model, while the remainder was used to train the AE architecture. A Jupyter Notebook was used to run this architecture using Keras and TensorFlow packages. The number of nodes and activation in the output layer, learning rate of the stochastic gradient descent (SGD) optimizer, batch size, and activation of the encoding layer were the five hyperparameters of AE optimized in this study. In order to minimize the data features by up to 50%, the Hidden Layer (HL) was searched between 4 and 400 nodes. ReLU (R), Sigmoid (S), and Tanh (T) functions were taken into consideration as three activation functions for the HL and Output Layer (OL). 0.001 to 0.99 and 64 to 1024, respectively, were chosen to search for the optimal learning rate and batch size. The patience and min_delta values for the EarlyStopping were 3 and 0, respectively, and there were 50 epochs. Mean Square Error (MSE) was used as the AE's loss function, and the SGD optimizer's momentum was adjusted to zero.

As shown in **Figure 4.5** and **Figure 4.6** for the Styblinski-Tang function, IDT-E performed poorly since it wasn't suited for optimizing the HPO with several hyperparameters. Therefore, the five hyperparameters of the autoencoder (AE) were not optimized using IDT-E. **Figure 4.12** displays the performance of the HPO algorithms (GS, RS, GP-LCB, RF-LCB, TPE, GA, and IDT-R) comprising the feature importances, the global convergence curve by computational time, and a box plot of the best results from HPO. The random search performed just slightly better than the coarse grid of the hyperparameter space. The performance of GA, RF-LCB, and RS was only somewhat superior to RS with the current AE settings. The GP-LCB, TPE, and IDT-R performed comparably in this problem. TPE, on the other hand, converged more quickly than other algorithms, around the 50th search (about 25 minutes), before sluggishly getting closer to the overall optimal position. In contrast to earlier problems, AE was not entirely stable, meaning that when we run the same configuration of AE multiple times, the outcomes frequently differ. In light of the possibility that these two methods trained the same AE configurations, certain GP-LCB and TPE trials had better validation loss than IDT-R trials.

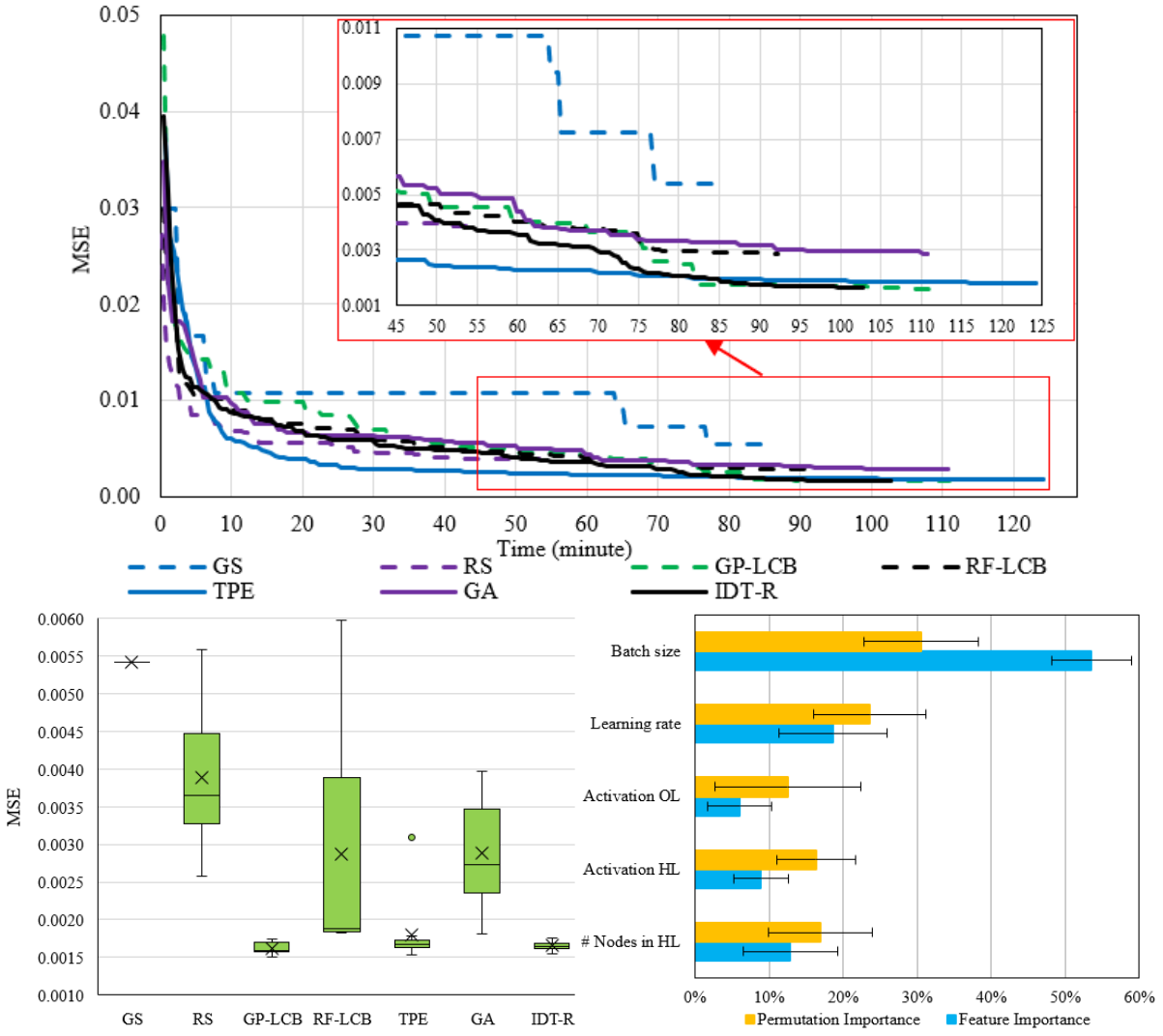

Figure 4.12 Numerical results of Autoencoder's HPO for MNIST dataset: (*top*) Global convergence curve, (*bottom-left*) Box plot of best results of HPO algorithms, and (*bottom-right*) Mean and STD of feature importances by IDT-R

The optimal hyperparameters were 400, Tanh, ReLU, 0.99, and 64 for Hidden Layer's (HL) number of nodes, HL's activation, Output Layer's (OL) activation, Learning Rate, and batch size, respectively. This implies that the amount of information lost decreases with increasing latent space (number of HL nodes). With the smallest batch size and greatest learning rate, AE achieved the lowest validation loss under the constrained number of epochs. However, it took more time to compute because of the smaller batch size. Because of this, TPE had the longest computing time—124.19 minutes—for the 200 total assessed points, as it converged more quickly than other methods, see **Figure 4.12**. For the 243 searches (3 coarse grids for each hyperparameter) in total, GS was trained for 85.5 minutes. GP-LCB took an average of 111 minutes to train, followed by GA (110.9 minutes), IDT-R (102.7 minutes), RF-LCB (92.3 minutes), and RS (54.6 minutes). Even though parallel computing can be used to speed up the computation of GA and IDT-R.

The feature and permutation importances are also shown in **Figure 4.12**. The impurity-based feature importances demonstrate that batch size (53.6%) had a significant influence on AE's performance, followed by learning rate (18.6%), HL's number of nodes (13.9%), HL's activation (8.9%), and OL's activation (6%), in that order. Contrarily, permutation-based feature importances increased this metric for the other four hyperparameters, such as learning rate (23.6%), HL's number of nodes (17%), HL's activation (16.4%), and OL's activation (12.5%), while decreasing the importance measure of batch size to 30.5 percent.

4.4.5 HPO result of CNNs for CIFAR-10 dataset

Convolutional neural networks (CNNs) are a popular variety of deep neural networks used in many different applications, particularly image recognition and computer vision tasks. Convolution and pooling are the two specific operations that makeup CNNs. As narrated in [86], LeNet, AlexNet, VGGNet, GoogLeNet, ResNet, and SENet are only a few examples of the potent deep convolutional neural network architectures that exist.

Modern performance in face recognition and picture classification is achieved by the straightforward deep convolutional neural network known as the Visual Geometry Group (VGGNet). Four learnable layers in a straightforward VGGNet were explored in this study [86]. As shown in **Figure 4.13**, our VGGNet design comprises the following layers: the output layer, two convolutional layers (CL), one max pooling layer, two dropout layers, and one fully connected layer (FCL). Using the CIFAR-10 dataset [84], this classification architecture was developed. 60,000 32x32 pixel color pictures of items from 10 classes make up this dataset, 50,000 for training and 10,000 for testing. The Keras and TensorFlow platforms were used to build the model. The majority of the model's hyperparameters were trained using default values, while 14 hyperparameters were optimized using the range depicted in **Table 4.4**. The Stochastic Gradient Descent (SGD) optimizer was used to optimize the parameters of CNNs, while its two parameters, momentum and learning rate, were tuned in the range as in **Table 4.4**. There were 50 epochs used, and categorical cross-entropy was the loss function.

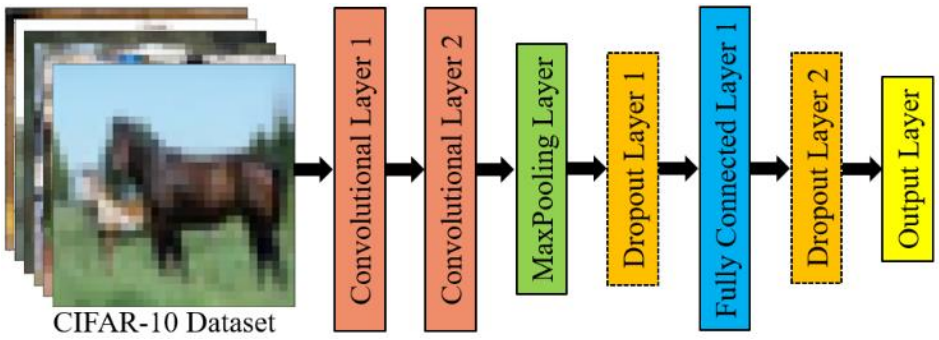

Figure 4.13 Architecture of Convolutional Neural Networks (CNNs) for the CIFAR-10 dataset

Table 4.4 Hyperparameter range for convolutional neural networks (CNNs)

Hyperparameter	Range	Hyperparameter	Range
Filter of CL1	8 - 128	Dropout rate 1	0 – 0.8
Kernel of CL1	2 - 10	Node in FCL	8 - 1024
Activation of CL1	R, S, T	Activation of FLC	R, S, T
Filter of CL2	8 - 128	Dropout rate 2	0 – 0.8
Kernel of CL2	2 - 10	Learning rate	0.001 – 0.99
Activation of CL2	R, S, T	Momentum	0.001 – 0.99
Pooling size	2 - 10	Batch size	64 - 1024

The importance metrics for the hyperparameters of CNNs are shown in **Figure 4.14,** along with the global convergence curve by training time. TPE converged quicker than other methods, similar to Autoencoder's HPO, before progressively moving into a near-global optimal point. At 73.95 percent and 78.98 percent accuracy, respectively, GS and RS remained stable. In this case, TPE converged more quickly than other algorithms, but it also had an easy time entering the local optima, with an average accuracy of 80.75% for the 300 total examined points. GP-LCB, RF-LCB, GA, and IDT-R all had accuracy ratings of 79.75 percent, 80.10 percent, 79.95 percent, and 80.21 percent, respectively. Compared to the baseline setup, which had an accuracy of just 67.07 percent [86], the optimized configuration of CNNs performed substantially better. However, several other hyperparameters could be optimized to achieve even higher accuracy, such as batch normalization, data augmentation, deeper configuration (including more VGG blocks), etc.

The first convolutional layer of CNNs had the best filter, kernel, and activation settings of 89, 2, and ReLU, while the second convolutional layer had the best filter, kernel, and activation settings of 118, 7, and Tanh. The fully connected layer of CNNs has the optimal nodes of 934 and Sigmoid as the activation function. Other ideal CNN hyperparameters included the batch size of 113, learning rate of 0.107, momentum of 0.537, dropout rate_1 of 0.564, dropout rate_2 of 0.437, and pooling size of 7. Longer training time is needed for CNNs' optimal architecture than for random architecture. Consequently, this TPE trial required 57.2 hours of training in total, compared to 32.81 hours for the other four trials. Due to utilizing 243 coarse grid points during training, GS required the least time—roughly 21.5 hours. The computation times for RS and GA were 31.85 and 32.49 hours, respectively. The GP-LCB required an average training duration of 36.55 hours, whereas the RF-LCB needed 36.28 hours. IDT- R's computation time

for the 300 total evaluated points is 33.10 hours, comparably less than other sequential-based algorithms (GP-LCB, RF-LCB, and TPE).

Based on feature importance, it is clear that the parameters, learning rate, and momentum of the SGD optimizer significantly impacted the performance of CNNs. These two hyperparameters had permuted-based feature importances of 28.08 percent and 15.19 percent, respectively, with FCL activation coming in third (12.94 percent). The number of FCL nodes (5.47%), batch size (5.43%), activation of CL1 (5.81%), and dropout rate 2 (4.98%) were a few additional relatively significant features. Other hyperparameters, such as the filter of CL2 (4.17 percent), pooling size (4.15 percent), filter of CL1 (3.90 percent), kernel of CL2 (2.96 percent), dropout rate 1 (2.58 percent), kernel of CL1 (2.47 percent), and activation of CL2 were less significant (1.87 percent).

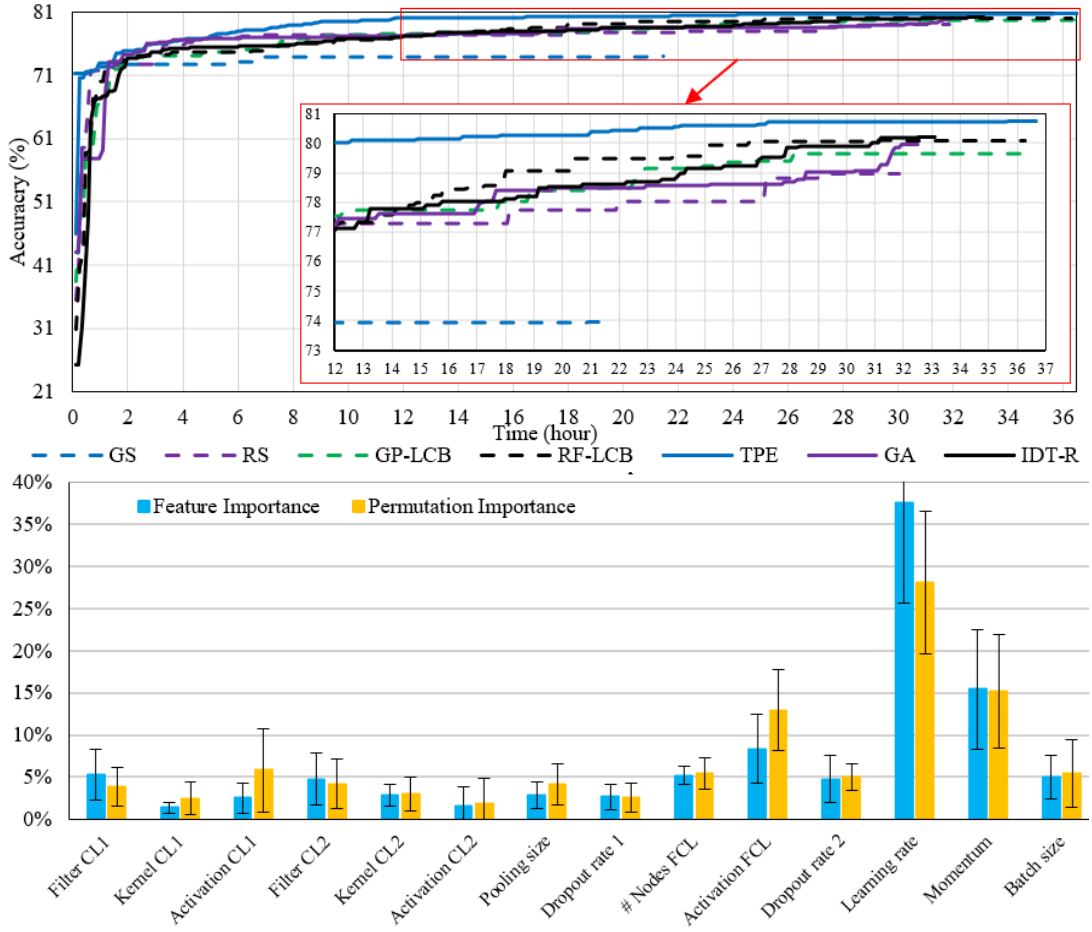

Figure 4.14 Numerical results of CNNs' HPO for CIFAR-10 dataset: (*top*) Global convergence curve and (*bottom*) Mean and STD of feature importances of CNNs by IDT-R

4.4.6 HPO result of RF and GRUs for shared e-scooter demand prediction

This section employed IDT-R to optimize the hyperparameters of random forest (RF) and gated recurrent units (GRUs) for the total hyperparameters of 4 and 10, respectively. To control the overfitting of these two models, the objective function is the summation of the MSE of evaluation data and the ratio of MSE of evaluation data and that of training data, i.e., $MSE_{eval} + MSE_{eval}/MSE_{train}$. Both RF and GRUs were trained with multiple outputs and normalized

scale. The hyperparameters of these two models (RF and GRUs) were optimized for the total evaluation points of 200. The number of running trials was 10 for IDT-R and 5 for other benchmark algorithms.

Four hyperparameters of RF were optimized, such as lookback length $[10, 170]$, sampling rate $[1, 24]$, number of trees in the forest $[10, 400]$, and maximum depth of the tree $[1, 10]$. In the case of GS, four grid points of these four hyperparameters were selected, resulting in 256 RF models being evaluated. On the other hand, the architecture of GRUs was the stack of the input layer with GRU cell, first dropout layer, one hidden layer with GRU cell, second dropout layer, and the output layer with conventional neurons and ReLU activation function. Therefore, the ten hyperparameters of GRUs are lookback length $[10, 170]$, sampling rate $[1, 24]$, number of nodes of input layer $[10, 512]$, activation function of input layer (ReLU, Sigmoid, and Tanh), dropout rate_1 $[0, 0.8]$, number of nodes of hidden layer $[10, 512]$, activation function of hidden layer (ReLU, Sigmoid, and Tanh), dropout rate_2 $[0, 0.8]$, learning rate $[0.0001, 0.01]$, and batch size $[8, 512]$. GRU GS of GRUs was trained for 216 combinations as some hyperparameters were fixed, such as sampling rate as 1, dropout rate_1 as 0.1, activation function of hidden layer as ReLU, and dropout rate_2 as 0.05. Other hyperparameters were selected for 2 or 3 each, such as lookback length $(36, 88, 140)$, number of nodes of input layer and hidden layer $(93, 259, 425)$, activation function of input layer (ReLU and Tanh), learning rate $[0.0026, 0.0076]$, and batch size $[152, 392]$.

4.4.6.1 Thammasat TH dataset

Figure 4.15 shows the convergence curve of HPO results of RF and GRUs for hourly shared e-scooter demand prediction using Thammasat TH dataset. In the case of RF, IDT-R achieved the lowest objective value (30.51) at a reasonable computational time of 125 minutes. TPE achieved a similar performance (30.78) but required the highest computational time of 232 minutes. In this case, RF-LCB performed better than GP-LCB in both performance (31.06 vs. 31.39) and training time (91 vs. 157 minutes). GA had a relatively short training time (66 minutes) while achieving the objective value of 31.63. GS and RS had the lowest training time, 28 and 32 minutes, respectively, but their performances are quite poor, i.e., 35.89 and 32.53, respectively. With the setting architecture of GRUs mentioned above, RF achieved slightly better performance compared to GRUs in both optimal objective value and computational time. For an instant, the performance of GRUs optimized by TPE had the lowest objective value of 30.95 for the computational time of 149 minutes. IDT-R and RF found a similar optimal value of 31.42 and 31.76, respectively, with a training time of 154 and 137 minutes. GA and GS had a similar objective value of 33.67 and 33.54, respectively, but GA had a shorter training time of only 72 minutes compared to 279 minutes by GS. RS had the shortest training time of 47 minutes but a poor objective value of 34.80. Interestingly, GP-LCB has the worst performance for both training time (366 minutes) and objective value (35.79).

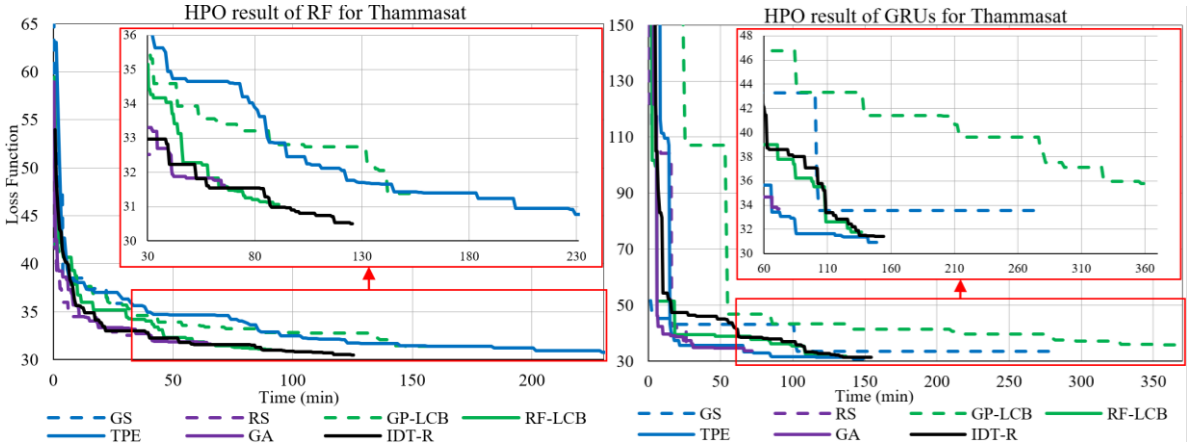

Figure 4.15 Numerical results of HPO of RF (*left*) and GRUs (*right*): Thammasat dataset

4.4.6.2 Minneapolis MN dataset

Figure 4.16 shows the convergence curve of HPO results of RF and GRUs for hourly shared e-scooter demand prediction using Minneapolis MN dataset. Difference from Thammasat, the performance of GRUs performed better than RF in both optimal objective value and computational time. Similar to the previous problem, IDT-R achieved the lowest objective value (64.89) with a computational time of 295 minutes. TPE and RF-LCB had similar objective values, 66.86 and 66.08, respectively, but TPE had more than twice the training time compared to RF-LCB, i.e., 670 vs. 258 minutes, respectively. RS, GP-LCB, and GA had comparable performances of 69.52, 70.21, and 70.51, with a training time of 127, 769, and 213 minutes, respectively. GS has the shortest training time, 89 minutes, but also had the worst performance (81.33). In the case of GRUs, IDT-R also achieved the best objective value of 54.80 with a training time of 198 minutes. TPE achieved a comparable objective value (55.50) which a shorter computational time of 151 minutes. RF-LCB performs better than GP-LCB in both objective values (57.40 vs. 63.32) and training time (186 vs. 251 minutes). GA had the shortest training time, 58 minutes, with a fairly poor optimal objective value of 59.40. GS achieved a relatively good objective value (55.64) but consumed the longest training time of 622 minutes. RS had the worst objective value of 61.11, with a training time of only 93 minutes.

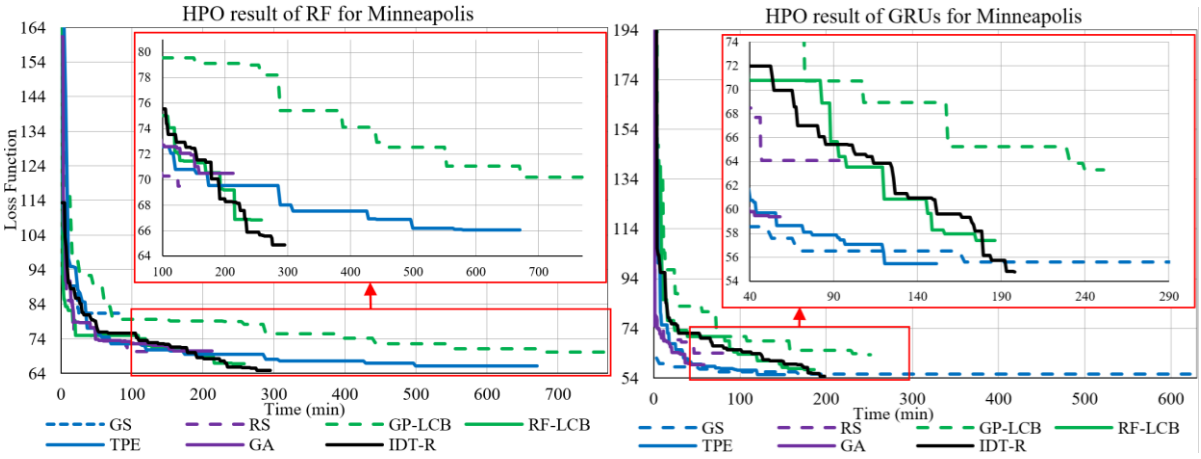

Figure 4.16 Numerical results of HPO of RF (*left*) and GRUs (*right*): Minneapolis dataset

4.4.6.3 Austin TX dataset

Figure 4.17 shows the convergence curve of HPO results of RF and GRUs for hourly shared e-scooter demand prediction using Austin TX dataset. Similar to Minneapolis dataset, GRUs provided better performance in both objective value and training time. This means that RF is more susceptible to the number of data (inputs and outputs) than GRUs. On other words, RF is not favorable for the prediction with multiple outputs, while the computational time exponentially increases in function of the number of data. For HPO of RF, RF-LCB, IDT-R, and TPE had the comparable optimal objective values (240.19, 241.33, and 241.60, respectively), while their training time were 1279, 810, and 2531 minutes, respectively. These three algorithms also had similar objective values (164.78, 162.32, and 158.28, respectively) in optimizing the hyperparameters of GRUs for the training time of 309, 408, and 604 minutes, respectively. RS, GP-LCB, and GA had similar objective values of around 257, but RS had a shorter training time of approximately 815 minutes compared to 3264 and 2437 minutes for GP-LCB and GA, respectively. The objective values of these algorithms for HPO of GRUs were 179.01, 171.79, and 187.49, with the training time of 158, 793, and 191 minutes, respectively. GS had the worst performance in optimizing the HPO of RF (objective value of 300.31) but had the best performance in optimizing the HPO of GRUs (objective value of 156.64). Contrary, the training time of GS was the shortest (495 minutes) for RF's HPO but the longest for GRUs' HPO (1049 minutes).

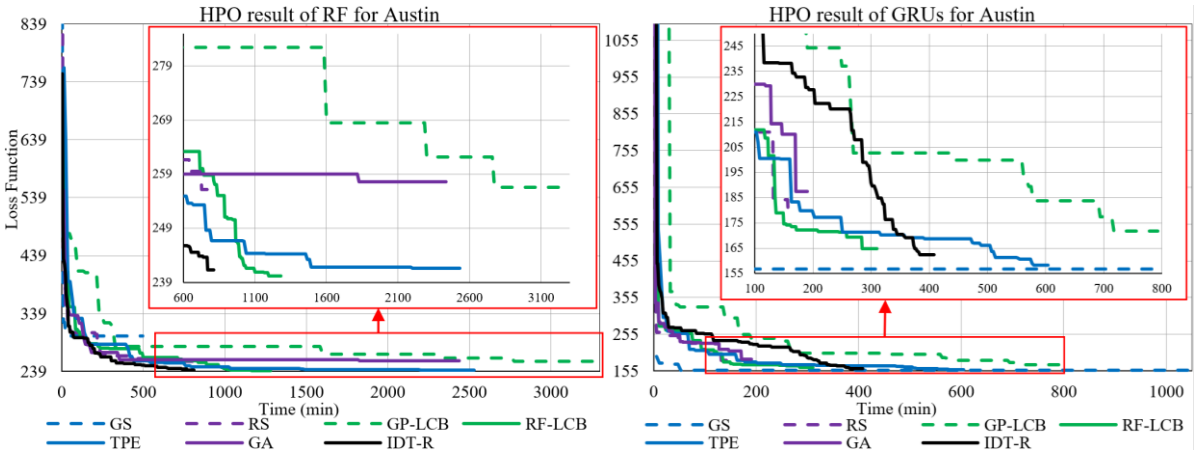

Figure 4.17 Numerical results of HPO of RF (*left*) and GRUs (*right*): Austin dataset

4.5 Discussion and sensitivity analysis

The overall findings demonstrate that each HPO method has its own benefits and drawbacks, which allows them to perform better than other algorithms in situations when computational resources are limited, such as training time or the number of examined points. As shown in **Figure 4.18**, TPE performs admirably when tuning deep learning architectures, particularly CNNs, but it performs badly when tuning nonconvex functions and the two machine learning models (SVM and RF). According to our numerical findings, RF-LCB typically performs worse than GP-LCB. In addition, the proposed algorithm (IDT-R) performs a little better than GP-LCB in most tasks. However, GA performs worse than the sequential-based HPO algorithms when there are fewer points to evaluate. As demonstrated in the problem of nonconvex functions, SVM, and RF, TPE converges more quickly than other methods but is

readily trapped in local optima. This suggests that while GP-LCB and IDT-R are probably superior if there are enough examined points, TPE performs better than other algorithms in limited numbers of searches. From the Pareto fronts in **Figure 4.18**, We can observe that IDT-R consistently achieves Pareto superiority, demonstrating its efficiency in terms of objective value and computing time.

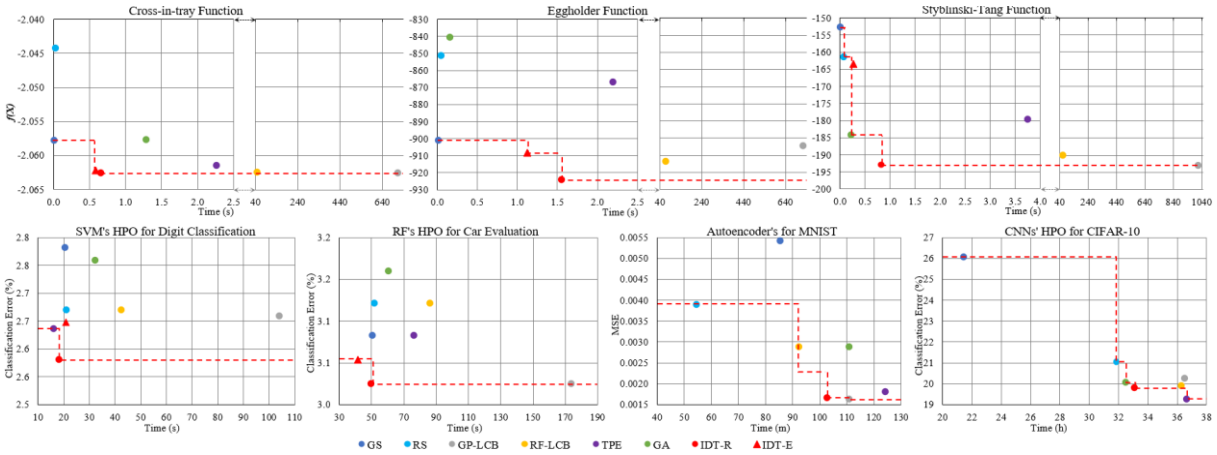

Figure 4.18 Pareto fronts of the performance of HPO algorithms in benchmark problems (Y-Objective value, X-Computational time)

Similarly, **Figure 4.19** shows the performance of HPO algorithms in optimizing the hyperparameters of RF and GRUs for shared e-scooter demand prediction. The proposed algorithm, IDT-R, is always on the Pareto fonts, except for the HPO of GRUs for the Thammasat dataset. RS shows consistent efficiency in computational time but limited performance, especially HPO of deep learning models. For demand prediction problems, GP-LCB offers poor performance in both optimal value and training time. On the other hand, RF-LCB tends to have a shorter training time compared to IDT-R, but IDT-R achieves a better objective value. TPE shows poor performance on the HPO of RF, specifically training time, but it performs better on the HPO of GRUs. This performance of TPE is also similar to the results from the benchmark problems above.

To better understand how IDT-R settings affect the objective value or how to balance the exploration and exploitation of IDT-R, sensitivity analysis was investigated. Due to time constraints, only three problems were subjected to sensitivity analysis: one deep learning model (Autoencoder), one machine learning algorithm (Random Forest), and one nonconvex function (Styblinski-Tang function). The three IDT-R parameters, comprising number of initial random points ($N \in [5, 150]$), number of best-performed leaves ($T \in [2, 10]$), and the number of random points in each leaf ($R \in [1, 7]$), were randomly chosen while each problem was trained for 40 trials. These three parameters for the Styblinski-Tang Function were randomly chosen from the range of up to 300, 15, and 10, respectively. Since the total number of searches was chosen as the stopping criterion, the approximate number of iterations can be calculated as $I = (200 - N)/(T \times R)$, where T is the product of the number of best-performed leaves, R is the number of random points per leaf, and N is the total number of searches.

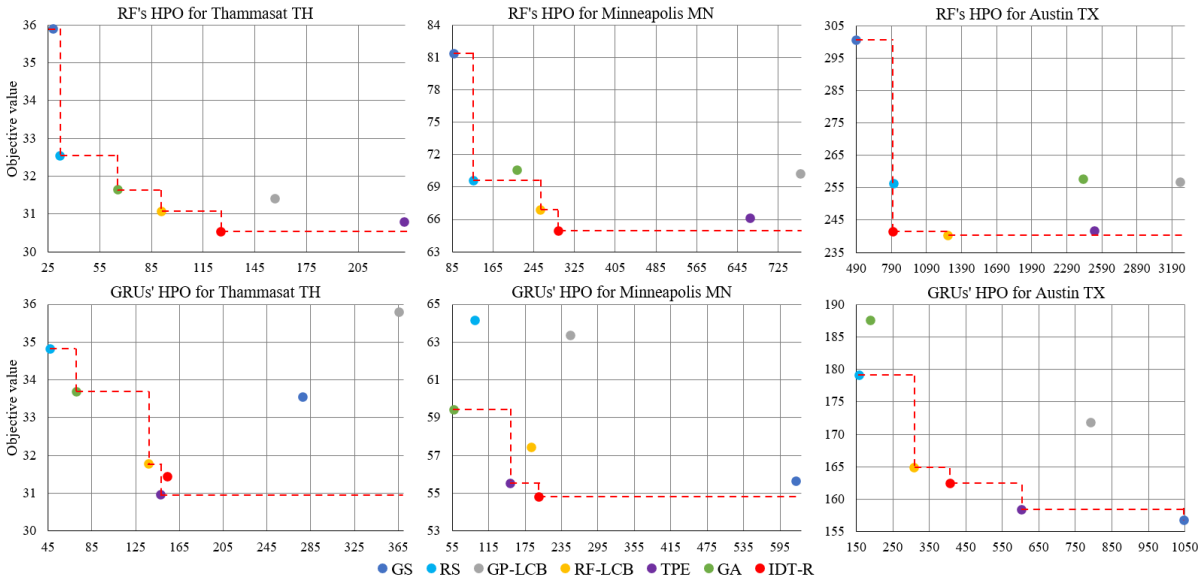

Figure 4.19 Pareto fronts of the performance of HPO algorithms for shared e-scooter demand prediction models, RF and GRUs (Y-Objective value, X-Computational time)

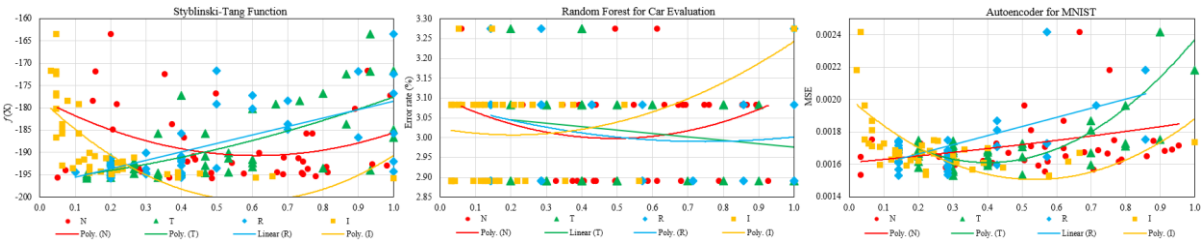

Figure 4.20 Sensitivity analysis for parameters (number of initial points (*N*), number of best-performed leaves (*T*), number of random points (*R*) in each leaf, and number of iterations (*I*)) of Iterative Decision Tree with Random (IDT-R) for the case of Styblinski-Tang function, Random Forest, and Autoencoder.

The scatter plot of the best objective value (y-axis) vs. the parameters' maximum values (x-axis) on a normalized scale is shown in **Figure 4.20**. The Styblinski-Tang function demonstrates that as *T* and *R* increase, the performance of IDT-R decreases since these two factors indicate the exploration. The relationship between the number of initial randoms (*N*) and the number of iterations (*I*) demonstrates that IDT-R performs better for greater values of *N* and *I*, but after a point, its performance degrades. Therefore, it is important to balance these four parameters in order to achieve good performance with enough exploitation (number of iterations). The number of best-performed leaves (*T*) and the number of random points per leaf (*R*), for instance, should be set at a lower value if the initial random points (*N*) are quite high. The pattern of each of the four IDT-R parameters in the context of Autoencoder's HPO closely resembles that of the Styblinski-Tang problem. On the other hand, RF's HPO shares characteristics with the Styblinski-Tang problem in terms of *N* and *I*, but performance is probably improved by using greater values of *T* and *R*.

In summary, it is possible that the patterns of the IDT-R parameters vary from one problem to another, but the patterns of initial random points (*N*) and iteration number (*I*) are usually very clear-cut. The trade-off between IDT-R exploration and exploitation should therefore be based

on three factors: the number of initial random points (*N*), the number of best-performed leaves (*T*), and the number of randoms in each leaf (*R*). This will allow for an adequate number of exploitation iterations, around 30 - 50. The number of best-performed leaves (*T*) can be chosen between [2, 4], while the number of randoms in each leaf (*R*) can range between [1, 3] for the total evaluation points of 200. The proportion of initials random points (*N*) in the overall number of searches could range from 20 to 50%.

4.6 Conclusion

This study evaluated the proposed HPO algorithms, Iterative Decision Tree (IDT) on various problems, including benchmark problems shared e-scooter demand prediction by RF and GRUs. This method used decision tree regression as a surrogate function, and in each iteration, a number of new candidates were assessed as the extreme points (IDT-E) or the random points (IDT-R) from several promising leaves. This property permits concurrent training of IDT, which reduces training time. In contrast to sequential model-based techniques that are already in use, IDT does not necessitate computing the acquisition function or exhaustively updating the surrogate function. Since IDT does not train the repetitive candidates, it can deal with the reproducibility problem of the existing algorithms and possibly terminates the optimizing procedure before reaching the Maximum Iteration criterion (i.e., shorter training time).

A benchmark set of cutting-edge algorithms were used, including Grid Search, Random Search, Gaussian Process with Lower Confidence Bound, Random Forest with Lower Confidence Bound, Tree-structured Parzen Estimator, and Genetic Algorithm to assess the effectiveness of the proposed framework. The proposed approach is more successful than other algorithms according to the numerical results (**Figure 4.18** and **Figure 4.19**), and it is on par with Bayesian Optimization or Tree-structured Parzen Estimator in terms of efficiency. Additionally, IDT-R performs better than IDT-E, which is inappropriate for optimizing problems with numerous hyperparameters (more than 3). Under a constrained set of evaluation points, IDT-R outperforms the benchmark algorithms in most optimization problems with acceptable computational time and result stability. For instance, TPE has good performance on HPO of deep learning models but has relatively poor performance on HPO of machine learning models and optimization of nonconvex functions.

Similarly, GP-LCB performs ineffectively in some problems, especially demand prediction problems, and requires longer training time. RF-LCB also provides quite good performance only on some problems. However, IDT-R performs well in most problems (i.e., model stability), while other algorithms can sometimes give slightly better objective values. But if we account for the training time as well, IDT-R performs efficiently as it mostly appears on the Pareto front line (12 out of 13 problems) compared to RF-LCB (8/13), TPE (5/13), and GP-LCB (4/13).

Even IDT-R performs effectively in searching for a near-global solution, its parameters need to be properly set to balance the exploration and exploitation. This trade-off can be done between the number of iterations and the other three parameters, such as number of initial points, number of best-performed leaves, and number of random points picked up from each leaf. For example, if the number of initial points is high, the number of best-performed leaves should be small and vice-versa, while the number of random points in each leaf should be just

one or two. This balance should be done in a way that there are enough iterations for exploitation.

References

[1] F. Hutter, L. Kotthoff and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges*. Cham, Switzerland: Springer Nature, 2019.

[2] R. Khalid and N. Javaid, "A survey on hyperparameters optimization algorithms of forecasting models in smart grid," *Sustainable Cities and Society,* vol. 61, pp. 102275, 2020, doi: https://doi.org/10.1016/j.scs.2020.102275.

[3] L. Li *et al.*, "A system for massively parallel hyperparameter tuning," in *Proceedings of Machine Learning and Systems 2*, 2020, pp. 230-246.

[4] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv:2003.05689,* Available: https://arxiv.org/abs/2003.05689

[5] D. Maclaurin, D. Duvenaud and R. Adams, "Gradient-based Hyperparameter Optimization through Reversible Learning," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015, vol. 37, pp. 2113-2122.

[6] A. Brock, T. Lim, J. M. Ritchie and N. Weston, "Smash: one-shot model architecture search through hypernetworks," *arXiv:1708.05344,* Available: https://arxiv.org/abs/1708.05344

[7] J. Lorraine and D. Duvenaud, "Stochastic hyperparameter optimization through hypernetworks," *arXiv:1802.09419,* Available: https://arxiv.org/abs/1802.09419

[8] G. Kunapuli, K. P. Bennett, J. Hu and J.-S. Pang, "Classification model selection via bilevel programming," *Optimization Methods and Software,* vol. 23, no. 4, pp. 475-489, 2008, doi: https://doi.org/10.1080/10556780802102586.

[9] C. Lin *et al.*, "Online hyper-parameter learning for auto-augmentation strategy," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6579-6588.

[10] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi and M. Pontil, "Bilevel Programming for Hyperparameter Optimization and Meta-Learning," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, vol. 80, pp. 1568-1577.

[11] Y. Guo, J.-Y. Li and Z.-H. Zhan, "Efficient Hyperparameter Optimization for Convolution Neural Networks in Deep Learning: A Distributed Particle Swarm Optimization Approach," *Cybernetics and Systems,* vol. 52, no. 1, pp. 36-57, 2020, doi: https://doi.org/10.1080/01969722.2020.1827797.

[12] C. Wang, Q. Wu, S. Huang and A. Saied, "Economic hyperparameter optimization with blended search strategy," in *International Conference on Learning Representations*, 2020.

[13] G. I. Diaz, A. Fokoue-Nkoutche, G. Nannicini and H. Samulowitz, "An effective algorithm for hyperparameter optimization of neural networks," *IBM Journal of Research and Development,* vol. 61, no. 4/5, pp. 9:1-9:11, 2017, doi: https://doi.org/10.1147/JRD.2017.2709578.

[14] J. Bergstra, R. Bardenet, Y. Bengio and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Advances in neural information processing systems*, 2011, vol. 24, pp. 2546–2554.

[15] R. Andonie, "Hyperparameter optimization in learning systems," *Journal of Membrane Computing,* vol. 1, no. 4, pp. 279-291, 2019, doi: 10.1007/s41965-019-00023-0.

[16] S. Kaur, H. Aggarwal and R. Rani, "Hyper-parameter optimization of deep learning model for prediction of Parkinson's disease," *Machine Vision and Applications,* vol. 31, no. 5, pp. 32, 2020, doi: https://doi.org/10.1007/s00138-020-01078-1.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research,* vol. 13, no. 2, 2012.

[18] A.-C. Florea and R. Andonie, "Weighted Random Search for Hyperparameter Optimization," *International Journal of Computers Communications & Control,* vol. 14, no. 2, pp. 154-169, 2019, doi: https://doi.org/10.15837/ijccc.2019.2.3514.

[19] Z. Karnin, T. Koren and O. Somekh, "Almost Optimal Exploration in Multi-Armed Bandits," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, vol. 28, no. 3, pp. 1238-1246.

[20] K. Jamieson and A. Talwalkar, "Non-stochastic Best Arm Identification and Hyperparameter Optimization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, vol. 51, pp. 240-248.

[21] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research,* vol. 18, no. 1, pp. 6765-6816, 2017.

[22] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei and S.-H. Deng, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," *Journal of Electronic Science and Technology,* vol. 17, no. 1, pp. 26-40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[23] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee and W. Rhee, "Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks," *IEEE Access,* vol. 8, pp. 52588-52608, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2981072.

[24] M. Parsa, J. P. Mitchell, C. D. Schuman, R. M. Patton, T. E. Potok and K. Roy, "Bayesian Multi-objective Hyperparameter Optimization for Accurate, Fast, and Efficient Neural Network Accelerator Design," *Frontiers in Neuroscience,* vol. 14, pp. 667, 2020, doi: https://doi.org/10.3389/fnins.2020.00667.

[25] F. Hutter, H. H. Hoos and K. Leyton-Brown, "Sequential Model-Based Optimization for General Algorithm Configuration," in *International conference on learning and intelligent optimization*, 2011, pp. 507-523.

[26] Y. Ozaki, M. Yano and M. Onishi, "Effective hyperparameter optimization using Nelder-Mead method in deep learning," *IPSJ Transactions on Computer Vision and Applications,* vol. 9, no. 1, p. 20, 2017, doi: https://doi.org/10.1186/s41074-017-0030-7.

[27] K. Swersky, J. Snoek and R. P. Adams, "Freeze-Thaw Bayesian Optimization," *arXiv:1406.3896,* Available: https://arxiv.org/abs/1406.3896

[28] Z. Dai, H. Yu, B. K. H. Low and P. Jaillet, "Bayesian Optimization Meets Bayesian Optimal Stopping," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, vol. 97, pp. 1496-1506.

[29] S. Falkner, A. Klein and F. Hutter, "BOHB: Robust and Efficient Hyperparameter Optimization at Scale," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, vol. 80, pp. 1437-1446.

[30] E. Contal, D. Buffoni, A. Robicquet and N. Vayatis, "Parallel Gaussian Process Optimization with Upper Confidence Bound and Pure Exploration," in *European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 225-240.

[31] J. Wang, S. C. Clark, E. Liu and P. I. Frazier, "Parallel Bayesian Global Optimization of Expensive Functions," *Operations Research,* vol. 68, no. 6, pp. 1850-1865, 2020, doi: https://doi.org/10.1287/opre.2019.1966.

[32] J. Wu and P. Frazier, "The Parallel Knowledge Gradient Method for Batch Bayesian Optimization," in *Advances in Neural Information Processing Systems*, 2016, vol. 29, pp. 3126-3134.

[33] S. R. Jino Ramson, K. Lova Raju, S. Vishnu and T. Anagnostopoulos, "Nature Inspired Optimization Techniques for Image Processing—A Short Review," in *Nature Inspired Optimization Techniques for Image Processing Applications* Cham, Switzerland: Springer, 2019, pp. 113-145.

[34] I. Loshchilov and F. Hutter, "CMA-ES for hyperparameter optimization of deep neural networks," *arXiv:1604.07269,* Available: https://arxiv.org/abs/1604.07269

[35] C. Guo, L. Li, Y. Hu and J. Yan, "A Deep Learning Based Fault Diagnosis Method With Hyperparameter Optimization by Using Parallel Computing," *IEEE Access,* vol. 8, pp. 131248-131256, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3009644.

[36] R. Jie, J. Gao, A. Vasnev and M. Tran, "HyperTube: A Framework for Population-Based Online Hyperparameter Optimization with Resource Constraints," *IEEE Access,* vol. 8, pp. 69038-69057, 2020, doi: https://doi.org/10.1109/ACCESS.2020.2986456.

[37] Y. Yoo, "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches," *Knowledge-Based Systems,* vol. 178, pp. 74-83, Aug. 2019 2019, doi: https://doi.org/10.1016/j.knosys.2019.04.019.

[38] D. Srivastava, Y. Singh and A. Sahoo, "Auto Tuning of RNN Hyper-parameters using Cuckoo Search Algorithm," in *2019 Twelfth International Conference on Contemporary Computing (IC3)*, 2019, pp. 1-5, doi: 10.1109/IC3.2019.8844900.

[39] B. Nakisa, M. N. Rastgoo, A. Rakotonirainy, F. Maire and V. Chandran, "Long Short Term Memory Hyperparameter Optimization for a Neural Network Based Emotion Recognition Framework," *IEEE Access,* vol. 6, pp. 49325-49338, 2018, doi: https://doi.org/10.1109/ACCESS.2018.2868361.

[40] C.-W. Tsai, C.-H. Hsia, S.-J. Yang, S.-J. Liu and Z.-Y. Fang, "Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing," *Applied Soft Computing,* vol. 88, p. 106068, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106068.

[41] W.-Y. Lee, S.-M. Park and K.-B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm," *Optik,* vol. 172, pp. 359-367, 2018, doi: https://doi.org/10.1016/j.ijleo.2018.07.044.

[42] B. Guo, J. Hu, W. Wu, Q. Peng and F. Wu, "The Tabu_Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms," *Electronics,* vol. 8, no. 5, pp. 579, 2019.

[43] V. Bibaeva, "Using Metaheuristics for Hyper-Parameter Optimization of Convolutional Neural Networks," in *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2018, pp. 1-6, doi: https://doi.org/10.1109/MLSP.2018.8516989.

[44] A. Martín, V. M. Vargas, P. A. Gutiérrez, D. Camacho and C. Hervás-Martínez, "Optimising Convolutional Neural Networks using a Hybrid Statistically-driven Coral Reef Optimisation algorithm," *Applied Soft Computing,* vol. 90, pp. 106144, 2020, doi: https://doi.org/10.1016/j.asoc.2020.106144.

[45] D. L. Marino, K. Amarasinghe and M. Manic, "Building energy load forecasting using deep neural networks," in *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 7046-7051: IEEE.

[46] J. Fan, Q. Li, J. Hou, X. Feng, H. Karimian and S. Lin, "A spatiotemporal prediction framework for air pollution based on deep RNN," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* vol. 4, pp. 15, 2017.

[47] P. Chen, H. Hsieh, X. K. Sigalingging, Y. Chen and J. Leu, "Prediction of Station Level Demand in a Bike Sharing System Using Recurrent Neural Networks," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, 2017, pp. 1-5, doi: 10.1109/VTCSpring.2017.8108575.

[48] C. Yin, Y. Zhu, J. Fei and X. He, "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access,* vol. 5, pp. 21954-21961, 2017, doi: https://doi.org/10.1109/ACCESS.2017.2762418.

[49] W. Kong, Z. Y. Dong, D. J. Hill, F. Luo and Y. Xu, "Short-term residential load forecasting based on resident behaviour learning," *IEEE Transactions on Power Systems,* vol. 33, no. 1, pp. 1087-1088, 2017.

[50] Q. Wu, K. Ding and B. Huang, "Approach for fault prognosis using recurrent neural network," *Journal of Intelligent Manufacturing,* vol. 31, no. 7, pp. 1621-1633, 2020, doi: 10.1007/s10845-018-1428-5.

[51] C. Xu, J. Ji and P. Liu, "The station-free sharing bike demand forecasting with a deep learning approach and large-scale datasets," *Transportation Research Part C: Emerging Technologies,* vol. 95, pp. 47-60, 2018, doi: https://doi.org/10.1016/j.trc.2018.07.013.

[52] B. Wang and I. Kim, "Short-term prediction for bike-sharing service using machine learning," *Transportation Research Procedia,* vol. 34, pp. 171-178, 2018, doi: https://doi.org/10.1016/j.trpro.2018.11.029.

[53] S. Kumar, L. Hussain, S. Banarjee and M. Reza, "Energy Load Forecasting using Deep Learning Approach-LSTM and GRU in Spark Cluster," in *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)*, 2018, pp. 1-4, doi: https://doi.org/10.1109/EAIT.2018.8470406.

[54] C.-J. Huang and P.-H. Kuo, "A deep cnn-lstm model for particulate matter (PM2. 5) forecasting in smart cities," *Sensors,* vol. 18, no. 7, pp. 2220, 2018.

[55] M. Nabil, M. Ismail, M. Mahmoud, M. Shahin, K. Qaraqe and E. Serpedin, "Deep Recurrent Electricity Theft Detection in AMI Networks with Random Tuning of Hyper-parameters," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 740-745, doi: 10.1109/ICPR.2018.8545748.

[56] N. F. Syed, M. Ge and Z. Baig, "Fog-cloud based intrusion detection system using Recurrent Neural Networks and feature selection for IoT networks," *Computer Networks,* vol. 225, pp. 109662, 2023, doi: https://doi.org/10.1016/j.comnet.2023.109662.

[57] X. Liu, A. Gherbi, W. Li and M. Cheriet, "Multi features and multi-time steps LSTM based methodology for bike sharing availability prediction," *Procedia Computer Science,* vol. 155, pp. 394-401, 2019.

[58] Y. Pan, R. C. Zheng, J. Zhang and X. Yao, "Predicting bike sharing demand using recurrent neural networks," *Procedia Computer Science,* vol. 147, pp. 562-566, 2019, doi: https://doi.org/10.1016/j.procs.2019.01.217.

[59] T. Le Quy, W. Nejdl, M. Spiliopoulou and E. Ntoutsi, "A Neighborhood-Augmented LSTM Model for Taxi-Passenger Demand Prediction," in *International Workshop on Multiple-Aspect Analysis of Semantic Trajectories*, 2019, pp. 100-116: Springer.

[60] A. H. Uddin, D. Bapery and A. S. M. Arif, "Depression Analysis of Bangla Social Media Data using Gated Recurrent Neural Network," in *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 2019, pp. 1-6: IEEE.

[61] M. Husein and I.-Y. Chung, "Day-ahead solar irradiance forecasting for microgrids using a long short-term memory recurrent neural network: A deep learning approach," *Energies,* vol. 12, no. 10, pp. 1856, 2019.

[62] G. Peter and M. Matskevichus, "Hyperparameters Tuning for Machine Learning Models for Time Series Forecasting," in *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 2019, pp. 328-332: IEEE.

[63] M. I. Khan and R. Maity, "Hybrid Deep Learning Approach for Multi-Step-Ahead Daily Rainfall Prediction Using GCM Simulations," *IEEE Access,* vol. 8, pp. 52774-52784, 2020.

[64] A. Moghar and M. Hamiche, "Stock Market Prediction Using LSTM Recurrent Neural Network," *Procedia Computer Science,* vol. 170, pp. 1168-1173, 2020, doi: https://doi.org/10.1016/j.procs.2020.03.049.

[65] A. Kisvari, Z. Lin and X. Liu, "Wind power forecasting – A data-driven method along with gated recurrent neural network," *Renewable Energy,* vol. 163, pp. 1895-1909, 2021, doi: https://doi.org/10.1016/j.renene.2020.10.119.

[66] X. Wei, L. Zhang, H.-Q. Yang, L. Zhang and Y.-P. Yao, "Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks," *Geoscience Frontiers,* vol. 12, no. 1, pp. 453-467, 2021, doi: https://doi.org/10.1016/j.gsf.2020.04.011.

[67] H. L. Vu, K. T. W. Ng, A. Richter and C. An, "Analysis of input set characteristics and variances on k-fold cross validation for a Recurrent Neural Network model on waste disposal rate estimation," *Journal of Environmental Management,* vol. 311, pp. 114869, 2022, doi: https://doi.org/10.1016/j.jenvman.2022.114869.

[68] H. L. Vu, K. T. W. Ng, A. Richter and G. Kabir, "The use of a recurrent neural network model with separated time-series and lagged daily inputs for waste disposal rates modeling during COVID-19," *Sustainable Cities and Society,* vol. 75, pp. 103339, 2021, doi: https://doi.org/10.1016/j.scs.2021.103339.

[69] J. Siłka, M. Wieczorek and M. Woźniak, "Recurrent neural network model for high-speed train vibration prediction from time series," *Neural Computing and Applications,* vol. 34, no. 16, pp. 13305-13318, 2022, doi: 10.1007/s00521-022-06949-4.

[70] A. Zanfei, B. M. Brentan, A. Menapace, M. Righetti and M. Herrera, "Graph Convolutional Recurrent Neural Networks for Water Demand Forecasting," *Water Resources Research,* vol. 58, no. 7, pp. e2022WR032299, 2022, doi: https://doi.org/10.1029/2022WR032299.

[71] H. Song and H. Choi, "Forecasting Stock Market Indices Using the Recurrent Neural Network Based Hybrid Models: CNN-LSTM, GRU-CNN, and Ensemble Models," *Applied Sciences,* vol. 13, no. 7, pp. 4644, 2023.

[72] E. Yahyaoui, "A Study of Hyperparameter Optimization Algorithms Applied to LSTM in Financial Time Series Forecasting," Master, EWHA Womans University, 2019.

[73] W. Kong *et al.*, "Effect of automatic hyperparameter tuning for residential load forecasting via deep learning," in *2017 Australasian Universities Power Engineering Conference (AUPEC)*, 2017, pp. 1-6: IEEE.

[74] A. Mashlakov, V. Tikka, L. Lensu, A. Romanenko and S. Honkapuro, "Hyper-parameter optimization of multi-attention recurrent neural network for battery state-of-charge forecasting," in *EPIA Conference on Artificial Intelligence*, 2019, pp. 482-494: Springer.

[75] N. A. Rashid, I. A. Aziz and M. H. B. Hasan, "Machine Failure Prediction Technique Using Recurrent Neural Network Long Short-Term Memory-Particle Swarm Optimization Algorithm," in *Computer Science On-line Conference*, 2019, pp. 243-252: Springer.

[76] S. Bouktif, A. Fiaz, A. Ouni and M. A. Serhani, "Multi-sequence LSTM-RNN deep learning and metaheuristics for electric load forecasting," *Energies,* vol. 13, no. 2, pp. 391, 2020.

[77] X. Zhang, X. Chen, L. Yao, C. Ge and M. Dong, "Deep neural network hyperparameter optimization with orthogonal array tuning," in *International Conference on Neural Information Processing*, 2019, pp. 287-295: Springer.

[78] J. Friedman, T. Hastie and R. Tibshirani, *The Elements of Statistical Learning* (no. 10). Springer series in statistics, New York, 2001.

[79] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[80] H. Tim, K. Manoj, N. Holger, L. Gilles and S. Iaroslav, "Scikit-Optimize," Available: https://scikit-optimize.github.io/

[81] T. Akiba, S. Sano, T. Yanase, T. Ohta and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631, doi: https://doi.org/10.1145/3292500.3330701.

[82] S. R. Mohammad, "geneticalgorithm," Available: https://github.com/rmsolgi/geneticalgorithm

[83] D. Dua and C. Graff, "UCI Machine Learning Repository," Available: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

[84] L. Yann, C. Corinna and C. J. C. Burges, "THE MNIST DATABASE of handwritten digits," Available: http://yann.lecun.com/exdb/mnist/

[85] K. Alex, "The CIFAR-10 dataset," Available: https://www.cs.toronto.edu/~kriz/index.html

[86] J. Brownlee, *Deep learning for computer vision: image classification, object detection, and face recognition in Python*. Machine Learning Mastery, 2019.

CHAPTER 5
5. REBALANCING SHARD E-SCOOTERS UNDER DEMAND UNCERTAINTY

5.1 Introduction

The first-and-last-mile is a prevalent transportation challenge witnessed across numerous urban regions worldwide, stemming from inadequate public transit planning, financial limitations, and the expansion of urban areas. The private sector has intervened to bridge this transportation gap by introducing collaborative transportation modes, encompassing shared bikes, shared electric bikes (e-bikes), and shared electric scooters (e-scooters), which are operated through either dock-based or dockless systems. Dock-based systems are predominantly utilized for shared bicycles, while users start and end their trips by bikes at designated stations. As the station's capacity is determined by the number of docks, users of shared dock-based bicycles may encounter situations wherein they are unable to end their journeys at the most convenient station due to a lack of capacity. Consequently, they are compelled to terminate their trips at a less convenient station that has the requisite capacity. In contrast, individuals utilizing dockless shared bicycles or e-scooters have the flexibility to pick up or appropriately park such vehicles within any public space within a designated operational zone. Nevertheless, despite the convenience of this dockless shared mobility approach, these dockless bicycles or e-scooters at times impede public access (for example, by obstructing sidewalks), exert adverse impacts on urban aesthetics, or fall prey to acts of vandalism. To address these challenges, operators must promptly clear excessive vehicles and establish proper parking zones.

Shared bicycles were initially introduced in Amsterdam in 1965 [1], whereas shared electric scooters (e-scooters) made their debut in Singapore in 2016 and later in the United States in 2017 [2]. However, a notable development occurred in the United States in 2019, where the cumulative ridership of shared e-scooter trips (amounting to 96 million trips) exceeded the combined ridership of both dockless and dock-based shared bicycles (totaling 40 million trips). This shift can be attributed to the widespread availability of e-scooter services in over 100 U.S. cities [3]. This phenomenon of shared micromobility has been extensively investigated in the scholarly literature [2, 4, 5]. Furthermore, the topic of shared e-scooters has attracted significant scholarly interest across various dimensions, including policy and regulatory aspects [6-12], spatiotemporal patterns of trips [4, 13-18], life cycle assessments [19-22], and societal perceptions [23-26].

In the domain of short-term demand prediction, an innovative neural network structure known as GCScoot, categorized as a spatiotemporal graph capsule neural network, has been developed to anticipate the movement patterns of shared e-scooters by taking into account the adjustments in deployment configurations [27, 28]. The evaluation of GCScoot against baseline models was conducted using openly available datasets from four U.S. cities: Austin, Texas (TX); Louisville, Kentucky; Minneapolis, Minnesota (MN); and Chicago, Illinois, and it displayed superior performance, establishing a new benchmark. In the realm of hourly demand forecasting for shared e-scooters, distinct methodologies have been explored. For instance, the seasonal autoregressive integrated moving average (SARIMA) model was employed to predict the hourly e-scooter demand at Thammasat University (Thailand), while the variance in this demand was addressed using the generalized autoregressive conditional heteroskedasticity (GARCH) model [29]. In a separate study by Ham et al. [30], an encoder-recurrent neural network-decoder

framework was utilized to predict latent temporal characteristics within a convolutional autoencoder. This approach was applied to satisfied and unmet e-scooter demands in the Gwangjin district of Seoul, South Korea. To address the challenge of demand scarcity, a masked fully convolutional network, guided by a mask model or a region of interest, was devised. This network was designed to concentrate solely on the active cells within Calgary, Canada [31]. Similarly, Xu et al. [32] introduced a spatiotemporal multi-graph transformer, leveraging various graph types such as adjacency, functional similarity, demographic similarity, and transportation supply similarity graphs. This novel approach based on graph convolutional networks was used for predicting hourly shared e-scooter demand in Austin, TX, and Washington, District of Columbia (DC). Further studies explored different predictive techniques. For instance, a long short-term memory (LSTM) model was employed to forecast hourly shared e-scooter demand in Seoul, South Korea, specifically in the Seocho and Gangnam districts [33]. Recently, Khan et al. [34] proposed an ensemble model based on extreme gradient boosting (XGBoost), extra trees, and random forests, employed to predict the clustered daily demand of shared e-scooters using the *k*-means algorithm on Jeju Island, South Korea.

On the other hand, there exists a limited body of research pertaining to the operational strategies of shared e-scooters, encompassing facets such as e-scooter recharging, fleet size determination, e-scooter distribution and rebalancing strategies, and facility location planning. Masoud et al. [35] addressed the challenge of recharging shared e-scooters by adapting a College Admission algorithm to solve an Integer Linear Programming (ILP) formulation. This approach aimed to ascertain the optimal allocation of freelance chargers required for the task. Similarly, Ciociola et al. [36] leveraged Poisson processes to investigate the interaction between fleet size, battery charging, and simulated demand patterns for shared e-scooters. The utilization of deep learning models also came into play. The 3D-CLoST model was introduced to predict shared e-scooter demand, subsequently integrating a pragmatic relocating strategy executed by workers using a greedy approach [37]. Osorio et al. [38] introduced a mixed-integer program that accounted for the possibility of e-scooter charging on rebalancing vehicles during overnight periods. To address scalability concerns, a discrete-continuous hybrid model was developed, combining line haul and local operational considerations. This hybrid model was assessed through randomly generated demand scenarios based on normal distributions. Fathabad et al. [39] formulated a two-stage stochastic program for the short- and long-term operational planning of shared e-scooters. The primary stage aimed to minimize investment costs linked to charging infrastructure, e-scooter fleet sizing, and relocation schedules, while the secondary stage focused on optimizing short-term operational expenses encompassing relocation, charging, and unserved demand penalties. Losapio et al. [40] devised a novel approach termed "E-Scooter Balancing-Deep Q Network," which employed multi-agent deep reinforcement learning. This approach aimed to minimize the need for rebalancing operations and battery swapping, encouraging customers to retrieve e-scooters from nearby zones to mitigate imbalances. Another recent study, a multi-criteria decision protocol underpinned by geographical information systems was proposed by Altintasi and Yalcinkaya [41] to optimize the placement of charging stations for shared e-scooters. The objective was to seamlessly integrate the shared e-scooter system with existing public amenities, points of interest, and population densities, demonstrating a comprehensive approach.

Shared electric scooters are predominantly employed for short journeys, typically spanning around 1.5 kilometers in distance and lasting about 10 minutes. These rides are particularly popular for tourist activities and recreational purposes [5, 29, 31, 42]. It's noteworthy that the ridership patterns of shared e-scooters are more variable compared to shared bikes, which are primarily utilized for daily commuting. Shared bike ridership exhibits two peaks in demand: one during the morning rush hour and another during the evening rush hour. In contrast, shared e-scooter ridership remains consistently high from morning until late evening. Within a dock-based system, the pickup and drop-off demands conform to the constraints of the available docks or station capacity. However, in a dockless system, demand is not bound by such limitations, leading to greater demand fluctuations and increased demand volatility. Due to the nature of shared e-scooter ridership and the absence of docking stations, shared e-scooters require more frequent rebalancing efforts to meet their dynamically changing demand. Unlike shared bikes, which might suffice with two rebalancing operations during peak demand periods (morning and evening peaks), shared e-scooters necessitate a larger number of rebalancing actions and a shorter planning horizon. Furthermore, shared e-scooters require intensive maintenance [16] and possess a relatively short operational lifespan [20]. This is due to their design, focused on being lightweight and user-friendly, requiring battery replacement or recharging as their battery levels deplete. In contrast to the previous scenario, the latter situation aligns more seamlessly with the rebalancing process and was, therefore, the central focus of the present study. In this context, e-scooters with low battery levels are relocated to nearby charging facilities, with a particular emphasis on charging stations powered by renewable energy sources such as solar power [43].

Limited prior research has delved into the aspect of short-term operational planning for shared e-scooters, particularly focusing on daily planning involving hourly intervals, among others. In this context, the queuing model employing a Poisson distribution to model demand has found common usage in bike sharing applications. However, this approach is not without drawbacks. Firstly, the queuing model introduces higher demand uncertainty than demand prediction techniques, leading to escalated operational costs (as depicted in **Figure 5.3**). Secondly, actual demand patterns in shared micromobility are considerably erratic and influenced by various external factors, thereby diverging from the assumptions of a Poisson distribution (as detailed in **Section 5.4.1**). Conversely, several studies have harnessed machine learning or deep learning models to predict short-term demand for the purposes of rebalancing operations. Nevertheless, the deployment of e-scooters solely based on predictions from these models yields an undesirable service level due to the absence of consideration for prediction errors. Against this backdrop, this study introduces an original data-driven framework tailored for short-term rebalancing strategies for shared e-scooters, with an explicit incorporation of e-scooter and trip attributes. The primary objective of this framework revolves around reallocating the constrained e-scooters to locations projected to encounter the highest anticipated demand. Addressing the uncertainty inherent in shared e-scooter demand primarily involves demand prediction, while the residual uncertainty is managed by the SGARCH model. In this context, "Allocation" refers to the temporal correlation between forecasted variance via SGARCH, current trends in demand volatility, and the efficacy of demand prediction models. This stands in contrast to the conventional approaches involving constant or periodic (daily or weekly) variance assumptions. Our proposed framework aligns adeptly with the domain of static rebalancing planning, catering to planning horizons spanning from a few hours to several hours. Nevertheless, the scope of the

outlined framework can be extended to encompass multiple planning horizons in forthcoming research endeavors. The noteworthy contributions of this study are enumerated as follows:

- Monte Carlo simulation was utilized to model the uncertainty in shared e-scooter demand, taking into account the trip gap projections derived from GB regression and the variance as well as probability distribution forecasted through the seasonal generalized autoregressive conditional heteroskedasticity (SGARCH).
- The static vehicle-based rebalancing planning was expressed as an Integer Linear Programming (ILP) challenge, aiming to tackle demand uncertainty, relocation of faulty e-scooters to the central depot for repairing, and collection of e-scooters with low battery levels to neighboring charging stations. Two distinct ILP formulations were devised, one with predetermined route sequences and the other with undisclosed sequences. To enhance practical feasibility, modifications were introduced to the objective function and operational constraints. This entailed incorporating penalties for specific unfulfilled demands, as opposed to deviations in requests found in previous studies.
- The task of rebalancing optimization was addressed using both an Integer Linear Programming solver (GLPK) and a novel hybrid algorithm, namely the ant colony optimization–ILP (ACO-ILP) approach. These methods were applied to solve rebalancing optimizations posed by demand scenarios simulated through the Monte Carlo technique. For empirical validation, a real-world dataset encompassing dockless shared e-scooter operations in Minneapolis (MN) was selected as the focal point of the case study.

5.2 Literature review

As outlined in the preceding section, the scope of research concerning the operational strategies of shared e-scooters remains constrained. However, the knowledge base can be enriched by drawing insights from analogous sharing services, particularly shared bikes, which share a degree of resemblance with shared e-scooters. Shui and Szeto [44] conducted a comprehensive survey of existing studies, revealing that these inquiries have centered around diverse facets of shared-bike rebalancing. These aspects encompass a variety of elements such as objective functions (e.g., distance, cost, and emissions optimization), constraints (e.g., budget, service duration, and inventory considerations), optimization algorithms (ranging from precise to heuristic methodologies), deterministic or stochastic problem formulations, and scenarios involving either static or dynamic considerations. In alignment with this context, the present study is dedicated to addressing shared bike rebalancing challenges under the presence of demand uncertainty and predicted demand scenarios.

Shared bicycles find predominant utilization in commuting, thus exhibiting pronounced peak-demand periods during morning hours (6 am–10 am) and evening hours (4 pm–8 pm) [45, 46]. This characteristic, in tandem with the station capacities, results in a more stable demand pattern for shared bikes in comparison to dockless shared e-scooters. Consequently, conventional investigations have frequently operated under the assumption that bike-sharing demand conforms to a Poisson distribution. For example, there has been a bias towards representing dynamic shared-bike inventory levels using continuous-time Markov chains (CTMCs), incorporating Poisson processes for pickups and drop-offs. This portrayal translates into a double-ended queuing system model. Empirical validation of this modeling paradigm

was carried out utilizing a real-world dataset from Tel Aviv, Israel's bike-sharing system [47]. Likewise, the simulation approach by Monte Carlo and the approximation approach with Skellam distribution derived from historical ridership of shared bikes were utilized to quantify the unserved demand in reference [48]. Meanwhile, the CTMC framework was adapted for addressing overnight rebalancing operations within New York City's Citi Bike system. Both small- and large-scale problems were addressed through the employment of an ILP solver and a greedy algorithm, respectively [49]. Several bike sharing datasets, from Boston, Massachusetts, and Washington, were utilized as case studies evaluating the effectiveness of a non-stationary queuing (Mt/Mt/1/K) model characterized by exponentially distributed pickups and drop-offs [50]. In the pursuit of dynamic rebalancing, Seo [51] incorporated demand uncertainty by adopting a Markov decision process reliant on a Poisson distribution, wherein the mean demand was predicted through random forest regression. This approach was evaluated through a case study centered around bike sharing in Seoul, South Korea. Nonetheless, a chi-squared goodness-of-fit analysis unveiled that only 77% of the stations encompassed in the study exhibited demand patterns adhering to a Poisson distribution. Meanwhile, Lu [52] utilized the bike sharing dataset operated in New Taipei City, Taiwan, to examine the proposed robust multi-period bike fleet allocation scheme preventing the worst-case scenario (i.e., maximum demand).

Other academic literatures have addressed the challenge of demand uncertainty within rebalancing contexts by integrating sample average approximation through Monte Carlo sampling. For instance, the operational management of shared autonomous electric vehicles in Shanghai, China, harnessed the Monte Carlo method to simulate daily demand, employing a normal distribution derived from historical ridership [53]. In the domain of bike sharing, Monte Carlo sampling emerged as a pivotal tool. It was leveraged to generate demand scenarios for Bergamo, Italy, employing four distinct probability distributions (uniform, exponential, normal, and log-normal) utilizing mean and standard deviation values drawn from historical ridership [54]. Likewise, for New Taipei City, Taiwan, historical weekly ridership was employed to simulate demand scenarios using a truncated normal distribution [55]. Contrary, Dell'Amico et al. [56] employed historical daily ridership as individual scenarios and proceeded to resolve stochastic programming models by a few methodologies, including branch-and-cut, deterministic equivalent programs, L-shaped procedures, and heuristic algorithms. These endeavors were executed utilizing a couple of open datasets.

Conversely, machine learning and deep learning models have demonstrated commendable predictive capabilities, warranting their integration within rebalancing frameworks. For instance, Regue and Recker [57] introduced a chance-constrained programming approach for dynamic rebalancing of shared bikes, leveraging a normal distribution with the predicted demand from GB regression and the variance from prediction model's residuals. In a distinct application, RF was employed to predict station-level rental and return demands for bike sharing in Nanjing, China. Subsequently, static rebalancing was formulated to fulfill the predicted demand, conceptualizing the problem as a hub-first–route-second problem [58]. The truck-based rebalancing in New York City was constructed with the integration of a spatiotemporal graph neural network tailored to forecast city-wide bike demand [59]. Similarly, the deterministic dynamic rebalancing approach for shared bikes in Beijing, China, was framed within a data-driven framework fortified by a deep learning model [45]. Seoul, South Korea's

bike sharing operations were also subject to prediction-enhanced strategies by RF, adopted to forecast forthcoming demand and inventory levels for the purpose of repositioning [60]. Presenting a fusion of methodologies, Yu et al. [61] devised the SARIMA-LSTM hybrid model. This hybrid prediction model facilitated the prediction of pickup and drop-off demands, thereby underpinning the rebalancing planning for bike sharing centered around rail transit stations within Xicheng, Beijing.

To synthesize, the strategies employed for short-term operational planning within the domain of bike sharing have predominantly accommodated demand uncertainty via Markov chain models, often assuming a particular probability distribution, commonly the Poisson distribution, to model demand behavior. Alternatively, operational rebalancing endeavors have embraced the utilization of machine learning and deep learning models to predict demand. While Regue and Recker [57] incorporated the error stemming from demand prediction models into their bike-sharing rebalancing scheme, they did not comprehensively analyze the variance and underlying probability distribution. Hence, this current study employs the SGARCH model to systematically scrutinize the residuals generated by the demand prediction model. This regression-based methodology serves to diminish the average demand uncertainty and concurrently furnishes vital parameters, including probability distribution and temporal variance—essential components for the subsequent Monte Carlo sampling process. Past research has addressed the collection of malfunctioning bikes within their rebalancing problems [45]. However, the topic of recharging, particularly concerning electric bikes (e-bikes), remains largely unexplored. This disparity in attention might be attributed to the fact that despite their similar cost profile to shared e-scooters, e-bikes are frequently introduced at lower fleets in conjunction with traditional bicycles. Nonetheless, e-bikes are generally considered to be of lesser allure. Consequently, this study strives to encompass all three e-scooter types: usable, faulty, and low-battery e-scooters. Moreover, it endeavors to frame the rebalancing problem against the backdrop of stochastic demand by adopting the Sample Average Approximation (SAA) methodology. This approach offers the advantage of tailoring parameter settings to attain a target service level. Notably, this strategy facilitates the formulation of the rebalancing challenge as an Integer Linear Programming (ILP) problem, accommodating both known and unknown route sequences, and thereby allowing the rebalancing optimization to be solvable exact or heuristic algorithms. Furthermore, the proposed ACO-ILP algorithm boasts compatibility with parallel computing, rendering it amenable to scalability and practical implementation.

5.3 Methodology

5.3.1 Research framework

The standard protocol for dockless shared e-scooter journeys encompasses multiple stages: identifying nearby e-scooters through visual observation or employing a mobile application, proceeding to the designated e-scooter location on foot, activating the e-scooter lock mechanism via a mobile application, utilizing the e-scooter to travel to the desired destination, appropriately securing the e-scooter upon arrival, and ultimately ending the trip. While the termination process for dockless mode trips is undeniably convenient, customers may opt against initiating their e-scooter journey if the distance they must traverse to retrieve the e-

scooter is deemed excessively lengthy. To mitigate this concern, operators frequently divide their operational zones into walkable regions (e.g., within a range of 200 to 500 meters) and ensure that e-scooters are strategically positioned within these designated zones. A common strategy employed by operators involves rebalancing or relocating e-scooters from zones characterized by an excessive number of such e-scooters to zones where the number of scooters is comparatively lower. Operators frequently evaluate the condition of each zone and engage in rebalancing efforts to avoid instances of unsatisfied demand or zones experiencing scarcity, particularly due to operational limitations such as the restricted availability of e-scooters. This is done within the designated timeframe. To optimize operational efficiency, operators can leverage pertinent information and historical trip data to predict future demands—quantities of pickup and drop-off demands anticipated within specific time intervals for each zone. This predictive insight subsequently forms the basis for refining the rebalancing strategy. This involves determining the most optimal route path for the rebalancing vehicle and ascertaining the appropriate number of e-scooters to be collected from or redistributed to each respective zone.

Rebalancing operations can be classified into two primary categories: Static and Dynamic. In the context of static rebalancing, the rebalancing operational strategy, often revolving around vehicle-based rebalancing, is predicated upon several presumptions. These include scenarios wherein no e-scooter usage takes place during the rebalancing procedure, or the dynamic interaction between customers and the systems can be deemed negligible. Moreover, static rebalancing might involve predetermined time intervals and an extended planning horizon. It also assumes negligible influence from user activities, behaviors, or interventions, and presupposes that the number of e-scooters at each station remains constant throughout the rebalancing process. Contrastingly, dynamic rebalancing acknowledges the impact of users' usage patterns while rebalancing is underway. This category also accounts for short planning intervals, potentially extending to real-time planning, and leverages users' actions and behaviors as crucial inputs. Dynamic rebalancing takes into consideration the continuous alterations within the systems, particularly in instances where the systems are operational or in use. In this scenario, the optimal time for implementing static rebalancing is during nighttime when the system is inactive or experiencing minimal usage. If the planning horizon is relatively long and significant information such as requests, distribution of e-scooters, weather, etc. remains unchanged, rebalancing during daytime might be categorized as static rebalancing. In alignment with the foundational assumptions, the proposed framework for periodic rebalancing of shared e-scooters, as depicted in **Figure 4.1**, aligns itself more closely with the static rebalancing variant. This categorization is justified by the shared assumptions outlined in **Section 5.3.4**. However, it's noteworthy that the framework could be classified as dynamic rebalancing if the planning horizon is as short as an hour or less. Nevertheless, hinging upon the primary assumptions, our rebalancing framework is best classified within the static category, as illustrated in **Figure 4.1**.

Figure 4.1 illustrates the schematic structure of the research framework, comprising three key segments: (1) the process of collecting and manipulating data, (2) the prediction of trip gap utilizing GB regression and variance prediction employing SGARCH, and (3) optimization of the rebalancing process. Given that shared e-scooters serve as a form of transportation for short distances, their immediate demand is influenced by various external factors, encompassing

weather conditions, seasonal patterns (both on a weekly and yearly basis), holidays, and special events [4]. These influential factors were thus incorporated into the demand prediction models, subject to manipulation. E-scooters represent a form of shared transportation that operates without the need for designated docking stations. This allows users the flexibility to both retrieve and return e-scooters at any location within a specified operational region, with the exception of private properties or areas explicitly prohibited by governing bodies. The common practice of grid-based spatial aggregation is often employed for handling spatial trips. However, this approach falls short in encapsulating demand concentration or the congruity of trip intents. For instance, spatial trips occurring within locales like shopping malls, parks, or schools might be dispersed across couple grid cells if these sites are situated along the grid's periphery. Furthermore, past research has explored aggregation methods based on postal codes or administrative divisions like communities or wards. Nonetheless, these territorial units often prove overly expansive for addressing the exigencies of short-term rebalancing operation to shared e-scooters. Consequently, the present investigation leveraged a *k*-means clustering algorithm to aggregate the spatial ridership patterns of shared e-scooters. This entailed employing cluster range of 15, 30, and 60 to effectively categorize and consolidate spatial ridership trends.

In order to address the inherent scarcity in trip flow data, this study undertook the prediction of trip gap, denoting the net demand disparity between origin trips (referred to as trip generation or pickup demand) and destination trips (referred to as trip arrival or drop-off demand). In instances where the trip gap is positive, it signifies a prevalence of pickup trips over drop-off trips. The hyperparameters of the gradient boosting regressor for predicting trip gap were adjusted using an automated algorithm, namely Bayesian optimization (BO). Subsequently, the SGARCH model was trained using the residuals extracted from the trip gap prediction. Then, in light of the projected discrepancy and variability in trip demand, a Monte Carlo sampling technique was employed to model the unpredictable nature of shared e-scooter demand for the purpose of optimizing rebalancing operations.

The problem of rebalancing optimization is classified as NP-hard, indicating that the computational time required to solve it grows exponentially as the number of clusters rises. As a result, the presence of computational limitations posed challenges for exact algorithms in generating either the globally optimal solutions or even the viable solutions. In order to address this limitation, we investigated viable heuristic techniques in conjunction with the precise Integer Linear Programming (ILP) solver "GLPK". The devised heuristic solutions involved breaking down the overarching rebalancing issues into two distinct components: Routing Problems and Pickup or Drop-off Problems. A suite of population-based algorithms, namely Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), were subjected to scrutiny to optimize the routing problem. In parallel, the ILP solver and GA were enlisted to streamline the pickup and drop-off operations. Analyzing our numerical findings, as encapsulated in **Table 5.1**, reveals that the combination of heuristic algorithms for routing (GA, PSO, & ACO) and pickup/drop-off operations (GA) exhibited suboptimal efficacy. In fact, this hybrid scheme often struggled to locate feasible solutions, especially when grappling with expansive problem dimensions. This ineffectiveness was most noticeable with GA's inability to yield a feasible pickup and drop-off outcome, primarily due to violations of drop-off constraints. The occurrence of drop-off violations on small issue sizes

is initially minimal, but it increases exponentially as the number of nodes increases. Conversely, when routing optimization was governed by a blend of heuristic algorithms (GA, PSO, and ACO) and pick-up and drop-off operations were guided by the ILP solver, the ACO-ILP hybrid outperformed its counterparts—GA-ILP and PSO-ILP—particularly in the context of larger problem dimensions. Notably, ACO demonstrated a relatively longer computing time for generating feasible route sequences compared to GA and PSO, albeit with a faster convergence rate. In cases involving simple Traveling Salesman Problems, where the objective value (i.e., travel distance) entails computationally efficient evaluation, GA and PSO—with shorter computing times—are capable of assessing a larger pool of candidates within a confined time frame. Nevertheless, the computational cost associated with the objective value, namely the pickup and drop-off operation optimized by the ILP solver, is relatively high. Consequently, the total number of candidates (or route sequences) that can be assessed is constrained. In such scenarios, ACO, with its swift convergence, stands out for attaining superior optimal objective outcomes.

Table 5.1 Performance of several algorithms for Rebalancing Optimization

Number of Nodes	Optimization Algorithms		Time (min)	Routing	Penalty	Objective Value
	Routing Optimization	**Pickup & Drop-off Optimization**				
15 Stations	Integer Linear Programming (ILP) Solver		**30.0**	**35.9**	**41.0**	**76.9**
	Genetic Algorithm (GA)	ILP-Solver	31.3	40.2	41.0	81.2
	Particle Swarm Optimization (PSO)		31.8	41.0	41.4	82.4
	Ant Colony Optimization (ACO)		**33.0**	**38.1**	**41.0**	**79.1**
	GA	GA	43.4	59.7	42.6	102.3
	PSO		40.0	62.1	43.4	105.5
	ACO		**43.9**	**43.4**	**44.0**	**87.4**
30 Stations	ILP-Solver		**45.0**	**95.8**	**251.0**	**346.8**
	GA	ILP-Solver	44.2	142.1	249.5	391.6
	PSO		46.4	158.8	233.5	392.3
	ACO		**42.6**	**73.3**	**293.0**	**366.3**
	GA	GA	57.3	208.3	394.0	602.3
	PSO		54.4	199.3	401.5	600.8
	ACO		**40.2**	**100.2**	**494.5**	**594.7**
60 Stations	ILP-Solver		**60.0**	**279.4**	**2056.5**	**2335.9**
	GA	ILP-Solver	59.5	513.4	1815.7	2329.1
	PSO		58.7	491.1	1841.1	2332.2
	ACO		**65.6**	**163.7**	**1908.9**	**2072.6**
	GA	GA	Not Converge since GA can't find a feasible solution for pickup and drop-off problem.			
	PSO					
	ACO					

Consequently, the optimization problems outlined in the **Figure 5.1** framework were tackled using only an ILP solver and the hybrid approach merging ACO with ILP solver, ACO-ILP. Traditionally, studies have often inferred the stochastic characteristics of shared-bike demand from historical records, where each scenario has manifested as a seasonal snapshot, or a sample generated through Monte Carlo simulation [54-56]. In order to mitigate any unwarranted assumptions regarding the distribution of demand, which would not be practicable when considering the demand for shared e-scooters, historical data pertaining to daily and weekly patterns were chosen as the benchmarks for rebalancing planning. In the last step, a comparison was made between the objective values of the rebalancing optimization problems, including the results obtained from ILP solver and ACO-ILP approaches, for various types of trip gaps, namely the actual trip gaps, historical daily and weekly trip gaps, and simulated trip gaps. The examination of the comparison was conducted using a sample of 30 randomly selected cases from the testing dataset, as described in **Section 5.4.1**. The objective of conducting a comparison between the ILP solver and the proposed ACO-ILP algorithm is to showcase the efficacy of these two algorithms in addressing varying issue sizes, hence emphasizing their scalability. Furthermore, the objective of comparing the simulated demands using the Monte Carlo approach with the baseline scenarios, specifically the historical daily and weekly trip gaps, is to demonstrate the efficacy of reducing and distributing the uncertainty in demand through the medium of demand and variance prediction.

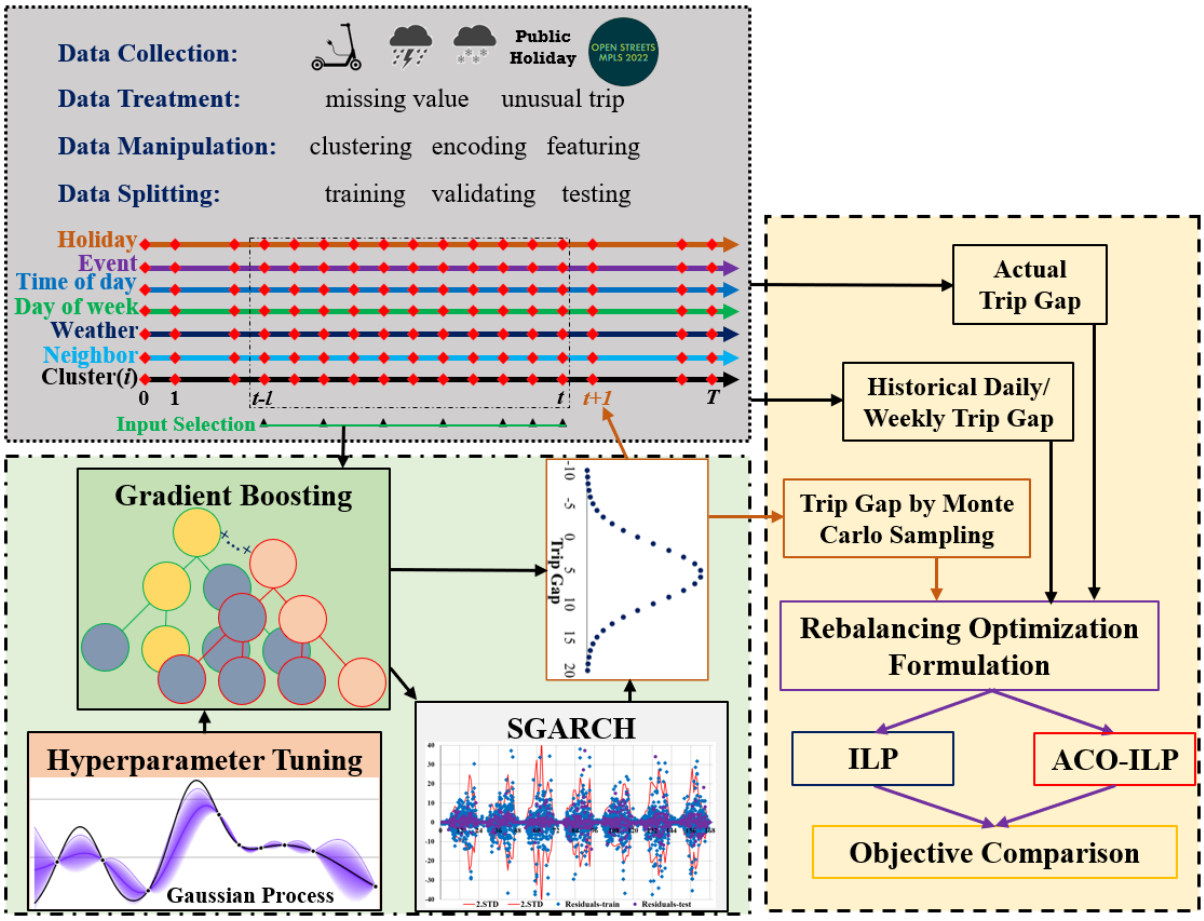

Figure 5.1 Research framework

5.3.2 Demand prediction by GB

The short-term trip gap or net demand of shared e-scooters was forecasted using GB, a machine-learning technique pioneered by Friedman [62], as seen in **Figure 4.1**. The GB model is an ensemble of decision trees that operates on a boosting framework, effectively enhancing prediction accuracy by progressively introducing new weak learners (decision trees) to minimize residual errors stemming from the preceding learners, as depicted in **Figure 5.2**. Built upon the foundation of classification and regression trees, GB possesses applicability in both classification and regression tasks. The intricate formulation and algorithm underpinning GB were elucidated by [62], while the iterative process (entailing the accumulation of decision trees) within the GB regressor encompasses several pivotal stages: calculation of the negative gradient (initializing the initial prediction with the mean value), adaptation of a regression tree to prognosticate the negative gradient, determination of the gradient descent step size (or learning rate), and refinement of the GB model or prediction efficacy. For the current investigation, the GB model was trained employing a Python module housed within the Scikit-Learn package, specifically the *GradientBoostingRegressor* [63].

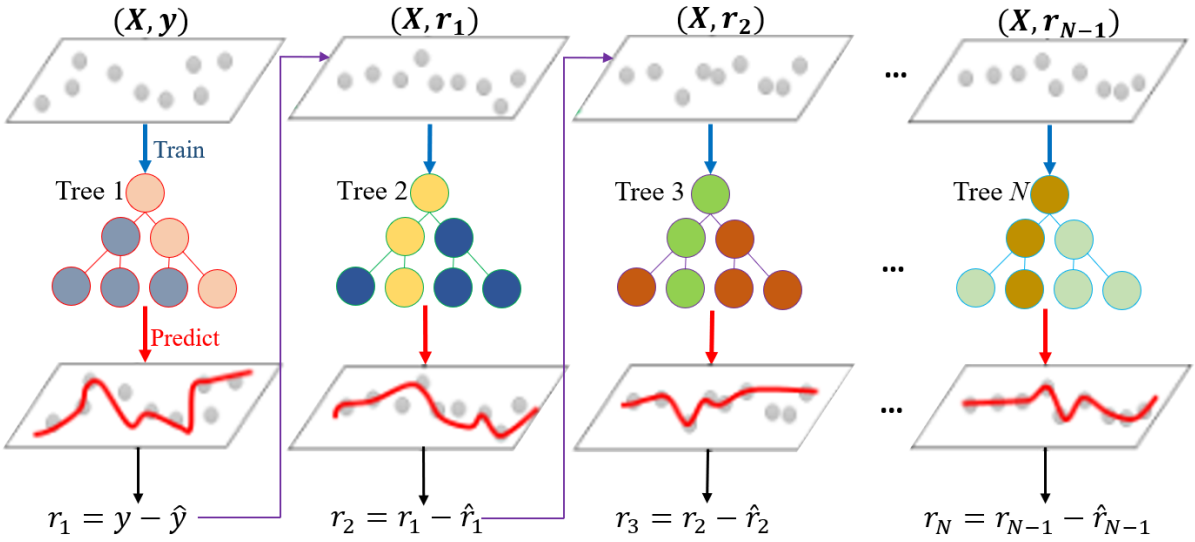

Figure 5.2 Flowchart of gradient boosting (GB)

The performance of the GB regressor can be equivalent to that of deep learning models [64], but it necessitates appropriate feature selection and hyperparameter tweaking. Several methodologies exist for hyperparameter optimization, including manual search, grid search, random search, sequential model-based approach, and population-based approach. The sequential-based technique is frequently employed for optimizing hyperparameters in both machine learning and deep learning domains due to its capacity to yield a near-globally optimal solution in a relatively short amount of computational time. The aforementioned methodology employs surrogate and acquisition functions in a sequential manner to propose a novel candidate until the predetermined stopping criteria are met. Among the well-recognized algorithms embracing this approach are Bayesian Optimization (BO) and tree-structured Parzen estimator. The Bayesian optimization (BO) algorithm employs a Gaussian process to create a surrogate function based on the assessed samples, which initially consist of randomly selected

data points. The selection of the new candidate is determined by maximizing the expected performance, such as minimizing the mean squared error (MSE) on evaluation data. This is achieved by utilizing acquisition functions such as the probability of improvement, expected improvement, or lower confidence bound. Elaborative insights into the BO algorithm can be sourced from existing literature [65]. The BO method was chosen in this study for its capacity to effectively address local optima. This is attributed to the inclusion of a parameter, *kappa*, which effectively balances the tradeoff between exploration and exploitation.

The current investigation utilized BO with the lower confidence bound to tune the hyperparameters of the GB regressor in the Python package, Scikit-Optimize or *skopt* [66]. Predominantly, the default values were retained for most parameters of the BO algorithm, with the exception of key parameters. Specifically, the number of initial random samples (*n_random_starts*), the total number of evaluations (*n_calls*), and the coefficient of the lower confidence bound (*kappa*) were configured to 50, 200, and 1.8, respectively. The objective of BO optimization centered on minimizing the MSE of evaluation data and its proportion to that of training data, denoted as *MSE_eval* + *MSE_eval*/*MSE_train*. The primary goal of this objective function is to minimize overfitting and decrease the training time by discouraging the recommendation of sophisticated models that result in substantial reductions in *MSE_train* and only marginal reductions in *MSE_eval*. This BO algorithm configuration was implemented to optimize five hyperparameters of the GB algorithm: number of boosting stages (*n_estimators*), maximum depth of the decision tree (*max_depth*), learning rate (*learning_rate*), the lookback length (l), and sampling rate (r). Notably, two parameters (l and r) were associated with the selection of input variables (as shown in **Figure 5.1**), facilitating the GB regressor to prognosticate the forthcoming trip gap ($t + 1$). To make hourly predictions, the input selection was made by sampling from step t to step $t - l - 1$. One sample was chosen within each interval of length r, ranging from $t - l - 1$ to $t - r$. All of the samples were selected from the range $t - r - 1$ to t. As an illustration, given the values $l = 24$ and $r = 3$, the historical data list can be represented as $[t - 23, t - 20, t - 17, t - 14, t - 11, t - 8, -5, t - 2, t - 1, t]$. The values of l and r were within the intervals [13, 170] and [1, 13], respectively. The duration of lookback length encompassed the weekly trend in order to anticipate the hourly trip-gap. The remaining three hyperparameters of GB, namely *n_estimators*, *max_depth*, and *learning_rate*, were tuned within the specified ranges of [5, 400], [1, 20], and [0.01, 0.5], correspondingly. The remaining settings of the GB technique were left at their default values. Specifically, the loss function was specified as the squared error, the minimum number of samples required to perform a split was set to 2, and the quality of a split was determined using the Friedman mean squared error.

5.3.3 Variance prediction by SGARCH

Despite the advanced performance achieved by machine learning algorithms, these models still exhibit prediction errors, typically quantified using metrics like MSE, Root-Mean-Square Error (RMSE), or Mean Absolute Error. Consequently, providing e-scooters based solely on the predictions derived from these models (disregarding the error term) would lead to a Service Level Type I, or a probability of encountering shortages, of only 50%. To elucidate, if e-scooters are allocated to 100 locations in accordance with predicted demands, approximately 50 of these locations (right-hand side of the residuals' histogram or probability distribution) may face

shortages. In essence, the predicted demand represents an expected value or mean around which nearly 50% of the actual demand fluctuates above or below. In an effort to address this issue, historical mean and variance have been utilized to simulate stochastic scenarios under an assumed demand distribution. Nevertheless, the presumed distributions for both trip-starts and trip-ends are impractical, particularly when it comes to shared e-scooters, as discussed in **Section 5.4.1**, with elevated uncertainty translating into heightened operational costs. As seen in **Figure 5.3**, two Gaussian trip-gap models exhibiting varying degrees of variability yield distinct expected unmet demands—clearly, greater uncertainty corresponds to increased expected unmet demand. Accordingly, we can optimize operational costs by diminishing demand uncertainty through the utilization of demand forecast techniques and the selection of suitable variation and distribution models.

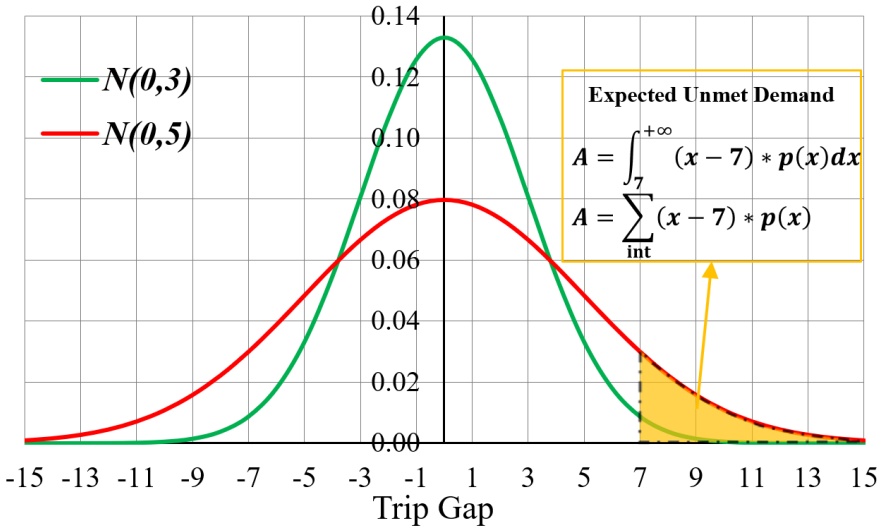

Figure 5.3 Effect of demand uncertainty on expected unmet demand

Recently, a discovery was made that the residuals of a short-term demand prediction model for shared e-scooters did not exhibit the characteristics of white noise [29]. As a result, this study embarked on an investigation into the heteroskedastic nature of the residuals derived from trip gap prediction. Given that prediction in this context entails a time-series approach, a Lagrange multiplier test was conducted to ascertain whether the residuals from the previous phase adhered to homoskedasticity; if they did, the variance would remain constant. Conversely, if homoskedasticity was not observed, the conditional variance of these residuals was estimated through the employment of the SGARCH model. An Autoregressive Conditional Heteroskedasticity (ARCH) model can be employed to forecast future variance based on conditional variance, distinguishing between high and low volatility periods. This model only includes past residuals as independent variables. However, its generalized counterpart (GARCH) incorporates not only prior residuals but also previously predicted variances [67]. To accommodate the seasonal patterns inherent in the data, the SGARCH model was developed by augmenting the GARCH model with seasonal residuals and predicted variances. A comprehensive exposition of this model, along with its extensions, can be found in the reference manual of the STATA software [67]. The fundamental expression of the SGARCH(p,q)(P,Q,S) model is presented as follows:

$$y_{t+1} = \boldsymbol{X}_{t+1}\beta + \epsilon_{t+1} \quad (5.1)$$

$$\sigma_{t+1}^2 = a_0 + \sum_{i=1}^{p} a_i \varepsilon_{t+1-i}^2 + \sum_{i=1}^{q} b_i \sigma_{t+1-i}^2 + \sum_{i=1}^{P} c_i \varepsilon_{t+1-iS}^2 + \sum_{i=1}^{Q} d_i \sigma_{t+1-iS}^2 \quad (5.2)$$

In this context, y_{t+1} represents the residuals originating from the demand prediction model, GB. Consequently, $\boldsymbol{X}_{t+1} = \mathbf{1}$, and y_{t+1} is determined by the summation of a constant value (β) often in proximity to zero, and the disturbance term ϵ_{t+1}. This model was constructed through the maximum log-likelihood estimator, employing versatile distributions such as the normal distribution $\epsilon_{t+1} \sim N(0, \sigma_{t+1}^2)$, Student's t distribution $\epsilon_{t+1} \sim t(0, \sigma_{t+1}^2, \mathrm{df})$, and a generalized error distribution. For the purpose of forecasting the hourly conditional variance σ_{t+1}^2, the SGARCH model encompassed parameters like $\sigma_t^2, \sigma_{t-1}^2, \sigma_{t-2}^2, \sigma_{t-23}^2, \sigma_{t-47}^2, \varepsilon_t^2, \varepsilon_{t-1}^2$, $\varepsilon_{t-2}^2, \varepsilon_{t-23}^2$ and ε_{t-47}^2. Insignificant parameters (95% confidence level) were eliminated. This signifies that the estimated variance at time t+1 is significantly influenced by the residuals and predicted variances from recent time steps (t, t-1, t-2), along with the corresponding hours from preceding days (t-23 and t-47). Such estimation of conditional variance effectively distributes uncertainty across the day, yielding a lower variance during nighttime and a heightened variance during daytime. Due to its emphasis on recent trends, the SGARCH model exhibits a greater tolerance for extended-term fluctuations when contrasted with daily variance (i.e., variance at the same hour of the day). Each cluster underwent independent SGARCH model training using the STATA statistical software [67]. Both the normal and Student's t distributions were considered, with the choice of distribution being contingent on the one yielding the smallest standard deviation. This chosen distribution, along with the projected variance, was subsequently leveraged in the Monte Carlo simulation to generate demand uncertainty.

5.3.4 Description of rebalancing problem

Given the constraints imposed by data availability and the intricate operational dynamics of shared micromobility encompassing bicycles, electric bikes (e-bikes), and electric scooters (e-scooters), the formulation of operational planning often necessitates reliance on a range of assumptions, which can diverge across different research endeavors. While certain assumptions can be managed via parameter configuration, worst-case analyses, and similar approaches, this study has established the subsequent assumptions.

- ***Assumption* 1**: The time distribution of both trip-starts (pickups) and trip-ends (drop-offs) within a specific cluster is assumed to follow a uniform pattern across the time interval (Δt). This implies that all e-scooters from trip-ends can potentially be used for pickup trips, particularly when the drop-off demand is less than the pickup demand (resulting in a positive trip gap). To illustrate, consider an empty cluster with 15 trip-starts and 10 trip-ends, yielding a trip gap of +5. Under ideal conditions and no other disruptions, this cluster would experience only five unmet demands. However, if all trip-starts occur in the first half of the time interval and all trip-ends are concentrated in the second half, the unmet demands could escalate to as many as 15.
- ***Assumption* 2**: The demand within each cluster remains constant throughout the planning and rebalancing process.
- ***Assumption* 3**: Due to the dockless sharing system, a user retrieves an e-scooter when it's accessible within the same cluster, irrespective of walking distance; otherwise, the user opts out

of the system, leading to unmet demand.

- ***Assumption* 4**: Faulty or broken e-scooters are exclusively fixed at the depot, while e-scooters with low-battery levels are either recharged at charging stations or brought back to the depot.
- ***Assumption* 5**: A single rebalancing vehicle is available, and it is required to visit all nodes, encompassing both the charging stations and demand clusters.

The process of minimizing demand uncertainty is primarily achieved through demand prediction, which is evidenced by the lower Mean Squared Error (MSE) compared to historical averages. With the inclusion of explanatory features in the demand prediction model (GB), the residuals are expected to exhibit characteristics of white noise or a random walk. Nevertheless, the residual uncertainty, represented by variance, can be further addressed through a variance prediction model tailored for datasets characterized by heteroscedasticity. The key purpose of variance prediction is to distribute uncertainty using the principle of conditional variance, focusing on temporal variance. By utilizing the forecasted trip gaps and variances, the operator can strategically relocate the limited e-scooters to areas where the potential profit is maximized.

This study focuses on the rebalancing problem inside a complete network $G = (N, A)$, where N represents the collection of all nodes, encompassing the depot, charging stations, and demand clusters, while A denotes the arcs connecting these nodes. Additional symbols employed in this study can be found in **Table 5.2**. This study examines three distinct categories of e-scooters: malfunctioning (or broken or faulty), low-battery, and operational (or usable) e-scooters. Defective electric scooters are retrieved and transported to the designated facility for necessary repairs, while e-scooters with depleted battery levels are either relocated to charging stations or the depot to undergo the recharging process. "Faulty" in this context refers to e-scooters exhibiting electronic or structural issues warranting attention from depot technicians, a status usually communicated by users. The operator retains the prerogative to determine the battery threshold that designates e-scooters as "low-battery" (e.g., a battery level sufficient for an average trip duration or the entire planning period). Therefore, the operator's awareness of these two e-scooter types is presumed at the planning phase. The depot and charging stations are assumed to have zero demand, while Monte Carlo simulation was employed to generate the predicted net demand g_i^θ in scenario θ in each cluster i for the total scenarios of Θ. Within this framework, a single vehicle with a capacity of B is designated for relocating operational and low-battery e-scooters, as well as for retrieving faulty e-scooters. The overarching operational objective seeks to minimize the cumulative expense encompassing driving costs, pickup costs, penalty costs stemming from unmet demand, and the lingering presence of faulty and low-battery e-scooters within the system.

In this research, the net demand g_i^θ, generated b through Monte Carlo simulation via normal or Student's *t* distribution, is rounded to either an integer value or zero decimal places. A positive net demand signifies an excess of pickup demands compared to drop-off demands. Conversely, a negative net demand indicates a surplus of drop-off trips relative to pickup trips. Consequently, all variables and decision variables (outlined in **Table 5.2**) adopt nonnegative-integer values, barring x_{ij} and a_i, as they pertain to the e-scooter unit or the e-scooter trip unit. The rebalancing task investigated in this paper encompasses two primary categories of decision variables: routing variables (x_{ij} and a_{ij}), and pickup and drop-off variables associated with

various types of e-scooters ($p_i^f, p_i^l, d_i^l, p_i^u,$ and d_i^u). In prior research, it was usual practice to aggregate pickup and drop-off operations into a single decision variable. This approach involved characterizing pickup activities as positive indicators and drop-off activities as negative signs. In the present study, a clear distinction is made between the two activities, with the requirement that all choice variables be strictly positive integers (nonnegative-integer). Furthermore, the implementation of a penalty on pickup operations is proposed as a means to reduce the occurrence of needless pickups and to attain a certain service level.

For usable e-scooters, the number of pickups (p_i^u) is allowed if there are more usable e-scooters than the specific safety stock ($\underline{C_i}$), while the drop-offs (d_i^u) are constrained by the availability on the rebalancing vehicle. There is only one activity whether to pick up ($p_i^u > 0$) or to drop off ($d_i^u > 0$) usable e-scooters at each station *i*, otherwise there is no pickup or drop-off activity $p_i^u = d_i^u = 0$. Therefore, these two decision variables are strictly positive integers (non-integer). In each cluster i and scenario θ, unmet demand ($U_i^\theta > 0$) occurs when the available (v_i^u) and the drop-off amount (p_i^u) of usable e-scooters are less than the sum of the positive net demand ($g_i^\theta > 0$) and safety stock ($\underline{C_i}$). Consequently, the total inventory tends to approach the upper bound value of the predicted net demand under the constraint of total usable e-scooters. This approach leads to an improvement in the service level while reducing the impact of potential demand. Additionally, this study introduces another parameter, the minimum number of usable e-scooters ($\underline{C_i}$), also referred to as safety stock. This parameter allows the operator to consider potential demand, especially when utilizing a demand prediction model trained on historical ridership data, and address the limitations of assumptions (1)-(3) and distribution regulations. As reviewed in the previous chapter, certain regulations mandate that operators promptly remove excessive e-scooters that obstruct pedestrians or have an adverse effect on the aesthetic environment. Therefore, this study incorporates an additional penalty term, referred to as excess e-scooters (E_i^θ), into the rebalancing objective function. The inclusion of this term aims to prevent such unfavorable events from occurring. In this case, the excess e-scooters (E_i^θ) is strictly positive integer values (non-integer) as it represents the number of e-scooters that surpass a specific threshold value, $\bar{C}_i$.

Faulty e-scooters refer to those with electronic or frame issues that require repair by technicians at the depot, and this status is commonly reported by customers. The operator has the flexibility to define the threshold of battery level for categorizing e-scooters as low-battery (e.g., the battery level required for an average trip duration or the entire planning horizon). Therefore, the status of these two types of e-scooters is assumed to be known by the operator during the planning stage. v_i^f denotes the number of faulty e-scooters at each node *i*, and it is strictly a positive integer. Therefore, the variable p_i^f denoting the number of faulty e-scooters picked up at node *i*, is also strictly positive integer. This decision variable p_i^f is constrained by the number of faulty e-scooters in each node, v_i^f. Low-battery e-scooters can be charged at any charging station or the depot, while the number of low-battery e-scooters in each node are denoted as v_i^l. p_i^l denotes the number of low-battery e-scooters to be picked up at each node, so this decision variable is strictly a positive integer. The pickup activity (p_i^l) of low battery e-scooter may be required if there are low-battery e-scooters present in each demand cluster and if there are more low-battery e-scooters than charging docks at each charging station. On the

other hand, d_i^l denotes the number of low-battery e-scooters that can be dropped off at each node, hence this decision variable is also a strictly positive integer. The drop-off activity of low-battery e-scooters is constrained by the number of low-battery e-scooters on the rebalancing vehicle and only allowed if there are available charging docks.

As shown in **Figure 5.1**, the rebalancing optimization problem was formulated in two approaches, which were solved by ILP and ACO-ILP. To make it easy to understand, we provide a list of notations (see **Table 5.2**) for the rebalancing problem, while the formulation can be found in the following sections.

Table 5.2 List of notations for rebalancing optimization

Notation		Description
Set	N	Set of nodes (depot, charging stations, and clusters), with component n
	A	Set of links in the network, with component (i, j)
	Θ	Set of scenarios, with component θ
Parameter	v_i^f	Number of faulty e-scooters at node i
	v_i^l	Number of low-battery e-scooters at node i
	v_i^u	Number of usable e-scooters at node i
	D_i	Number of charging docks at node i
	$\bar{C}_i$	Maximum number of e-scooters at node i
	$\underline{C}_i$	Minimum number of usable e-scooters at node i
	g_i^θ	Trip gap in scenario θ and node i
	B	Capacity of vehicle
	c_{ij}	Driving distance between nodes i and j
	β_0	Unit cost of driving distance
	$\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$	Unit costs of penalty functions of picking up e-scooters, remaining faulty e-scooters, remaining low-battery e-scooters, unmet demand, and excess e-scooters, respectively
Variable	R_i^f	Nonnegative-integer: Remaining faulty e-scooters at node i
	R_i^l	Nonnegative-integer: Remaining low-battery e-scooters at node i
	h_i^f	Nonnegative-integer: number of faulty e-scooters on the vehicle at node i
	h_i^l	Nonnegative-integer: number of low-battery e-scooters on the vehicle at node i
	h_i^u	Nonnegative-integer: number of usable e-scooters on the vehicle at node i
	U_i^θ	Nonnegative-integer: unmet demand in scenario θ and at node i
	E_i^θ	Nonnegative-integer: excess e-scooters in scenario θ and at node i
Decision Variable	x_{ij}	Binary: 1 if the rebalancing vehicle passes the link (i, j), 0 otherwise.
	a_i	Nonnegative-integer: auxiliary variable for subtour elimination
	p_i^f	Nonnegative-integer: number of faulty e-scooters picked up at node i
	p_i^l	Nonnegative-integer: number of low-battery e-scooters picked up at node i
	d_i^l	Nonnegative-integer: number of low-battery e-scooters dropped off at node i
	p_i^u	Nonnegative-integer: number of usable e-scooters picked up at node i
	d_i^u	Nonnegative-integer: number of usable e-scooters dropped off at node i

5.3.5 Rebalancing formulation by ILP solver

As elucidated in **Section 5.2**, prior research efforts have undertaken the formulation of rebalancing quandaries, each characterized by diverse objective functions, with primary emphasis on metrics like total driving distance [46] and generalized cost [45, 56, 58, 59, 68]. Within these generalized cost functions, there exists a shared pool of terms, including factors such as driving distance or duration, and common constraints such as vehicle capacity, pickup and drop-off requirements, and more. However, these generalized cost functions are also marked by distinctive elements and constraints, tailored to the specific objectives of each study. For example, Chang et al. [45] introduced deterministic rebalancing models pertinent to dockless bike sharing, devising a generalized cost function that encompasses parameters such as driving cost, pickup and drop-off expenses, and penalties for unattended zones harboring pending requests. On the contrary, Dell'Amico et al. [56] designed stochastic rebalancing strategies for station-based bike sharing, with their generalized cost function integrating driving cost alongside penalized charges linked to the variance in supply (excess and shortage) concerning stochastic requests.

In this investigation, we likewise embrace a generalized cost function to act as the underlying objective for addressing the stochastic rebalancing task within the realm of dockless shared e-scooters. The fundamental aim of our generalized cost function is the minimization of costs tied to driving distances and the intricacies of pick-up procedures. Nonetheless, a distinctive feature of our approach lies in the incorporation of penalties that are specifically tailored to address instances of unfulfilled demand, surplus e-scooters, as well as the presence of malfunctioning (i.e., damaged or broken) and low-battery e-scooters. The penalty attributed to unmet demand, U_i^{θ}, encapsulates not only the simulated demand for each scenario, g_i^{θ}, but also takes into consideration the parameter $\underline{C_i}$, which pertains to the safety stock. In parallel, the imposition of an excess e-scooter penalty becomes effective when the cumulative number of e-scooters in a given zone crosses a predefined threshold, $\bar{C_i}$. By virtue of these enhancements, our objective function gains enhanced interpretability, particularly for individuals who lack extensive expertise in the field, thereby facilitating pragmatic parameter fine-tuning during the practical implementation phase. Additionally, our formulated objective function conveniently facilitates a trade-off consideration encompassing the realms of driving distances and the intricacies associated with pick-up and drop-off activities. This provides a valuable avenue for striking an optimal balance between these facets, rather than solely adhering to the strict confines dictated by request-based constraints.

As delineated in the research framework depicted in **Figure 5.1**, the technique of Monte Carlo simulation was harnessed to engender the demand uncertainty, drawing upon the projected trip gap information expounded upon in **Section 5.3.2**, alongside the envisaged variance and associated distribution elaborated upon in **Section 5.3.3**. The simulation process yielded randomized trip gap scenarios, with an equal and uniform likelihood distribution across the spectrum. Consequently, the formulation of the transient rebalancing strategy for shared e-scooters can be articulated as follows.

Minimize

$$\beta_0 \sum_{(i,j)\epsilon A} c_{ij}\, x_{ij} + \beta_1 \sum_{i\epsilon N}\left(p_i^f + p_i^l + p_i^u\right) + \beta_2 \sum_{i\epsilon N} R_i^f + \beta_3 \sum_{i\epsilon N} R_i^l + \frac{\beta_4}{\Theta}\sum_{i\epsilon N;\theta\epsilon\Theta} U_i^\theta + \frac{\beta_5}{\Theta}\sum_{i\epsilon N;\theta\epsilon\Theta} E_i^\theta \quad (5.3)$$

Subject to

$$\sum_{i\epsilon N} x_{ij} = 1 \qquad \forall j\epsilon N \quad (5.4)$$

$$\sum_{j\epsilon N} x_{ij} = 1 \qquad \forall i\epsilon N \quad (5.5)$$

$$a_i - a_j + N x_{ij} \le N - 1 \quad \forall i,j|(i,j)\epsilon A - \{1\}, i \ne j \quad (5.6)$$

$$0 \le p_i^f \le v_i^f \qquad \forall i\epsilon N \quad (5.7)$$

$$0 \le p_i^l \le \max(0, v_i^l - D_i) \quad \forall i\epsilon N \quad (5.8)$$

$$0 \le d_i^l \le \max(0, D_i - v_i^l) \quad \forall i\epsilon N \quad (5.9)$$

$$0 \le p_i^u \le \max(0, v_i^u - \underline{C_i}) \quad \forall i\epsilon N \quad (5.10)$$

$$h_j^f - h_i^f - p_j^f + M\left(1 - x_{ij}\right) \ge 0 \quad \forall i,j|(i,j)\epsilon A, j \ge 2 \quad (5.11)$$

$$h_i^f - h_j^f + p_j^f + M\left(1 - x_{ij}\right) \ge 0 \quad \forall i,j|(i,j)\epsilon A \quad (5.12)$$

$$R_i^f = v_i^f - p_i^f \qquad \forall i\epsilon N \quad (5.13)$$

$$h_j^l - h_i^l - p_j^l + d_j^l + M\left(1 - x_{ij}\right) \ge 0 \quad \forall i,j|(i,j)\epsilon A, j \ge 2 \quad (5.14)$$

$$h_i^l - h_j^l + p_j^l - d_j^l + M\left(1 - x_{ij}\right) \ge 0 \quad \forall i,j|(i,j)\epsilon A \quad (5.15)$$

$$R_i^l = \max\{v_i^l - D_i, 0\} - p_i^l \qquad \forall i\epsilon N \quad (5.16)$$

$$h_j^u - h_i^u - p_j^u + d_j^u + M\left(1 - x_{ij}\right) \ge 0 \qquad \forall i,j|(i,j)\epsilon A, j \ge 2 \quad (5.17)$$

$$h_i^u - h_j^u + p_j^u - d_j^u + M\left(1 - x_{ij}\right) \ge 0 \qquad \forall i,j|(i,j)\epsilon A \quad (5.18)$$

$$\underline{C_i} - v_i^u + p_i^u - d_i^u + \max\{g_i^\theta, 0\} - U_i^\theta \le 0 \qquad \forall i\epsilon N, \theta\epsilon\Theta \quad (5.19)$$

$$R_i^f + R_i^l + v_i^u - p_i^u + d_i^u - g_i^\theta - \overline{C_\iota} - E_i^\theta \le 0 \quad \forall i\epsilon N, \theta\epsilon\Theta \quad (5.20)$$

$$h_i^f + h_i^l + h_i^u \le B \quad \forall i\epsilon N \quad (5.21)$$

The primary objective of the transient rebalancing process, as articulated in Eq. 5.3, is to minimize the amalgamated cost encompassing driving distance and a constellation of penalty costs. These penalties pertain to pickup undertakings, the retention of faulty and low-battery e-scooters within the system, unmet demands, and an undue accumulation of e-scooters. The constraints delineated in Eq. 5.4–5.6 encapsulate routing-related stipulations that govern arrivals and departures at all nodes while precluding the emergence of subtours. Eq. 5.7–5.10 are pickup and drop-off constraints for each type of e-scooters. Eq. 5.11 and Eq. 5.12 encapsulate loading rules for faulty e-scooters onto the rebalancing vehicle, while Eq. 5.13 ensures the equivalence between initial faulty e-scooters minus those picked up and the remaining faulty e-scooters. Similarly, Eq. 5.14 and 5.15 embody loading and unloading conservation for low-battery e-scooters, and the remaining is computed through Eq. 5.16. The loading and unloading balance for usable e-scooters is enforced by Eq. 5.17 and 5.18. The quantification of unmet demand ensues via Eq. 5.19, necessitating the tally of usable e-scooters within each cluster to surpass a stipulated threshold ($\underline{C_i}$) following both the rebalancing procedure and the end of the planning time frame. This threshold serves as a safety buffer, accommodating the aforementioned assumptions, along with potential demands and regulatory

frameworks. The number of excessive e-scooters is quantified by Eq. 5.20, a measure that forestalls overloading at particular locations and ensures swift intervention. Lastly, the vehicle capacity is constrained by Eq. 5.21.

5.3.6 Rebalancing formulation by hybrid ACO-ILP algorithm

Solving NP-hard optimization problems with ILP solvers often presents challenges in generating satisfactory solutions, let alone feasible ones, within imposed time constraints, particularly in the realm of stochastic problems. Conversely, heuristic algorithms, although incapable of ensuring exact solutions, frequently yield improved feasible outcomes within a confined computational timeframe. Against this backdrop, this study proposes a hybrid heuristic technique named the ACO–ILP algorithm. This approach combines ant colony optimization (ACO) with an ILP solver to tackle the aforementioned rebalancing optimization predicaments. A similar hybrid algorithm, denoted ACO–CP, was previously formulated for deterministic bike sharing rebalancing [68]. Nonetheless, our approach diverges in its treatment of the rebalancing vehicle's routing issue, viewing it as a variant of the traveling salesman problem. This facet can be addressed using the ACO algorithm, which optimizes the driving distance cost and penalty cost through input from the above-described ILP solver. The ACO algorithm, introduced in the early 1990s, draws inspiration from the foraging conduct of ants, who employ pheromone trails for indirect communication regarding the shortest path between their nest and food sources [69]. The ACO algorithm emulates this foraging behavior by deploying artificial ants to progressively adjust the path based on transition probabilities, influenced by the concentration of "pheromones" and a visibility function.

Algorithm 5.1 delineates the rebalancing optimization procedure through the utilization of ACO. The initial steps (Lines 1–4) encompass the inputs required for ACO, encompassing a graph delineated by a set of nodes N and a set of links A, a distance function denoted as c that encapsulates the driving distance derived from Bing Maps (https://www.bing.com/maps), a comprehensive cost function that amalgamates the driving distance cost and penalty costs, and the pertinent ACO parameters. Within each iteration, a sequence of actions is undertaken, including the computation of transition probabilities, the construction of route sequences for individual ants based on the calculated probabilities, the assessment of the performance of each ant, the retention of the best-found solution, and the updating of the pheromone trails. These pheromone trails are initially set with uniform weights (values of 1). The intricate formulations underpinning the ACO algorithm are enumerated in the following depiction.

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}(\eta_{ij})^{\beta}}{\sum_{l \epsilon N_i^k}[\tau_{il}(t)]^{\alpha}(\eta_{il})^{\beta}} \quad \forall j \epsilon N_i^k \tag{5.22}$$

$$\eta_{ij} = 1/c_{ij} \tag{5.23}$$

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{5.24}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{5.25}$$

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L_k}, & \text{The k}^{th} \text{ ant passes between } i \text{ and } j \\ 0, & \text{Otherwise} \end{cases} \tag{5.26}$$

Algorithm 5.1: the ACO algorithm for the rebalancing problem

Input:
    complete non-directed graph: $G = (N, A)$
    distance function: c
    cost function: L
    set all of the ACO parameters $\{\alpha, \beta, \rho, \#ants, \#iterations\}$
initialize all of the pheromone trails τ_0
$iteration\_result = [\,]$
for $t \leftarrow 1$ to $\#iterations$ **do**
    calculate the probability matrix: $P_t = [\tau_{t-1}]^{\alpha}(1/c)^{\beta}$
    for $k \leftarrow 1$ to $\#ants$ **do**
        $route_{Ant(k)}[0] \leftarrow 0$
        for $i \leftarrow 1$ to $N - 1$ **do**
            list the nodes to be visited: N_i^k
            normalize the probability of the remaining nodes: $P_{t,i}^k$
            randomly choose the next node according to the probability: $next\_node$
            $route_{Ant(k)}[i] \leftarrow next\_node$
        evaluate the cost function of each ant: L_k
    $iteration\_result.\mathrm{add}([\min\{L_k\};\ \mathrm{argmin}\{L_k\}])$
    update the pheromone trails: $\tau_t = (1 - \rho)\tau_{t-1} + \Delta\tau_t$

In the presented equations, $P_{ij}^k(t)$ signifies the probability linked to ant k moving from the ongoing node i to the subsequent node j within the t iteration. The ensemble N_i^k stands for the collection of nodes that ant k has yet to traverse. The visibility of node j from node i, denotes as η_{ij}, is characterized as the reciprocal of the distance between these two nodes. The parameters α and β denote the respective significance of the pheromone and visibility aspects (with the pheromone undergoing updates as per Eq. 5.24). Meanwhile, ρ and $\Delta\tau_{ij}(t)$ symbolize the coefficient for pheromone evaporation and the cumulative pheromone deposited by all ants (amounting to *m* ants), correspondingly. Lastly, L_k refers to the cost function, encompassing the description of both driving distance and penalty costs, associated with ant k. In the course of this investigation, the ACO algorithm was executed utilizing the Scikit-opt Python library [70]. Default values were assigned to $\alpha, \beta,$ and ρ, with their values set as 1, 2, and 0.1, respectively.

In this subsection, the objective function remains consistent with the one described in the preceding section (Eq. 5.3), with the exception that the integration of the ACO algorithm and ILP is employed to tackle the problem of routing for the rebalancing vehicle and the corresponding pickup/drop-off activities. In this composite approach, the ILP solver handles the optimization of pickups and drop-offs for a predetermined route sequence (... $\rightarrow i \rightarrow j \rightarrow \cdots$, $\forall i, j \epsilon N$) established by the ACO algorithm. Subsequently, the ACO algorithm fuses the penalty cost with the driving distance cost to iteratively enhance the arrangement of the route sequence. Consequently, the ILP formulation for rebalancing involving a pre-determined route sequence is represented as follows:

Minimize

$$\beta_1 \sum_{j\epsilon N}\left(p_j^f + p_j^l + p_j^u\right) + \beta_2 \sum_{j\epsilon N} R_j^f + \beta_3 \sum_{j\epsilon N} R_j^l + \frac{\beta_4}{\Theta} \sum_{j\epsilon N;\theta\epsilon\Theta} U_j^\theta + \frac{\beta_5}{\Theta} \sum_{j\epsilon N;\theta\epsilon\Theta} E_j^\theta \quad (5.27)$$

Subject to

$$0 \le p_j^f \le v_j^f \quad \forall j\epsilon N \quad (5.28)$$

$$0 \le p_j^l \le \max(0, v_j^l - D_j) \quad \forall j\epsilon N \quad (5.29)$$

$$0 \le d_j^l \le \max(0, D_j - v_j^l) \quad \forall j\epsilon N \quad (5.30)$$

$$0 \le p_j^u \le \max(0, v_j^u - \underline{C_j}) \quad \forall j\epsilon N \quad (5.31)$$

$$h_j^f = h_i^f + p_j^f \quad \forall i, j\epsilon N \quad (5.32)$$

$$R_j^f = v_j^f - p_j^f \quad \forall j\epsilon N \quad (5.33)$$

$$h_j^l = h_i^l + p_j^l - d_j^l \quad \forall i, j\epsilon N \quad (5.34)$$

$$R_j^l = \max(0, v_j^l - D_j) - p_j^l \quad \forall j\epsilon N \quad (5.35)$$

$$h_j^u = h_i^u + p_j^u - d_j^u \quad \forall i, j\epsilon N \quad (5.36)$$

$$\underline{C_j} - v_j^u + p_j^u - d_j^u + \max\{g_j^\theta, 0\} - U_j^\theta \le 0 \quad \forall j\epsilon N, \theta\epsilon\Theta \quad (5.37)$$

$$R_j^f + R_j^l + v_j^u - p_j^u + d_j^u - g_j^\theta - \overline{C_j} - E_j^\theta \le 0 \quad \forall i\epsilon N, \theta\epsilon\Theta \quad (5.38)$$

$$h_j^f + h_j^l + h_j^u \le B \quad \forall j\epsilon N \quad (5.39)$$

In this context, Eq. 5.28–5.31 correspond to the pickup and drop-off constraints at the present node j. Eq. 5.32, 5.34, and 5.36 provide the cumulative quantities of faulty, low-battery, and usable e-scooters, respectively, situated on the rebalancing vehicle at the current node j. These values are the summation of the quantities on the rebalancing vehicle at the preceding node i and the quantities that are either picked up or dropped off at the current node j. Meanwhile, Eq. 5.33 and 5.35 furnish the remaining amounts of faulty and low-battery e-scooters, respectively. Eq. 5.37 and 5.38 yield the unmet demands and surplus numbers of e-scooters, respectively. Finally, Eq. 5.39 enforces the stipulation on the capacity of the rebalancing vehicle. The commencement of the rebalancing vehicle's journey is mandated from the depot, consequently necessitating that all decision variables pertaining to this specific node are set to zero, except for the variables associated with the pickup of usable e-scooters p_1^u and the number of usable e-scooters on the rebalancing vehicle h_1^u.

5.4 Application of demand and variance prediction

5.4.1 Data collection and description

Obtaining true demand information is crucial for effective operational planning. However, this data is often challenging to access unless operators authorize its extraction from user application interactions [30]. Given these data constraints, past ridership data is frequently utilized to assess the viability of suggested models or systems, as in references [4, 28, 29, 33, 34, 61]. Likewise, in this research, historical data is employed as a case study, with potential

demand being managed via the safety stock parameter, denoted as the minimum number of functional e-scooters ($\underline{C_i}$). In practical implementation, the proposed framework, particularly the demand forecasting model, may necessitate training using true demand data that encompasses unmet demand, as elaborated by Ham et al. [30].

The numerical examination was conducted utilizing an openly available dataset representing shared e-scooter ridership in Minneapolis, Minnesota (accessible at https://opendata.minneapolismn.gov). This dataset encompasses a total of 961,040 trips taken during the timeframe spanning May 13 to November 25, 2019. Pertinent trip details include the trip's ID, trip distance, trip duration, start/end date time, and start/end center line ID. To determine geographical coordinates, the central point of the street where each trip was picked up and dropped off was considered. During the process of data refinement, records with missing values were excluded, while those fulfilling the criteria of having a trip distance ranging from 20 meters to 10 kilometers, a trip duration spanning 20 seconds to 2 hours, alignment with the study period (from May 14 to November 24, 2019), and adherence to the study's geographical boundary were included. Post-cleaning, a total of 813,970 trips remained in the dataset, exhibiting an average duration of approximately 13 minutes, and covering an average distance of about 1.72 kilometers. As previously discussed, the trips were subjected to clustering using the *k*-means clustering algorithm with varying cluster counts of 15, 30, and 60 clusters, as depicted in **Figure 5.4**. For the purpose of forecasting hourly net demand ($\Delta t = 1$ hour), a dataset consisting of 4,680 samples was assembled. In the model development process, 80% of this data was allocated for training the model, while the remaining 20% was designated for model testing, as illustrated in **Figure 5.5**. To avert the risk of overfitting, a strategy of random sampling was employed to divide the training dataset into segments, with 75% employed for model construction and 25% reserved for model evaluation.

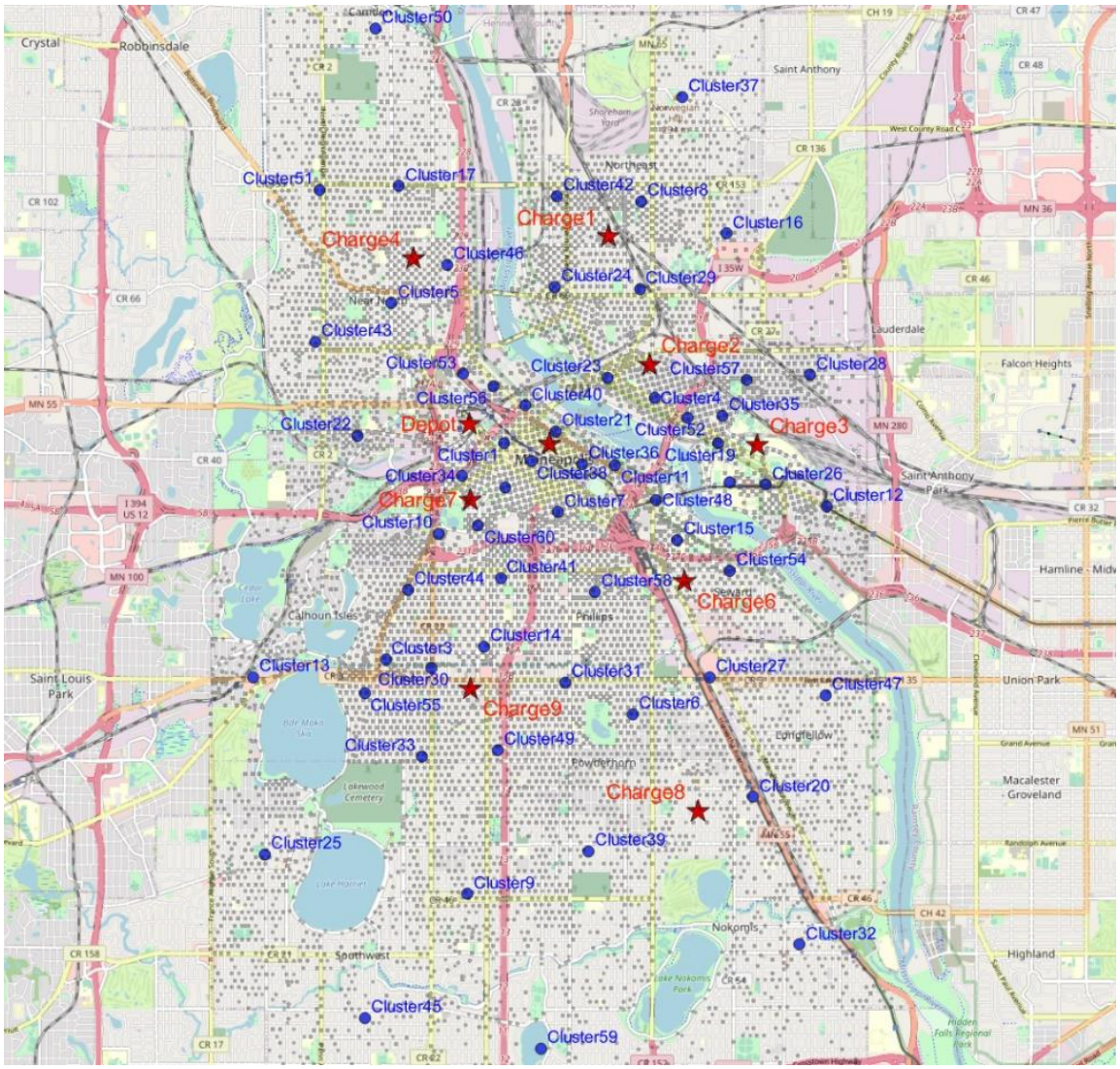

Figure 5.4 Trip clustering generated by the k-means algorithm (red stars = depot and charging stations; blue dots = centers of trip clusters; gray dots = street centers of pickup and drop-off trips)

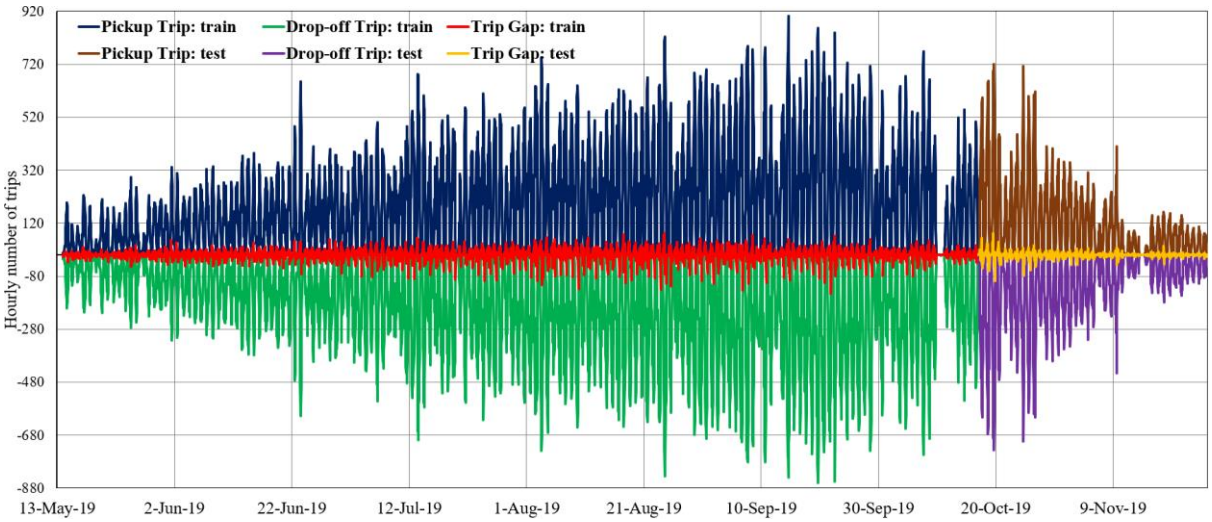

Figure 5.5 Hourly pickup and drop-off trips and the trip gap for shared e-scooters in Minneapolis, MN

Research findings have demonstrated a meaningful correlation between the utilization of shared e-scooters and various environmental variables such as weather conditions, public holidays, recurring yearly festivities, and weekly usage trends. In light of this, weather-related attributes encompassing temperature, precipitation, wind speed, humidity, wind gust, pressure, and dew point were procured from Weather Underground (www.wunderground.com). In instances where attribute data were absent, linear interpolation was employed to estimate these values. Furthermore, the model devised for predicting gaps between trips accounted for official public holidays, annual festivals, and noteworthy events such as open street events, a pride festival parade, a state fair festival, a stone arch bridge festival, and an uptown art fair. Notably, an exception was made for October 10, 2019, when the operation of shared e-scooters was suspended due to the state visit of the President of the United States to Minneapolis. It is vital to acknowledge that the demand for shared e-scooters is influenced by numerous factors. As such, it is inappropriate to assume a specific distribution—like the Poisson distribution—to represent the pickup and drop-off demand patterns, as indicated in **Figure 5.6**. This characteristic was verified through a goodness-of-fit test conducted on both daily patterns (at the same hour of the day) and weekly patterns (at the same hour of the day and day of the week). Given this insight, it's evident that rebalancing methodologies grounded in such assumptions (e.g., queue theory), which have commonly been applied to analyze shared bike services, may not be well-suited for examining shared e-scooter services.

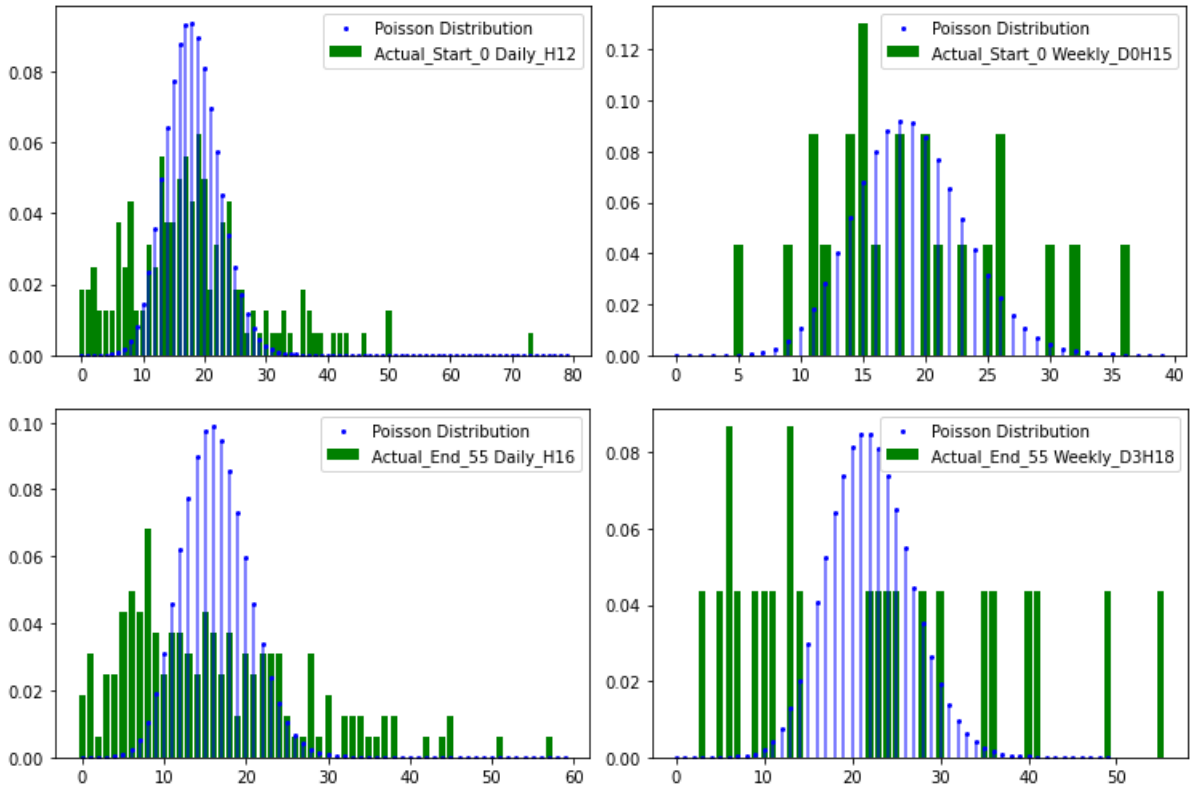

Figure 5.6 Histograms and Poisson distributions of the pickup and drop-off demands of shared e-scooters

5.4.2 Result of demand prediction

As previously mentioned, the *k*-means algorithm was applied to cluster the trips into groups of 15, 30, and 60. For enhanced predictive accuracy in trip gap estimations, the model was trained with spatial independence. This was achieved by incorporating external features (as outlined in **Section 5.3.2**), local historical data, and historical data sourced from four neighboring clusters into the model's inputs. Displayed in **Figure 5.7** (left), the convergence curve of Gaussian Process Bayesian Optimization (BO) is depicted, illustrating the hyperparameter tuning process for cluster 37 in the GB model. Hyperparameter tuning is a customary practice aimed at minimizing a loss metric on the evaluation dataset (e.g., *MSE_eval*). However, such optimization can potentially lead to overfitting, particularly when working with decision tree models. **Figure 5.7** (right) showcases the assessed GB models generated through BO, with the models arranged based on the sorted values of *MSE_train*. This visualization indicates that the minimal *MSE_eval* occurs within a range where the divergence between *MSE_train* and *MSE_eval* is notable. Beyond this point, *MSE_eval* deteriorates or loses its generalization capability. Additionally, it's worth noting that the modeling process in this region demands substantial inputs due to the inherent complexity of models, often necessitating a greater number of deeper decision-tree regressors and a lengthier historical data lookback. Hence, the objective function utilized for BO, encompassing the ratio of *MSE_eval* to *MSE_train*, effectively reduces training duration and promotes the construction of a more generalized GB model.

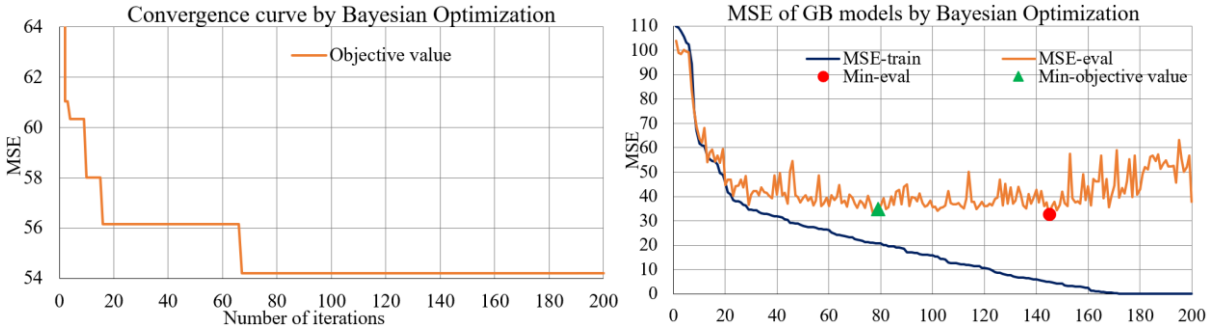

Figure 5.7 Hyperparameter optimization by Bayesian optimization for trip gap prediction

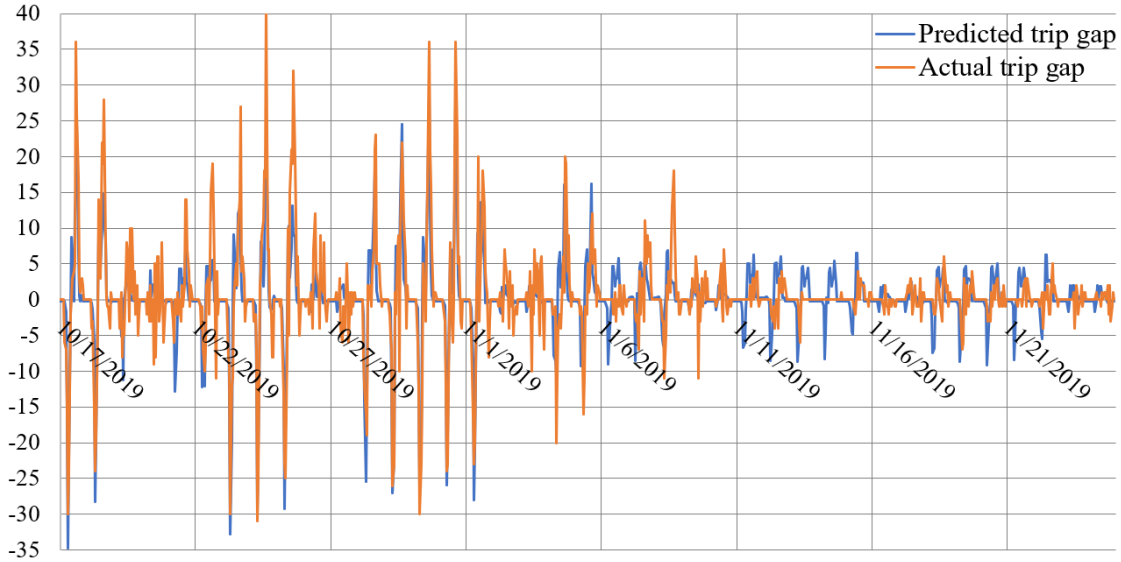

Figure 5.8 Trip gap predicted using the testing data for cluster 37

Figure 5.8 displays the trip gap projections generated by GB regressor for cluster 37. While the GB model effectively captured the temporal patterns, it exhibited residual errors in its predictions. Failing to address these discrepancies in the context of rebalancing planning could result in diminished service levels or profits compared to strategies that account for such errors. **Table 5.3** provides a comparative overview of the GB model's predictions with those of two benchmark models—the historical average model and the daily historical average model—using the RMSE as the metric for accuracy assessment. Our numerical findings derived from the training data showcase that the GB model yielded a significantly reduced RMSE, approximately 26% and 16% lower than the baseline historical average and daily historical average models, respectively. When evaluated against the testing dataset, the RMSE (indicative of uncertainty) associated with the GB model was roughly 19% and 15% lower than the corresponding RMSE values of the historical average model and the daily historical average model, respectively.

Table 5.3 Results of trip gap prediction and variance prediction

Model			15 Clusters	30 Clusters	60 Clusters
Trip Gap Prediction	Historical Average	RMSE-train	7.13	4.97	3.46
		RMSE-test	5.25	3.94	2.66
	Daily Historical Average	RMSE-train	6.00	4.47	3.17
		RMSE-test	5.10	3.73	2.55
	GB	RMSE-train	5.04	3.69	2.67
		RMSE-test	4.14	3.16	2.24
Variance Prediction for GB residuals	Constant Variance	Mean-STD	4.01	3.02	2.30
		Coverage	95.73%	96.15%	97.09%
	Daily Variance	Mean-STD	3.44	2.57	1.93
		Coverage	96.13%	95.65%	95.96%
	SGARCH Variance	Mean-STD	3.17	2.31	1.59
		Coverage	93.46%	91.55%	94.40%

5.4.3 Result of variance prediction

The utilization of SGARCH for predicting variance furnishes two pivotal parameters that find application in Monte Carlo simulation: the standard deviation (STD) and the distribution of residuals. As elaborated upon in **Section 5.3.3**, a lower STD or diminished uncertainty leads to a decrease in expected loss. Nonetheless, the STD must be minimized in a manner that doesn't compromise coverage—signifying the percentage of residuals contained within confidence bounds. For instance, the daily variance associated with GB residuals exhibited a Mean-STD (average standard deviation) that was smaller than that of constant variance, all without compromising coverage, as presented in **Table 5.3**. This outcome suggests that the variance prediction model possesses the potential to influence rebalancing planning by further mitigating uncertainty stemming from the demand prediction model. However, it's worth noting that the daily variance derived from historical data was incapable of capturing prolonged fluctuations, as evidenced by **Figure 5.9**. Notably, the SGARCH-derived variance exhibited heightened flexibility over time, encompassing seasonal or annual patterns, owing to a greater weighting

assigned to recent residuals compared to earlier ones. On the whole, in comparison to the daily variance, the SGARCH model yielded a marginally reduced Mean-STD; however, it also brought about a minor reduction in coverage.

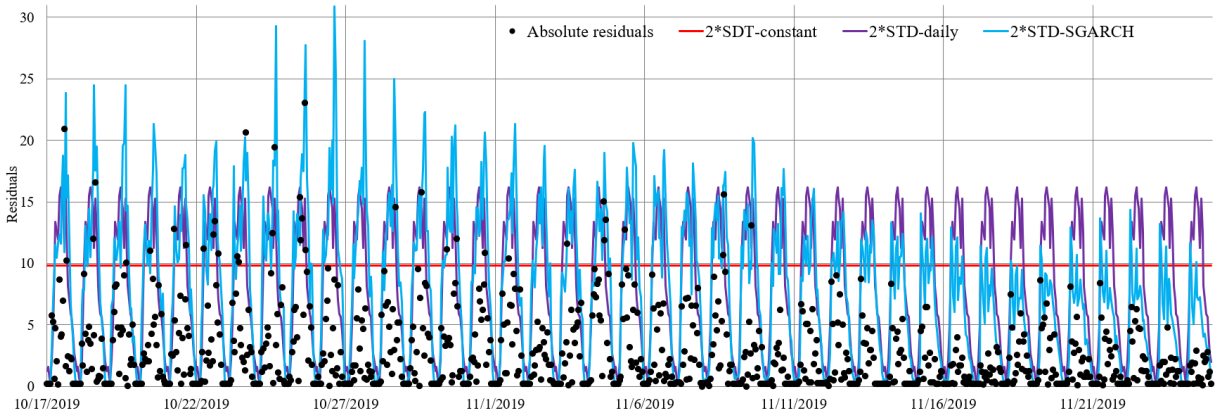

Figure 5.9 Variance prediction based on residuals of the GB model for cluster 37

5.5 Result of rebalancing optimization

5.5.1 Parameter settings

The accessible dataset includes trip information that doesn't encompass all the requisite operational planning parameters. Consequently, these parameters were simulated by assigning fixed and random values within designated ranges, as detailed in **Table 5.4**. The utilization of *k*-means clustering facilitated the division of the coordinates associated with dockless shared e-scooter trips. While augmenting the cluster count addressed Assumption 1, it simultaneously led to a decrease in the performance of demand prediction models, characterized by sparsity and random walk patterns. Additionally, this increment exponentially amplified the time required for rebalancing optimization. For the scenarios under consideration, the cluster count was established at 15, 30, and 60, yielding corresponding total node counts N (inclusive of the depot and charging stations) of 18, 35, and 70. Earlier studies typically restrict computational time to around 1 hour or 3600 seconds. Within this study, a maximum computational time of approximately 1 minute (or 60 seconds) per node was established, and the ultimate rounding time for the aforementioned scenarios was set at 20, 40, and 60 minutes, respectively. If optimization were carried out under the universal constraint of 3600 seconds for all three rebalancing problem instances, the performance of the ILP solver and ACO-ILP algorithm would be on par. Hence, a comparison of these two algorithms hinged on their speed in achieving optimal objective values. Alternatively, proportional allocation of computational time across these scenarios would enable a comparison of the two algorithms based on their respective optimal objective values. On the other hand, if the computational time for these three cases were set proportionally, we could compare the performance of these two algorithms based on their optimal objective values. The travel distance between nodes was acquired from Bing Maps. The testing dataset encompassed approximately 39 days or roughly five weeks, and 30 instances were randomly extracted from the high-demand timeframe spanning 10 am to 8 pm during the first (for the 15-cluster problem), second (for the 30-cluster problem), and third (for the 60-cluster problem) three weeks. The rebalancing process was executed using a single vehicle with a capacity of accommodating 35 e-scooters.

The total count of e-scooters was taken as 400, inclusive of 20 defective e-scooters (about 5%) and 60 e-scooters with low battery levels (approximately 15%). Consequently, the total pool of operational e-scooters amounted to 320, roughly equivalent to half of the hourly pickup demand during peak demand hours. These e-scooters classified as faulty, low-battery, and usable were allocated randomly across the clusters. Based on the average usage expenditure of shared e-scooters in Minneapolis, it was deduced that users typically spent around 3 USD per trip. For the purposes of this study, the penalty cost per unit of unfulfilled demand was defined as 2 USD, constituting approximately 67% of the revenue. The unit cost related to surplus e-scooters stood at 1 USD, while penalty costs associated with the remaining defective and low-battery e-scooters were determined as 5 and 3 USD, respectively. Travel distance expenses were assessed at a rate of 1 USD per kilometer traveled. To ensure that the same e-scooter wasn't both picked up and dropped off at the same location, a pickup cost of 0.1 USD was established. This decision not only prevented such instances but also served to harmonize the service level. This pickup cost represented approximately 5% of the unmet demand penalty, mirroring a service level categorized as Type II, denoting a 95% fulfillment level (i.e., demands with a probability below 5% didn't warrant incurring the pickup cost).

Table 5.4 Parameter settings for the rebalancing optimization

Parameter	15-Cluster	30-Cluster	60-Cluster
Computational time (minutes)	20	40	60
Testing week	1st, 2nd, 3rd	2nd, 3rd, 4th	3rd, 4th, 5th
Number of scenarios θ	100	100	100
Number of charging stations	2	4	9
Total e-scooters	400	400	400
Number of docks in each station (total of 100)	50	25	10–15
Maximum number of e-scooters in each cluster $\overline{C_i}$ (total of 700)	30–50	15–30	10–15
Minimum number of e-scooters in each cluster $\underline{C_i}$ (total of 80)	0–10	0–5	0–3
Number of faulty e-scooters in each cluster h_i^f (total of 20)	0–3	0–2	0–2
Number of low-battery e-scooters in each cluster h_i^l (total of 60)	0–10	0–5	0–3
Number of usable e-scooters in each cluster h_i^u (total of 320)	5–25	1–20	0–10
Vehicle capacity B	35	35	35
Unit cost of the driving distance β_0	1	1	1
Unit cost of picking up e-scooters β_1	0.1	0.1	0.1
Unit cost of remaining faulty e-scooters β_2	5	5	5
Unit cost of remaining low-battery e-scooters β_3	3	3	3
Unit cost of unmet demands β_4	2	2	2
Unit cost of excess e-scooters β_5	1	1	1

The optimization for rebalancing procedures was executed within the Spyder integrated development environment utilizing Python. For solving the Integer Linear Programming (ILP) rebalancing formulation, the ILP solver GLPK, integrated into the Pyomo Python library [71], was employed, consistent with the approach outlined in **Section 5.3.5**. The same ILP solver was also utilized for tackling the pickup and drop-off operations, as elaborated upon in **Section 5.3.6**. In parallel, the Ant Colony Optimization (ACO) algorithm was trained employing another Python library called Scikit-opt [70]. All of these operations were conducted within a Windows 10 environment operating on a 64-bit system architecture. The underlying hardware consisted of an Intel processor core i7-9750H CPU clocked at 2.60 GHz, complemented by 8.00 GB of RAM.

5.5.2 Sensitivity of the number of scenarios

Monte Carlo simulation was employed to generate demand uncertainty based on the predicted trip gap in **Section 5.4.2** and predicted variance and the distribution from **Section 5.4.3**. The number of samples to be generated can impact the optimization result. Sensitivity analysis was conducted to examine the impact of the number of simulated samples on optimal objective value. As we know, more samples likely provide more stable outputs, but it also increases the complexity of optimization problems. In other words, a higher number of scenarios creates more optimization parameters and constraint equations, resulting in a longer computational time to produce a feasible or global optimal solution. **Figure 5.10** shows the sensitivity analysis of the number of scenarios ranging from 50 to 1000, while three optimization trials were performed for each problem, 15-, 30-, and 60-cluster problems. In this case, the ILP solver could provide a feasible solution within the limited time (see **Table 5.4**) for 15- and 30-cluster problems, but this solver could produce a feasible solution only up to 500 scenarios for 60-cluster problems. Moreover, we observed a high variation of feasible solutions for 60-cluster problems for scenarios 800, 900, and 1000. We could also observe that the variation of optimal values is higher for a greater number of cluster problems. Overall, there is no significant pattern for the number of scenarios more than 100. Therefore, Monte Carlo simulation was conducted to generate 100 scenarios, whereas the benchmark cases using daily or weekly historical trip data had approximately 180 and 26 scenarios, respectively.

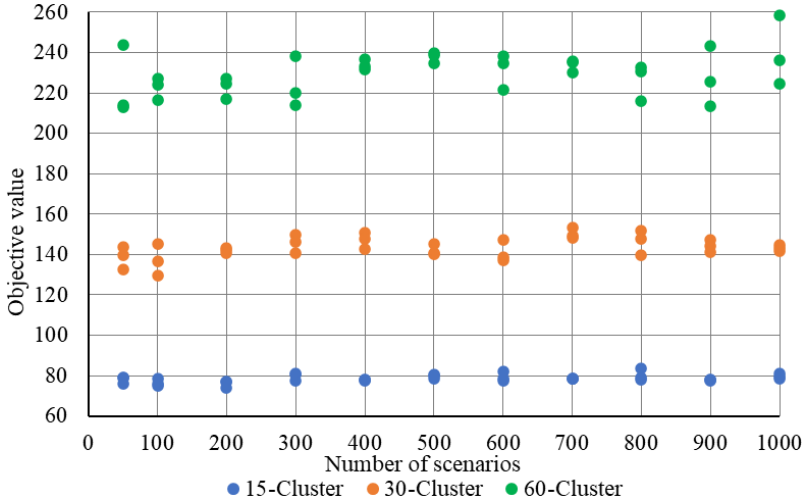

Figure 5.10 Sensitivity analysis on the number of scenarios

5.5.3 15-cluster problem

As detailed in **Table 5.4**, the computational timeframe for the balancing optimization was subject to constraints. Consequently, to strike a balance between exploration and exploitation within the hybrid ACO–ILP algorithm, two parameters from the ACO framework were employed: population size and the number of iterations. Notably, a larger population necessitates fewer iterations compared to a smaller population to successfully accomplish an optimization task within the designated computational limits. Illustrated in **Figure 5.11**, the left portion of the graph portrays the interplay between population size and the number of iterations, while the right segment showcases the convergence trajectory of the optimal trial (with population size set at 65 and the number of iterations at 25) for the rebalancing challenge pertaining to the 15-cluster problem. The benchmark instances, characterized by actual, historical daily, and historical weekly net demand, featured varying number of scenarios. Correspondingly, the population size was adjusted, either amplified or diminished, to accommodate these distinct scenarios.

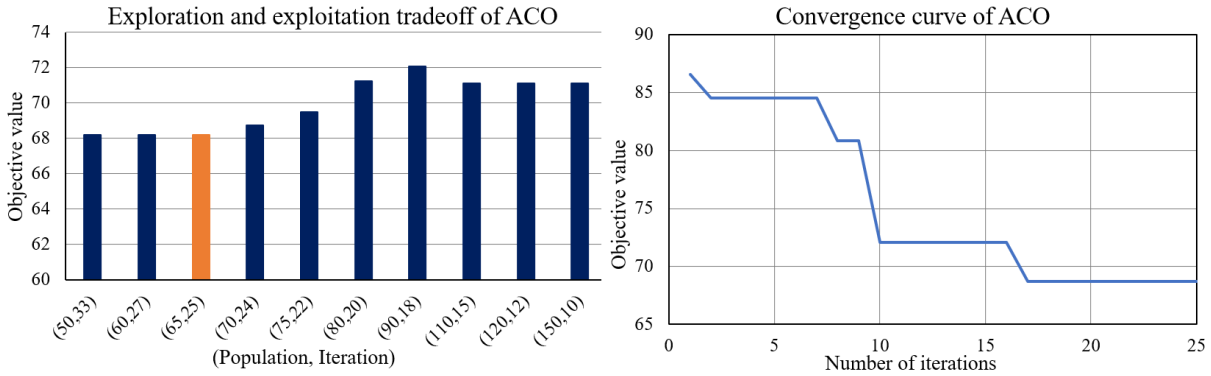

Figure 5.11 Exploration and exploitation tradeoff of ant colony optimization (*left*) and the convergence curve (*right*) for 15-cluster problems

The optimal parameters derived from the ACO methodology were subsequently utilized to optimize other instances of the 15-cluster rebalancing problem. **Figure 5.12** portrays the outcome of the optimal rebalancing process for one of the instances within the 15-cluster problem. Given the substantial penalty costs associated with the remaining defective and low-battery e-scooters, no remainder was left after the rebalancing operation. In this specific scenario, the rebalancing vehicle traversed multiple demand clusters, collecting low-battery e-scooters en route and subsequently depositing them at charging stations. During the initial nodes, the rebalancing vehicle retrieved 13 operational e-scooters from the depot, proceeded to collect 22 low-battery e-scooters, which were then delivered to charging station 2. Upon reaching cluster 6, five operational e-scooters were dropped off—this step might have been superfluous if demand uncertainty had been disregarded. Subsequently, the remainder (eight operational e-scooters) along with an additional ten operational e-scooters picked up at cluster 13 were transported to cluster 2, characterized by significant trip-gap variation. At cluster 2, the accumulated low-battery e-scooters were unloaded at charging station 1. Following this, nine operational e-scooters were collected and redeployed to either cluster 9 (six e-scooters) or cluster 5 (three e-scooters). Ultimately, from clusters 4 and 10, five and two operational e-scooters, respectively, were relocated to cluster 1, presumably resulting in higher anticipated

revenue. Subsequently, the amassed defective and low-battery e-scooters on the vehicle up to that point were returned to the depot for necessary repairs and recharging. This outcome underscores that the strategy managed to minimize the projected unmet demand within the confines of several constraints (such as vehicle capacity, driving distance, and the available number of operational e-scooters). This was achieved by redistributing operational e-scooters from locations characterized by an excess number of e-scooters or a low anticipated demand to areas where the expected demand was higher.

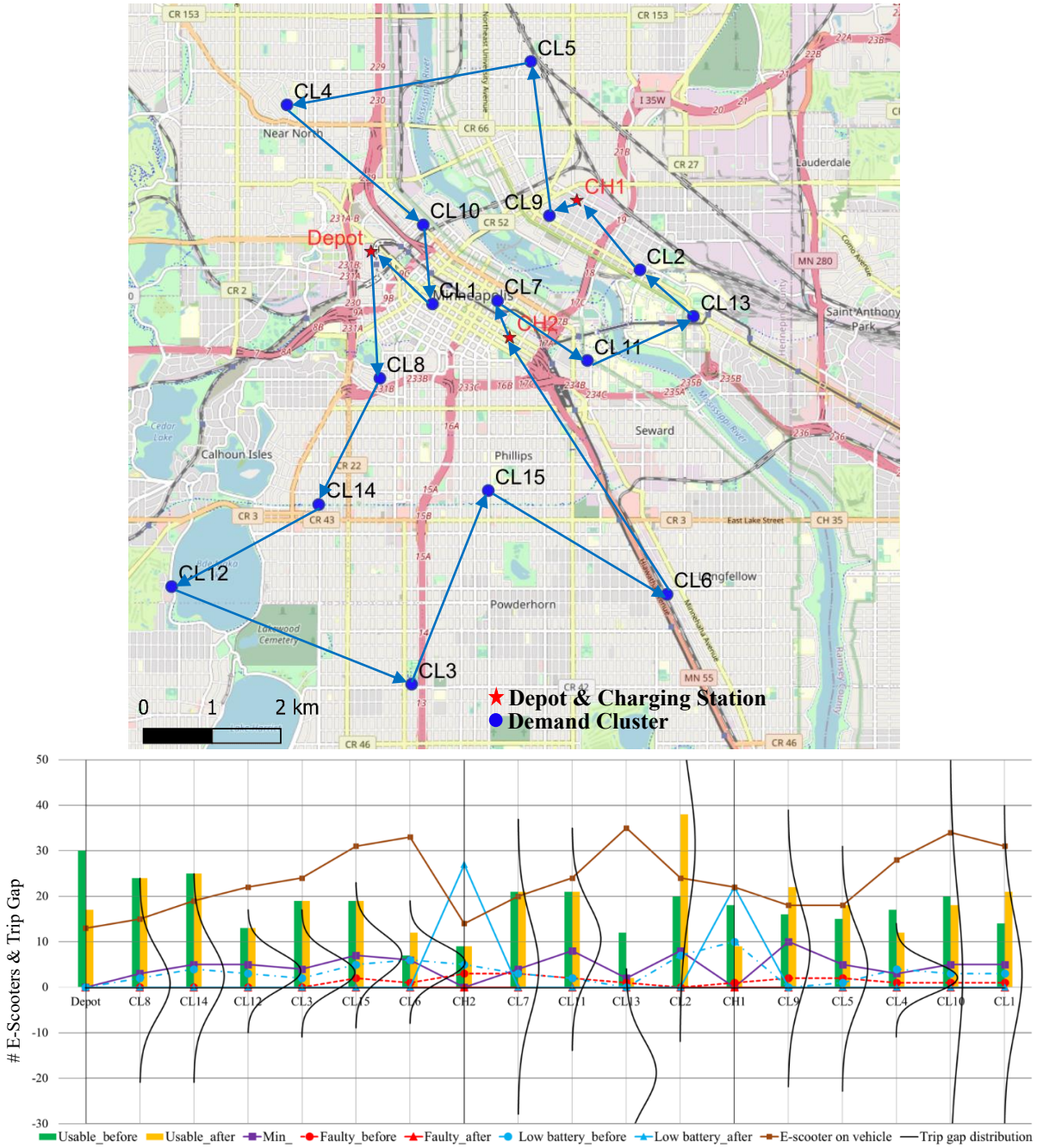

Figure 5.12 (*top*) Optimal route sequence of an instance in the 15-cluster problem and (*bottom*) its optimal pickup and drop-off results (CH: charging station, CL: cluster of trips)

In this scenario, the ILP solver demonstrated the capability to generate feasible solutions across all 30 instances and four distinct problem types (Actual, Sampling, Daily, and Weekly) within the 15-cluster problem. Depicted in **Figure 5.14** (*left*), the average of the optimal objective values from all 30 instances for the 15-cluster problem is presented. Given the relatively modest scale of this problem, the ILP solver "GLPK" yielded superior optimal objective values in contrast to the hybrid ACO-ILP approach. Analyzing the ILP numerical results, it's evident that the average driving distance remains consistent across the four cases (approximately 50 km); however, these cases exhibit variations in expected penalty costs. Specifically, the average optimal objective values for the four scenarios—Actual, Sampling, Daily, and Weekly—are recorded as 65 USD, 66.66 USD, 71.18 USD, and 68.85 USD, respectively. In essence, having perfect information resulted in the lowest penalty costs, while the sampling strategy demonstrated the ability to decrease penalty costs, along with introducing advantages in terms of temporal flexibility, as compared to the baseline daily and weekly scenarios. The sampling approach managed to lower penalty costs by approximately 6.35% and 3.18% when contrasted with the daily and weekly cases. The sampling approach could reduce penalty costs by roughly 6.35% and 3.18% compared to daily and weekly cases. With respect to penalty costs, excluding driving distance, the proposed methodology led to a reduction of penalty costs by 19.87% and 9.93% in comparison to these two baseline scenarios.

5.5.4 30-cluster problem

Figure 5.13 (*left*) shows the tradeoff between exploring and exploiting ACO's parameters, population size, and number of iterations for the 30-cluster problem. The optimal values of these two parameters were 90 and 19, respectively, while its convergence curve is shown in **Figure 5.13** (*right*). Similar to the problem above, the ILP solver can provide a feasible solution for all instances within the limited computational time, and it achieves better optimal solutions compared to the ACO-ILP algorithm. Based on optimal results from the ILP solver, as in **Figure 5.14**, the driving distance of the actual trip gaps is 80 km, while that of the other three cases is around 92 km. Moreover, the average objective value of the actual case is relatively low compared to the other three cases representing the necessity of perfect information or accurately forecasted demand. In this case, the combination of prediction demand variance could reduce the overall penalty costs by 8.25% and 2.75% compared to the implementation of historical actual daily and weekly trip data. If we exclude the driving distance, the percentage of reduction by sampling method is 22.28% and 13.55%, respectively.

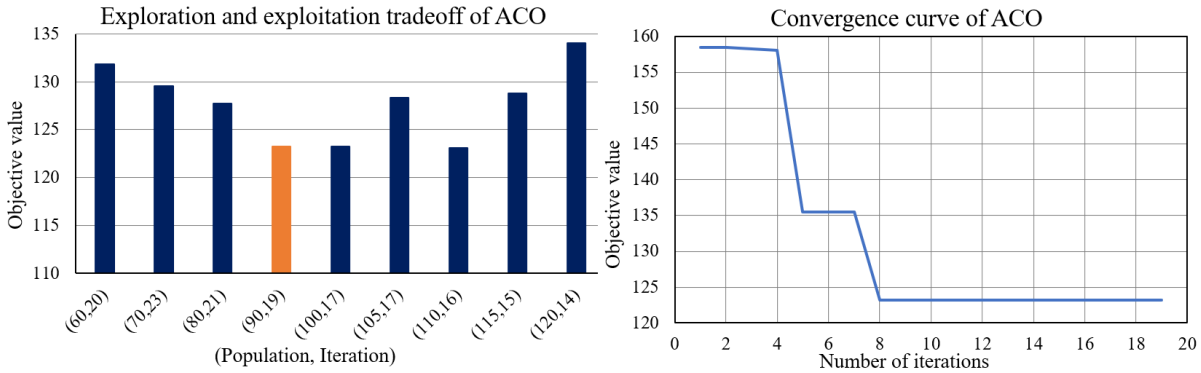

Figure 5.13 Exploration and exploitation tradeoff of ant colony optimization (*left*) and the convergence curve (*right*) for 30-cluster problems

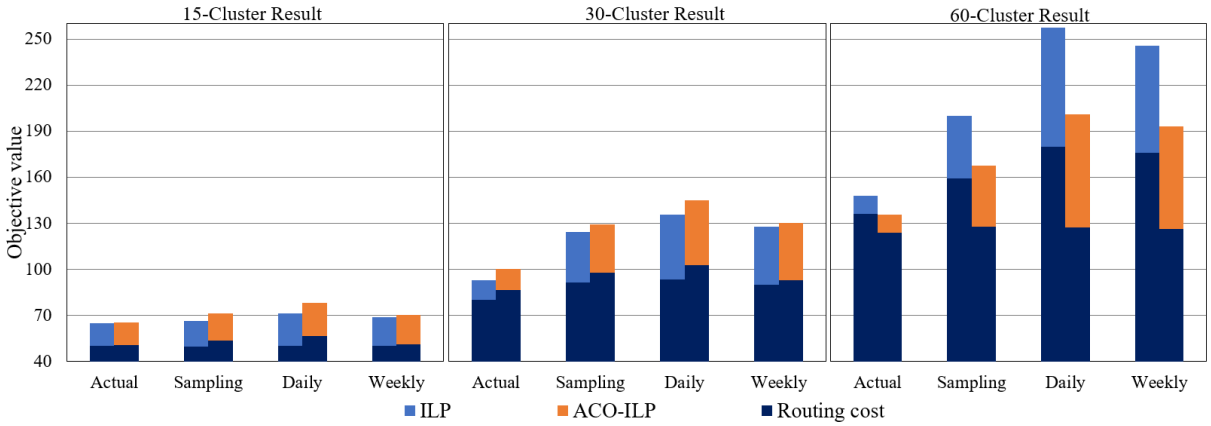

Figure 5.14 Average objective value for 30 random instances for 15-, 30-, and 60-cluster problems

5.5.5 60-cluster problem

Figure 5.15 (*left*) shows the balance of exploration and exploitation of ACO while the optimal values of population size and number of iterations are 7 and 160, respectively. The convergence curve of this case is shown in **Figure 5.15** (*right*), whereas the optimal objective value is 180.65. Unlike the previous two cases, ACO-ILP performs better than ILP solver, which cannot provide a feasible solution for some instances of historical daily (70%) and historical weekly (25%) trip gaps. There were approximately 200 and 28 scenarios for historical daily and historical weekly trip gaps, respectively, whereas the Monte Carlo sampling had 100 scenarios. Our result for Monte Carlo sampling showed that the ILP solver could provide a feasible solution for as many as 500 scenarios within the limited computational time. Hence, the distribution and variance of the trip gaps rather than just the number of scenarios contributed to the complexity of the optimization problem, which prevented the ILP solver from reaching a feasible solution. To obtain feasible results for all instances of historical daily and historical weekly trip gaps, the computational time was iteratively increased by 30 and 15 minutes, respectively. On average, the computational time required to reach a feasible solution in these two cases was 120 and 72 minutes, respectively.

As shown in **Figure 5.14**, the pattern of the average optimal objective values is similar, i.e., the Actual case has the lowest penalty costs, and the Sampling approach has lower penalty costs compared to the other two baselines. The driving distances of 60-cluster problems are around 123.88 km, 127.57 km, 127.27 km, and 126.24 km for Actual, Sampling, Daily, and Weekly cases, respectively. As shown in Figure 6.11, the average penalty costs of these four cases are 135.74 USD, 167.33 USD, 200.82 USD, and 192.94 USD, respectively. This means the sampling approach based on the forecasted trip gap and variance can reduce the overall penalty costs of 16.68% and 13.27% compared to the baselines. In other words, the reduction excluding driving distance achieved by the Sampling approach is 47.95% and 41.77%, respectively, compared to these two benchmark cases.

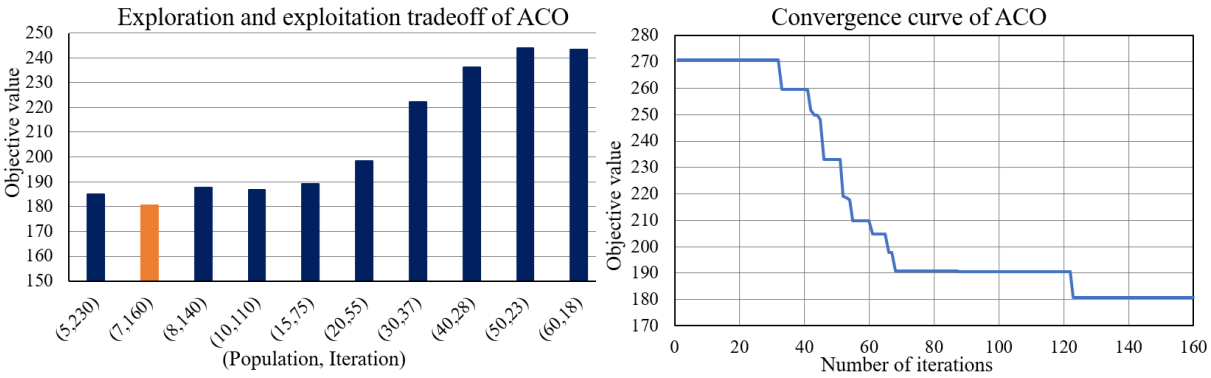

Figure 5.15 Exploration and exploitation tradeoff of ant colony optimization (*left*) and the convergence curve (*right*) for 60-cluster problems

5.5.6 Discussion

Figure 5.14 illustrates the average objective values for all three types of clusters, encompassing 30 random instances, while accounting for routing and penalty costs. This graphical representation reveals that regardless of the number of clusters, the perfect information (actual trip gap) corresponded to the lowest penalty cost. In contrast, the Sampling, Daily, and Weekly cases incurred higher penalty costs, roughly 20%, 35%, and 29% higher than the Actual case, respectively. This underscores the potential for further reduction in demand uncertainty, particularly through enhancements in demand and variance predictions. Furthermore, the graph indicates that for smaller-sized problems (15- and 30-cluster scenarios), the ILP solver "GLPK" outperformed, whereas for larger-sized problems (60-cluster scenario), the hybrid ACO-ILP algorithm exhibited better performance. It's noteworthy that Assumption 1 necessitates a higher quantity of clusters, which in turn results in elongated driving distances for rebalancing vehicles and amplified penalty costs due to heightened demand uncertainty. Moreover, while the ILP solver requires prolonged training times to generate feasible solutions for larger-scale problems involving an increased number of clusters, our findings highlight that the sampling approach utilizing the Monte Carlo method, founded on predicted demand and variance, managed to curtail demand uncertainty. This in turn expedited the ILP solver's ability to attain feasible solutions more swiftly than relying on historical daily and weekly net demands.

During a pilot initiative conducted in 2019, the city of Minneapolis sanctioned the deployment of up to 2,000 shared e-scooters distributed among various operators, including Lime, Lyft, JUMP, and Spin. For this present study, the total count of e-scooters was established at 400, a number roughly equivalent to those deployed by each individual operator. In a parallel context involving dockless shared bikes, a study by Hua et al. [72] identified that by providing merely 14.5% of the original fleet, it was possible to satisfy 96.8% of the trip demand. Adhering to a constraint of 400 e-scooters (about 20% of the permissible fleet), we managed to achieve a service level of approximately 96.6% through hourly rebalancing efforts. From an operational perspective, this accomplishment signifies the potential to curtail the deployed fleet size, especially if all operators collaboratively engaged in unified rebalancing planning or minimized territorial overlap by incorporating geofences. Taking environmental factors into account, frequent rebalancing actions, particularly when involving vehicles with internal combustion

engines, are ill-advised due to their substantial contribution to emissions (up to 50% according to [19]). Given the challenges in achieving complete operational harmonization among operators, a viable strategy involves judiciously segregating deployment zones, thereby minimizing overlaps and optimizing intrazonal trips. In such a framework, rebalancing efforts should be confined to a few instances daily, ideally conducted using electric rebalancing vehicles, while also minimizing the number of clusters to be visited.

Based on the empirical findings discussed earlier, the calculated driving distances approximately amount to 50 km, 90 km, and 120 km for the three distinct problem sizes: 15-cluster, 30-cluster, and 60-cluster scenarios, respectively. This indicates that the rebalancing vehicle would be unable to complete the entire rebalancing operation within the stipulated 1-hour timeframe, as demonstrated in the case study. Given this context, the practicality of our proposed framework might be better suited for extended rebalancing durations, such as 2 to 3 hours, or it may necessitate adjustments to meet specific objectives, including driving distance reduction or service level enhancements. Among potential modifications, the initial option involves enabling the rebalancing algorithm to bypass nodes that are deemed unnecessary or have already been balanced. The second modification suggests augmenting the number of rebalancing vehicles, particularly by implementing multiple smaller rebalancing vehicles. However, the third option is particularly advisable. This involves eliminating balanced demand clusters prior to the rebalancing optimization and execution phase. By doing so, both computational time and driving distance can be curtailed, owing to the reduction in the overall number of nodes. Within this framework, operators can concentrate their efforts solely on nodes displaying significant deviations from the desired supply level—summing predicted trip gaps, predicted trip gap variations with service level parameters, and safety stock considerations. When the quantity of nodes is relatively modest (below 15 nodes), rebalancing optimization can be executed efficiently through the employment of an ILP solver. For more extensive optimization challenges, an ACO-ILP strategy, especially when supplemented with parallel computing capabilities, becomes more apt. The viability of the proposed framework is further accentuated if operators choose to segregate operational territories, generating smaller rebalancing regions for streamlined management.

5.6 Conclusion

In summation, dockless shared e-scooters exhibit considerable potential as a solution for enhancing compact urban mobility, especially in addressing the challenges of the first-mile-last-mile dilemma, parking scarcities, and offering an alternative transportation mode. However, the nuanced characteristics of trips, vehicles, and regulatory considerations render the short-term operational planning for this mode notably more intricate compared to its closest counterpart, shared bikes. Notably, the pickup and drop-off patterns of shared e-scooters deviate from conventional Poisson distributions. This divergence implies that standard operational planning strategies—often utilized under conditions of demand uncertainty for shared bikes, such as the application of Markov chains or queue theory—might not be directly applicable to shared e-scooters.

In light of these complexities, this study introduces a novel framework tailored for the short-term rebalancing of shared e-scooters. This framework hinges on Monte Carlo simulation,

leveraging predicted demand and variance. Employing a GB model, the hourly trip gap of shared e-scooters is forecasted, while the residuals of this model are refined via SGARCH. This dual-model integration successfully mitigates demand uncertainty, as evidenced by the reduction in RMSE and average STD.

Furthermore, a novel ILP rebalancing formulation is devised, encompassing demand uncertainty, malfunctioning e-scooters, and low-battery instances. Given the NP-hard nature of this problem, an ILP solver is deployed to tackle smaller-scale scenarios, while the solution to larger-scale problems is achieved through a hybrid ACO–ILP algorithm. Numerical results, derived from real-world data in the context of Minneapolis, MN, and focused on the 60-cluster scenario, indicate that the application of our framework leads to a reduction of the operational burden by 20.01% and 15.30% relative to benchmark practices relying on historical daily and weekly demands, respectively. Moreover, while the ILP solver at times struggled to furnish a feasible solution within computational time limits for these benchmark cases, this challenge was circumvented by our simulated demand approach. Importantly, the ILP solver necessitated a lengthier duration to yield feasible solutions for baseline problems compared to the efficiency demonstrated by our Monte Carlo sampling strategy.

References

[1] T. Benarbia, K. Labadi, A. M. Darcherif, J. P. Barbot and A. Omari, "Real-time inventory control and rebalancing in bike-sharing systems by using a stochastic Petri net model," in *3rd International Conference on Systems and Control*, 2013, pp. 583-589, doi: https://doi.org/10.1109/ICoSC.2013.6750920.

[2] N. Saum and M. Piantanakulchai, "A Review on an Emerging New Mode of Transport: The Shared Dockless Electric Scooter," presented at the 13th International Conference of the Eastern Asia Society for Transportation Studies (EASTS 2019), Colombo, 2019. Available: http://www.easts.info/on-line/proceedings/vol.12/head.htm.

[3] NACTO. *Shared Micromobility in the U.S.: 2019*. Available: https://nacto.org/shared-micromobility-2019/.

[4] A. Hosseinzadeh, A. Karimpour and R. Kluger, "Factors influencing shared micromobility services: An analysis of e-scooters and bikeshare," *Transportation Research Part D: Transport and Environment,* vol. 100, pp. 103047, 2021, doi: https://doi.org/10.1016/j.trd.2021.103047.

[5] K. Wang, X. Qian, D. T. Fitch, Y. Lee, J. Malik and G. Circella, "What travel modes do shared e-scooters displace? A review of recent research findings," *Transport Reviews,* pp. 1-27, 2022, doi: https://doi.org/10.1080/01441647.2021.2015639.

[6] W. Riggs, M. Kawashima and D. Batstone, "Exploring best practice for municipal e-scooter policy in the United States," *Transportation Research Part A: Policy and Practice,* vol. 151, pp. 18-27, 2021, doi: https://doi.org/10.1016/j.tra.2021.06.025.

[7] A.-H. Kirstin, B. Brandon, O. N. Riley and S. Smith C, "Governing micro-mobility: A nationwide assessment of electric scooter regulations," presented at the Transportation Research Board 98th Annual Meeting, Washington D.C., 2019. doi: https://trid.trb.org/view/1572811.

[8] K. Button, H. Frye and D. Reaves, "Economic regulation and E-scooter networks in the USA," *Research in Transportation Economics,* vol. 84, pp. 100973, 2020, doi: https://doi.org/10.1016/j.retrec.2020.100973.

[9] A. Brown, "Micromobility, Macro Goals: Aligning scooter parking policy with broader city objectives," *Transportation Research Interdisciplinary Perspectives,* vol. 12, pp. 100508, 2021, doi: https://doi.org/10.1016/j.trip.2021.100508.

[10] A. Pashkevich, T. E. Burghardt, S. Puławska-Obiedowska and M. Šucha, "Visual attention and speeds of pedestrians, cyclists, and electric scooter riders when using shared road – a field eye tracker experiment," *Case Studies on Transport Policy,* vol. 10, no. 1, pp. 549-558, 2022, doi: https://doi.org/10.1016/j.cstp.2022.01.015.

[11] S. Sareen, D. Remme and H. Haarstad, "E-scooter regulation: The micro-politics of market-making for micro-mobility in Bergen," *Environmental Innovation and Societal Transitions,* vol. 40, pp. 461-473, 2021, doi: https://doi.org/10.1016/j.eist.2021.10.009.

[12] A. Brown, N. J. Klein, C. Thigpen and N. Williams, "Impeding access: The frequency and characteristics of improper scooter, bike, and car parking," *Transportation Research Interdisciplinary Perspectives,* vol. 4, pp. 100099, 2020, doi: https://doi.org/10.1016/j.trip.2020.100099.

[13] H. Li, Z. Yuan, T. Novack, W. Huang and A. Zipf, "Understanding spatiotemporal trip purposes of urban micro-mobility from the lens of dockless e-scooter sharing," *Computers, Environment and Urban Systems,* vol. 96, pp. 101848, 2022, doi: https://doi.org/10.1016/j.compenvurbsys.2022.101848.

[14] C. S. Smith and J. P. Schwieterman, "E-scooter scenarios: evaluating the potential mobility benefits of shared dockless scooters in Chicago," Chaddick Institute for Metropolitan Development, Depaul University. 2018.

[15] O. Caspi, M. J. Smart and R. B. Noland, "Spatial associations of dockless shared e-scooter usage," *Transportation Research Part D: Transport and Environment,* vol. 86, pp. 102396, 2020, doi: https://doi.org/10.1016/j.trd.2020.102396.

[16] R. Zhu, X. Zhang, D. Kondor, P. Santi and C. Ratti, "Understanding spatio-temporal heterogeneity of bike-sharing and scooter-sharing mobility," *Computers, Environment and Urban Systems,* vol. 81, pp. 101483, 2020, doi: https://doi.org/10.1016/j.compenvurbsys.2020.101483.

[17] H. Younes, Z. Zou, J. Wu and G. Baiocchi, "Comparing the Temporal Determinants of Dockless Scooter-share and Station-based Bike-share in Washington, D.C," *Transportation Research Part A: Policy and Practice,* vol. 134, pp. 308-320, 2020, doi: https://doi.org/10.1016/j.tra.2020.02.021.

[18] G. McKenzie, "Urban mobility in the sharing economy: A spatiotemporal comparison of shared mobility services," *Computers, Environment and Urban Systems,* vol. 79, pp. 101418, 2020, doi: https://doi.org/10.1016/j.compenvurbsys.2019.101418.

[19] A. de Bortoli and Z. Christoforou, "Consequential LCA for territorial and multimodal transportation policies: method and application to the free-floating e-scooter disruption in Paris," *Journal of Cleaner Production,* vol. 273, pp. 122898, 2020, doi: https://doi.org/10.1016/j.jclepro.2020.122898.

[20] H. Moreau, L. de Jamblinne de Meux, V. Zeller, P. D'Ans, C. Ruwet and W. M. J. Achten, "Dockless E-Scooter: A Green Solution for Mobility? Comparative Case Study between

Dockless E-Scooters, Displaced Transport, and Personal E-Scooters," *Sustainability,* vol. 12, no. 5, pp. 1803, 2020, doi: https://doi.org/doi:10.3390/su12051803.

[21] H. Peng, Y. Nishiyama and K. Sezaki, "Assessing environmental benefits from shared micromobility systems using machine learning algorithms and Monte Carlo simulation," *Sustainable Cities and Society,* vol. 87, pp. 104207, 2022, doi: https://doi.org/10.1016/j.scs.2022.104207.

[22] S. Severengiz, S. Finke, N. Schelte and N. Wendt, "Life Cycle Assessment on the Mobility Service E-Scooter Sharing," presented at the 2020 IEEE European Technology and Engineering Management Summit (E-TEMS), Dortmund, 2020. doi: https://doi.org/10.1109/E-TEMS46250.2020.9111817.

[23] M. Javadinasr, S. Asgharpour, E. Rahimi, P. Choobchian, A. K. Mohammadian and J. Auld, "Eliciting attitudinal factors affecting the continuance use of E-scooters: An empirical study in Chicago," *Transportation Research Part F: Traffic Psychology and Behaviour,* vol. 87, pp. 87-101, 2022, doi: https://doi.org/10.1016/j.trf.2022.03.019.

[24] R. G. Öztaş Karlı, H. Karlı and H. S. Çelikyay, "Investigating the acceptance of shared e-scooters: Empirical evidence from Turkey," *Case Studies on Transport Policy,* vol. 10, no. 2, pp. 1058-1068, 2022, doi: https://doi.org/10.1016/j.cstp.2022.03.018.

[25] M. Abouelela, C. Al Haddad and C. Antoniou, "Are young users willing to shift from carsharing to scooter–sharing?," *Transportation Research Part D: Transport and Environment,* vol. 95, pp. 102821, 2021, doi: https://doi.org/10.1016/j.trd.2021.102821.

[26] H. Fitt and A. Curl, "The early days of shared micromobility: A social practices approach," *Journal of Transport Geography,* vol. 86, pp. 102779, 2020, doi: https://doi.org/10.1016/j.jtrangeo.2020.102779.

[27] S. He and K. G. Shin, "Dynamic flow distribution prediction for urban dockless e-scooter sharing reconfiguration," presented at the WWW '20: Proceedings of The Web Conference 2020, Taipei, 2020. doi: https://doi.org/10.1145/3366423.3380101.

[28] S. He and K. G. Shin, "Distribution Prediction for Reconfiguring Urban Dockless E-Scooter Sharing Systems," *IEEE Transactions on Knowledge and Data Engineering,* vol. 34, no. 12, pp. 5722-5740, 2022, doi: https://doi.org/10.1109/TKDE.2021.3062074.

[29] N. Saum, S. Sugiura and M. Piantanakulchai, "Short-Term Demand and Volatility Prediction of Shared Micro-Mobility: a case study of e-scooter in Thammasat University," presented at the 2020 Forum on Integrated and Sustainable Transportation Systems (FISTS), Delft, 2020. doi: https://doi.org/10.1109/FISTS46898.2020.9264852.

[30] S. W. Ham, J.-H. Cho, S. Park and D.-K. Kim, "Spatiotemporal Demand Prediction Model for E-Scooter Sharing Services with Latent Feature and Deep Learning," *Transportation Research Record,* vol. 2675, no. 11, pp. 34-43, 2021, doi: https://doi.org/10.1177/03611981211003896.

[31] S. Phithakkitnukooon, K. Patanukhom and M. G. Demissie, "Predicting Spatiotemporal Demand of Dockless E-Scooter Sharing Services with a Masked Fully Convolutional Network," *ISPRS International Journal of Geo-Information,* vol. 10, no. 11, pp. 773, 2021.

[32] Y. Xu, X. Zhao, X. Zhang and M. Paliwal, "Real-Time Forecasting of Dockless Scooter-Sharing Demand: A Spatio-Temporal Multi-Graph Transformer Approach," 2021.

[33] S. Kim, S. Choo, G. Lee and S. Kim, "Predicting Demand for Shared E-Scooter Using Community Structure and Deep Learning Method," *Sustainability,* vol. 14, no. 5, pp. 2564, 2022.

[34] P. W. Khan, S.-J. Park, S.-J. Lee and Y.-C. Byun, "Electric Kickboard Demand Prediction in Spatiotemporal Dimension Using Clustering-Aided Bagging Regressor," *Journal of Advanced Transportation,* vol. 2022, pp. 8062932, 2022, doi: https://doi.org/10.1155/2022/8062932.

[35] M. Masoud, M. Elhenawy, M. H. Almannaa, S. Q. Liu, S. Glaser and A. Rakotonirainy, "Heuristic Approaches to Solve E-Scooter Assignment Problem," *IEEE Access,* vol. 7, pp. 175093-175105, 2019, doi: https://doi.org/10.1109/ACCESS.2019.2957303.

[36] A. Ciociola, M. Cocca, D. Giordano, L. Vassio and M. Mellia, "E-Scooter Sharing: Leveraging Open Data for System Design," in *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, 2020, pp. 1-8, doi: 10.1109/DS-RT50469.2020.9213514.

[37] L. Tolomei, S. Fiorini, A. Ciociola, L. Vassio, D. Giordano and M. Mellia, "Benefits of Relocation on E-scooter Sharing - a Data-Informed Approach," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3170-3175, doi: 10.1109/ITSC48978.2021.9564809.

[38] J. Osorio, C. Lei and Y. Ouyang, "Optimal rebalancing and on-board charging of shared electric scooters," *Transportation Research Part B: Methodological,* vol. 147, pp. 197-219, 2021, doi: https://doi.org/10.1016/j.trb.2021.03.009.

[39] A. M. Fathabad, X. Li, J. Cheng and Y.-J. Wu, "Data-Driven Optimization for E-Scooter System Design," (in English), Tech Report 2022.

[40] G. Losapio, F. Minutoli, V. Mascardi and A. Ferrando, "Smart balancing of E-scooter sharing systems via deep reinforcement learning," in *22nd Workshop "From Objects to Agents"*, Bologna, Italy, 2022, vol. 2963, pp. 83–97: CEUR-WS.org.

[41] O. Altintasi and S. Yalcinkaya, "Siting charging stations and identifying safe and convenient routes for environmentally sustainable e-scooter systems," *Sustainable Cities and Society,* vol. 84, p. 104020, 2022, doi: https://doi.org/10.1016/j.scs.2022.104020.

[42] G. McKenzie, "Spatiotemporal comparative analysis of scooter-share and bike-share usage patterns in Washington, D.C," *Journal of Transport Geography,* vol. 78, pp. 19-28, 2019, doi: https://doi.org/10.1016/j.jtrangeo.2019.05.007.

[43] A. Martínez-Navarro, V. A. Cloquell-Ballester and S. Segui-Chilet, "Photovoltaic Electric Scooter Charger Dock for the Development of Sustainable Mobility in Urban Environments," *IEEE Access,* vol. 8, pp. 169486-169495, 2020, doi: https://doi.org/10.1109/ACCESS.2020.3023881.

[44] C. S. Shui and W. Y. Szeto, "A review of bicycle-sharing service planning problems," *Transportation Research Part C: Emerging Technologies,* vol. 117, pp. 102648, 2020, doi: https://doi.org/10.1016/j.trc.2020.102648.

[45] X. Chang, J. Wu, H. Sun, G. H. d. A. Correia and J. Chen, "Relocating operational and damaged bikes in free-floating systems: A data-driven modeling framework for level of service enhancement," *Transportation Research Part A: Policy and Practice,* vol. 153, pp. 235-260, 2021, doi: https://doi.org/10.1016/j.tra.2021.09.010.

[46] M. Dell'Amico, E. Hadjicostantinou, M. Iori and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," *Omega,* vol. 45, pp. 7-19, 2014, doi: https://doi.org/10.1016/j.omega.2013.12.001.

[47] T. Raviv and O. Kolka, "Optimal inventory management of a bike-sharing station," *IIE Transactions,* vol. 45, no. 10, pp. 1077-1093, 2013, doi: https://doi.org/10.1080/0740817X.2013.770186.

[48] R. Alvarez-Valdes *et al.*, "Optimizing the level of service quality of a bike-sharing system," *Omega,* vol. 62, pp. 163-175, 2016, doi: https://doi.org/10.1016/j.omega.2015.09.007.

[49] E. O'Mahony, "Smarter Tools For (Citi)Bike Sharing," Computer Science, Cornell University, Cornell University, 2015.

[50] J. Schuijbroek, R. C. Hampshire and W. J. van Hoeve, "Inventory rebalancing and vehicle routing in bike sharing systems," *European Journal of Operational Research,* vol. 257, no. 3, pp. 992-1004, 2017, doi: https://doi.org/10.1016/j.ejor.2016.08.029.

[51] Y.-H. Seo, "A Dynamic Rebalancing Strategy in Public Bicycle Sharing Systems Based on Real-Time Dynamic Programming and Reinforcement Learning," Doctoral dissertation, Seoul National University, 2020.

[52] C.-C. Lu, "Robust Multi-period Fleet Allocation Models for Bike-Sharing Systems," *Networks and Spatial Economics,* vol. 16, no. 1, pp. 61-82, 2016, doi: https://doi.org/10.1007/s11067-013-9203-9.

[53] Y. Chen and Y. Liu, "Integrated Optimization of Planning and Operations for Shared Autonomous Electric Vehicle Systems," *Transportation Science,* 2022, doi: https://doi.org/10.1287/trsc.2022.1156.

[54] F. Maggioni, M. Cagnolari, L. Bertazzi and S. W. Wallace, "Stochastic optimization models for a bike-sharing problem with transshipment," *European Journal of Operational Research,* vol. 276, no. 1, pp. 272-283, 2019, doi: https://doi.org/10.1016/j.ejor.2018.12.031.

[55] S. Yan, C.-C. Lu and M.-H. Wang, "Stochastic fleet deployment models for public bicycle rental systems," *International Journal of Sustainable Transportation,* vol. 12, no. 1, pp. 39-52, 2018, doi: https://doi.org/10.1080/15568318.2017.1324586.

[56] M. Dell'Amico, M. Iori, S. Novellani and A. Subramanian, "The Bike sharing Rebalancing Problem with Stochastic Demands," *Transportation Research Part B: Methodological,* vol. 118, pp. 362-380, 2018, doi: https://doi.org/10.1016/j.trb.2018.10.015.

[57] R. Regue and W. Recker, "Proactive vehicle routing with inferred demand to solve the bikesharing rebalancing problem," *Transportation Research Part E: Logistics and Transportation Review,* vol. 72, pp. 192-209, 2014, doi: https://doi.org/10.1016/j.tre.2014.10.005.

[58] D. Huang, X. Chen, Z. Liu, C. Lyu, S. Wang and X. Chen, "A static bike repositioning model in a hub-and-spoke network framework," *Transportation Research Part E: Logistics and Transportation Review,* vol. 141, p. 102031, 2020.

[59] R. Guo *et al.*, "BikeNet: Accurate Bike Demand Prediction Using Graph Neural Networks for Station Rebalancing," presented at the 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation

(SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Leicester, 2019. doi: https://doi.org/10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00153.

[60] J.-H. Cho, Y.-H. Seo and D.-K. Kim, "Efficiency Comparison of Public Bike-Sharing Repositioning Strategies Based on Predicted Demand Patterns," *Transportation Research Record,* vol. 2675, no. 11, pp. 104-118, 2021, doi: https://doi.org/10.1177/03611981211016859.

[61] L. Yu, T. Feng, T. Li and L. Cheng, "Demand Prediction and Optimal Allocation of Shared Bikes Around Urban Rail Transit Stations," *Urban Rail Transit,* 2022, doi: https://doi.org/10.1007/s40864-022-00183-w.

[62] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics,* vol. 29, no. 5, pp. 1189-1232, 2001, doi: https://doi.org/10.1214/aos/1013203451.

[63] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.

[64] A. Oyedele *et al.*, "Deep learning and Boosted trees for injuries prediction in power infrastructure projects," *Applied Soft Computing,* vol. 110, pp. 107587, 2021, doi: https://doi.org/10.1016/j.asoc.2021.107587.

[65] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei and S.-H. Deng, "Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization," *Journal of Electronic Science and Technology,* vol. 17, no. 1, pp. 26-40, 2019, doi: https://doi.org/10.11989/JEST.1674-862X.80904120.

[66] T. Head, G. Louppe, H. Nahrstaedt, I. Shcherbatyi and M. Kumar, "Scikit-Optimize," Available: https://github.com/scikit-optimize

[67] StataCorp, *Stata: Time-Series Reference Manual Release 17* (Statistical Software). TX: StataCorp LLC: College Station, 2021.

[68] L. Di Gaspero, A. Rendl and T. Urli, "A Hybrid ACO+CP for Balancing Bicycle Sharing Systems," in *Hybrid Metaheuristics*, Berlin, Heidelberg, 2013, pp. 198-212: Springer Berlin Heidelberg.

[69] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine,* vol. 1, no. 4, pp. 28-39, 2006, doi: https://doi.org/10.1109/MCI.2006.329691.

[70] Guofei, Agrover, Ilikega, Zidong, Zhangxiao and e. al., "Scikit-opt," Available: https://github.com/guofei9987/scikit-opt

[71] W. E. Hart, J.-P. Watson and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in Python," *Mathematical Programming Computation,* vol. 3, no. 3, pp. 219, 2011, doi: https://doi.org/10.1007/s12532-011-0026-8.

[72] M. Hua, X. Chen, J. Chen and Y. Jiang, "Minimizing fleet size and improving vehicle allocation of shared mobility under future uncertainty: A case study of bike sharing," *Journal of Cleaner Production,* vol. 370, pp. 133434, 2022, doi: https://doi.org/10.1016/j.jclepro.2022.133434.

CHAPTER 6
6. CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

Shared e-scooter is a new dockless sharing micromobility, expanding across the globe for its exciting riding experience (ease of riding and parking), time- and cost-effectiveness for short-range trips (dealing with first- and last-mile problems), and environmentally friendly. However, this shared transportation mode faces many challenges in short-term operational planning, such as high demand volatility, intensive maintenance, short-service life, emission from rebalancing, and relevant regulations. Therefore, this study aims to explore historical ridership data to improve the short-term management planning for this shared micromobility. To achieve this objective, this study is divided into three main parts: supply planning design, hyperparameter optimization, and rebalancing planning. The research findings and recommendations from these three parts are summarized as follows.

6.1.1 Findings and recommendations from short-term supply planning

Several findings and recommendations can be drawn from the numerical results of short-term supply planning design, such as:

- Box Cox transformation can improve the demand prediction accuracy, specifically MAE, and remove the heteroscedasticity.
- Box Cox transformation is suitable for supply planning at a low service level (or lower confidence bound), while a ceiling value for supply level is necessary for a higher service level, especially when $\lambda < 0$ and $\lambda \to 0$.
- Box Cox transformation can deal with outliers or data with insufficient explanatory variables.
- Hyperparameter optimization is necessary for machine learning and deep learning models, otherwise it might have worse performance than statistical regression models (ex., SARIMAX).
- The residuals of machine learning and deep learning models are not white noise but heteroscedastic or conditional variance.
- Even deep learning models can have higher demand prediction accuracy, their supply level models might be worse than SARIMAX if their residuals are not properly examined. In other words, the demand uncertainty can be minimized by combining demand prediction and variance analysis.
- Daily variance can be an efficient variance model for supply level estimation, but it is not flexible for long-term trends, especially seasonal or annual trends.
- SGARCH is more temporally flexible than daily variance and is suitable for supply level estimation at a high service level (or upper confidence bound).
- Mean Oversupply metric can be used to evaluate the efficiency of supply-level estimation models, while previous studies used two or more metrics.
- Based on three real-world datasets, accounting for heteroscedasticity in supply planning can reduce the oversupply by 26.22% at 95% served demand.

6.1.2 Findings and recommendations from hyperparameter optimization

Several findings and recommendations can be drawn from the empirical results of hyperparameter optimization, such as:

- Compared with sequential-based algorithms (ex., TPE and BO), Iterative Decision Tree (IDT) has several advantages such as ease of parallel computation, no acquisition model, less intensive training of the surrogate model, better avoiding local optimal solutions, and support for multiple objectives.
- Compared with population-based algorithms (ex., GA), IDT has several advantages, such as keeping historical evaluated points, avoiding training repetitive candidates, providing feature importance, and faster convergence.
- IDT-R performs better than IDT-E, which is unsuitable for optimizing problems with more than three parameters.
- TPE easily falls into local optimal points, so it is not suitable for optimization problems with multiple locally optimal solutions like optimization of nonconvex functions and HPO of machine learning models. However, TPE performs well in optimizing the hyperparameters of deep learning models.
- IDT-R has stable performance across optimization problems in searching for a near-global optimal solution. Moreover, IDT-R has comparable performance with TPE for HPO of deep learning models while outperforming it and other baseline algorithms in searching optimal solutions for nonconvex functions and HPO of machine learning models.
- Random Forest (RF) can predict multiple output problems (ex., predict spatial demands), but its performance reduces for a greater number of spatial demands, especially in comparison with GRUs. Moreover, it requires more extended training than GRUs if the number of data and outputs increases.

6.1.3 Findings and recommendations from rebalancing planning

The lesson learned from the two sections above was used to construct an efficient framework for rebalancing shared e-scooters. In addition, from the numerical results evaluated using ridership data in Minneapolis MN, we achieved several more findings and recommendations as follows:

- As many external factors influence shared e-scooter ridership, the arrival and departure trips do not follow the typical Poisson distribution, which is the most important assumption for Markov Chain used for operational planning in previous studies (especially shared bikes).
- Sample Average Approach (SAA) based on Monte Carlo method can be employed to generate the demand uncertainty, which can be minimized through demand prediction by robust machine or deep learning models and the estimation of variance and distribution by SGARCH.
- Integer Linear Programming (ILP) solver “GLPK” is suitable for solving small-size rebalancing optimization problems (ex., 15- and 30-cluster problems), but hybrid Ant Colony Algorithm with ILP (ACO-ILP) is more suitable for large-size problems (ex., 60-cluster problems).

- Besides the number of scenarios, the distribution and variance of the trip gaps also contribute to the complexity of the optimization problem, which prevents the ILP solver from reaching a feasible solution.
- A greater number of demand clusters requires longer computational time and increase the demand uncertainty and driving distance of rebalancing vehicle.
- From the most practical case (60-cluster problems), the proposed framework (SAA with predicted trip gaps and variances) reduces the operational burden by around 16.68% and 13.27% compared to historical daily and weekly demands.
- The proposed rebalancing planning is suitable for periodic rebalancing or distribution, so it can be integrated with other strategies such as (1) real-time rebalancing to respond to the instant demand spikes and (2) customer incentivizing to minimize the rebalancing operation.
- From the emission viewpoint, there should be unified rebalancing planning among operators or proper separation of e-scooter deployment zones (ex., minimize inter-zonal trips).

6.2 Recommendations for future study

There are a couple of directions for future studies such as:

- SGARCH model is strongly susceptible to significant prediction errors, so future studies may employ or adopt other variance prediction models that can deal with this limitation.
- Including the conditional variance in demand prediction models is also a promising technique for future works as it possibly improves the prediction accuracy and promptly provides the expected demand and variance.
- The performance of the proposed algorithm (IDT-R) can be compared in future works under the constraint of computational time, especially in the case of parallelization and multiple objectives.
- IDT-R can achieve higher efficiency if we combine it with the multi-fidelity approach, as it could reduce the training time of some poor performance configurations of deep learning.
- IDT-R gives one additional piece of information, feature importance. Further research may focus on applying this metric, ex., narrowing the search space.
- Another direction of the future study related to IDT-R is examining it on varied and conditional search spaces.
- In terms of rebalancing planning, future studies could focus on dynamic rebalancing, including rebalancing planning a few steps ahead or in real-time.
- Moreover, for more practical applications, future research could implement parallel computing to reduce the computational time of the rebalancing optimization.
- Future studies could also focus on sensitivity analysis of the unit costs of the rebalancing penalty terms (e.g., unmet demand) to assess its impact on operational costs, service level, or driving distance.
- Future studies may consider the constraints of driving distance in the operational planning optimization as discussed in **Section 5.5.6**, specifically the remedy recommendations such as using multiple small vehicles, allowing the rebalancing vehicle to skip some nodes, and removing the rebalanced clusters.

Appendix: Python Code

```
# Import All Necessary Libraries
import pandas as pd
import numpy as np
import os
import xgboost as xgb
import tensorflow as tf
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout, SimpleRNN, GRU, Flatten, LSTM
import sklearn
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer, MinMaxScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.ensemble import RandomForestRegressor
from sklearn import datasets, svm, metrics
import skopt
from skopt import gp_minimize, forest_minimize
from skopt.space import Integer, Real, Categorical
import optuna
from sklearn import tree
from geneticalgorithm import geneticalgorithm as ga
from pyomo.environ import *
from pyomo.gdp import *
from pyomo.dae import *
import pyomo.environ as pyEnv
from sko.ACA import ACA_TSP     # Ant Colony Algorithm

# Data Transformation
sc = MinMaxScaler(feature_range=(0,1))      # Normalization to range of 0-1
data_norm = sc.fit_transform(data)
pt = PowerTransformer(standardize=False)   # Box Cox transformation
data_bc = sc.fit_transform(data)
MaxDemand = np.array(data[Regions_].max())
# Input Selection
def datasplit_(features, targets, lookback, sampling_rate, split_fraction):
    train_split = int(split_fraction*len(features))
    # Target split
    train_target = targets[lookback:train_split]
    test_target = targets[train_split:]
    # Feature preparation
    inputs_ = []
    for i in range(lookback, len(features)):
        candidate_ = features[i-lookback:i]
        inputs_.append(candidate_[np.arange(0,lookback,sampling_rate),:].ravel())
    train_input = inputs_[0:train_split-lookback]
    test_input = inputs_[train_split-lookback:]
    return train_input, test_input, train_target, test_target

# Objective Function for Random Forest
def Objective_(params_):
    # Test and Train Split
    x_train, x_test, y_train, y_test = datasplit_(np.array(Variables), np.array(Trips),int(params_[0]),int(params_[1]),0.7018)
    x_train, x_test, y_train, y_test = np.array(x_train), np.array(x_test), np.array(y_train), np.array(y_test)
    # Training and Evaliation Split
    X_training, X_validation, y_training, y_validation = train_test_split(x_train, y_train, test_size=0.25, random_state=128)
    # Model Construction
    regr_rf = RandomForestRegressor(n_estimators=params_[2], max_depth=params_[3], random_state=128)
    regr_rf.fit(X_training, y_training)
    mse_train_ = np.round_(mean_squared_error(np.array(regr_rf.predict(X_training))*MaxDemand,
                                np.array(y_training)*MaxDemand), decimals=6)
    mse_eval_ = np.round_(mean_squared_error(np.array(regr_rf.predict(X_validation))*MaxDemand,
                                np.array(y_validation)*MaxDemand), decimals=6)
    mse_test_ = np.round_(mean_squared_error(np.array(regr_rf.predict(x_test))*MaxDemand,
                                np.array(y_test)*MaxDemand), decimals=6)
    print(mse_train_, mse_eval_, mse_test_)
    return mse_eval_+mse_eval_/mse_train_
```

```
# Grid Search
x0_ = np.arange(10+20, 170, 40)
x1_ = np.arange(1+3, 24, 6)
x2_ = np.arange(10+50, 400, 100)
x3_ = np.arange(1+1, 10, 2)
x0_, x1_, x2_, x3_ = np.meshgrid(x0_, x1_, x2_, x3_)
x0_, x1_, x2_, x3_  = np.ravel(x0_), np.ravel(x1_), np.ravel(x2_), np.ravel(x3_)
gridspace = np.vstack((x0_, x1_, x2_, x3_)).T
result = []
for i in range(len(gridspace)):
   print('Interation:', i+1)
   result.append(Objective_(gridspace[i]))

# Random Search
x0_, x1_, x2_, x3_, result  = [], [], [], [], []
for i in range(200):
   x0 = np.random.randint(10, 171)
   x1 = np.random.randint(1, 25)
   x2 = np.random.randint(10, 401)
   x3 = np.random.randint(1, 11)
   x0_.append(x0)
   x1_.append(x1)
   x2_.append(x2)
   x3_.append(x3)
   print('Interation:', i+1, [x0, x1, x2, x3])
   result.append(Objective_([x0, x1, x2, x3]))

# Bayesian Optimization GP-LCB and RF-LCB
SPACE = [Integer(10, 170, name='x0_'),
        Integer(1, 24, name='x1_'),
        Integer(10, 400, name='x2_'),
        Integer(1, 10, name='x3_')]
res = gp_minimize(Objective_, SPACE,
                acq_func="LCB", n_calls=200,
                kappa= np.random.default_rng().uniform(1.0, 2.0),
                n_random_starts= np.random.randint(50, 150),
                random_state=1042578)
res = forest_minimize(Objective_, SPACE,
                   acq_func="LCB", n_calls=200,
                   kappa= np.random.default_rng().uniform(1.0, 2.0),
                   n_random_starts= np.random.randint(50, 150),
                   base_estimator='RF', random_state=1294571)

# Tree-Structured Parzen Estimator
def Objective_(trial):
   # Variables Declaration
   x0_ = trial.suggest_int('x0_', 10, 170)
   x1_ = trial.suggest_int('x1_', 1, 24)
   x2_ = trial.suggest_int('x2_', 10, 400)
   x3_ = trial.suggest_int('x3_', 1, 10)
   # Test and Train Split
   x_train, x_test, y_train, y_test = datasplit_(np.array(Variables), np.array(Trips), x0_, x1_, 1.0)
   x_train, x_test, y_train, y_test = np.array(x_train), np.array(x_test), np.array(y_train), np.array(y_test)
   # Training and Evaliation Split
   X_training, X_validation, y_training, y_validation = train_test_split(x_train, y_train, test_size=0.25, random_state=128)
   # Model Construction
   regr_rf = RandomForestRegressor(n_estimators=x2_, max_depth=x3_, random_state=128)
   regr_rf.fit(X_training, y_training.ravel())
   mse_train_ = np.round_(mean_squared_error(regr_rf.predict(X_training)*115, 115*y_training), decimals=6)    # 115 is max demand
   mse_eval_ = np.round_(mean_squared_error(regr_rf.predict(X_validation)*115, 115*y_validation), decimals=6) # 115 is max demand
   print(mse_train_)
   return mse_eval_+mse_eval_/mse_train_
study = optuna.create_study(direction="minimize",
                     pruner=optuna.samplers.TPESampler(n_startup_trials=np.random.randint(50, 150),
                                          n_ei_candidates=np.random.randint(20, 50),
                                          multivariate=True, group=True, seed=1541278))
study.optimize(Objective_, n_trials=200)
```

```
# Genetic Algorithm
np.random.seed(5672849)
varbound = np.array([[10, 170], [1, 24], [10, 400], [1, 10]])
vartype = np.array([['int'], ['int'], ['int'], ['int']])
population_ = np.random.randint(10, 30)
mutation_ = np.random.uniform(0.01, 0.5)
elit_ = np.random.uniform(0.0, 0.1)
crossover_ = np.random.uniform(0.2, 0.7)
parents_ = np.random.uniform(0.1, 0.5)
print('population_ %5.1f.' % population_,'mutation_ %5.4f.' % mutation_,'elit_ %5.4f.' % elit_,
     'crossover_ %5.4f.' % crossover_,'parents_ %5.4f.' % parents_)
algorithm_param = {'max_num_iteration': 16,   # Change to make it at least 200 Trials
                'population_size': population_ ,
                'mutation_probability': mutation_,
                'elit_ratio': elit_,
                'crossover_probability': crossover_,
                'parents_portion': parents_,
                'crossover_type':'uniform',
                'max_iteration_without_improv':None}
model=ga(function=Objective_,dimension=4, variable_type_mixed=vartype,
        algorithm_parameters=algorithm_param,variable_boundaries=varbound,
        function_timeout = 30000)
model.run()

# Objective Function for GRUs
def Objective_(params_):   # Train with Categorical Variables
    # Test and Train Split
    x_train, x_test, y_train, y_test = datasplit_(np.array(Variables),np.array(Trips),int(params_[0]),int(params_[1]),0.7018)
    x_train, x_test, y_train, y_test = np.array(x_train), np.array(x_test), np.array(y_train), np.array(y_test)
    # Training and Evaliation Split
    X_training, X_validation, y_training, y_validation = train_test_split(x_train, y_train, test_size=0.25, random_state=128)
    # Model Construction
    model = keras.Sequential()
    model.add(layers.GRU(units=int(params_[2]), input_shape=(np.shape(X_training)[1], np.shape(X_training)[2]),
                    activation= params_[3], kernel_initializer='glorot_uniform',return_sequences=True))
    model.add(layers.Dropout(rate = float(params_[4])))
    model.add(layers.GRU(units=int(params_[5]), activation= params_[6],
                      kernel_initializer='glorot_uniform', return_sequences=False))
    model.add(layers.Dropout(rate = float(params_[7])))
    model.add(layers.Dense(units=30, activation='relu'))
    callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
    model.compile(optimizer=optimizers.Adam(learning_rate=float(params_[8])), loss='mse')
    history = model.fit(X_training, y_training, epochs=100, callbacks=[callback], verbose=0,
                    validation_data=(X_validation, y_validation), batch_size=int(params_[9]))
    #print(np.round_(np.min(history.history["loss"]),decimals=6), np.round_(np.min(history.history["val_loss"]),decimals=6))
    mse_train_ = np.round_(mean_squared_error(np.array(model.predict(X_training))*MaxDemand,
                                      np.array(y_training)*MaxDemand), decimals=6)
    mse_eval_ = np.round_(mean_squared_error(np.array(model.predict(X_validation))*MaxDemand,
                                      np.array(y_validation)*MaxDemand), decimals=6)
    mse_test_ = np.round_(mean_squared_error(np.array(model.predict(x_test))*MaxDemand,
                                      np.array(y_test)*MaxDemand), decimals=6)
    print(mse_train_, mse_eval_, mse_test_, mse_eval_+mse_eval_/mse_train_)
    return mse_eval_+mse_eval_/mse_train_

# Grid Search
x0_ = np.arange(10+26, 171, 52)
x1_ = 1
x2_ = np.arange(10+83, 513, 166)
x3_ = ['relu', 'tanh']
x4_ = 0.1
x5_ = np.arange(10+83, 513, 166)
x6_ = 'relu'
x7_ = 0.05
x8_ = np.arange(0.0001+0.0025, 0.0101, 0.005)
x9_ = np.arange(32+120, 513, 240)
x0_, x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, x9_ = np.meshgrid(x0_, x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, x9_)
x0_, x1_, x2_, x3_, x4_,  = np.ravel(x0_), np.ravel(x1_), np.ravel(x2_), np.ravel(x3_), np.ravel(x4_)
x5_, x6_, x7_, x8_, x9_,  = np.ravel(x5_), np.ravel(x6_), np.ravel(x7_), np.ravel(x8_), np.ravel(x9_)
gridspace = np.vstack((x0_, x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, x9_)).T
result = []
```

```
for i in range(len(gridspace)):
    print('Interation:', i+1)
    result.append(Objective_(gridspace[i]))
# Random Search
x0_, x1_, x2_, x3_, x4_, x5_, x6_, x7_, x8_, x9_, result  = [], [], [], [], [], [], [], [], [], [], []
for i in range(200):
    x0 = np.random.randint(10, 171)
    x1 = np.random.randint(1, 25)
    x2 = np.random.randint(10, 513)
    x3 = np.random.choice(['relu', 'sigmoid', 'tanh'])
    x4 = np.random.choice(np.arange(0.0, 0.801, 0.001))
    x5 = np.random.randint(10, 513)
    x6 = np.random.choice(['relu', 'sigmoid', 'tanh'])
    x7 = np.random.choice(np.arange(0.0, 0.801, 0.001))
    x8 = np.random.choice(np.arange(0.0001, 0.0101, 0.0001))
    x9 = np.random.randint(8, 257)
    x0_.append(x0)
    x1_.append(x1)
    x2_.append(x2)
    x3_.append(x3)
    x4_.append(x4)
    x5_.append(x5)
    x6_.append(x6)
    x7_.append(x7)
    x8_.append(x8)
    x9_.append(x9)
    print('Interation:', i+1, [x0, x1, x2, x3, x4, x5, x6, x7, x8, x9])
    result.append(Objective_([x0, x1, x2, x3, x4, x5, x6, x7, x8, x9]))

# Bayesian Optimization GP-LCB and RF-LCB
SPACE = [Integer(10, 170, name='x0_'),
         Integer(1, 24, name='x1_'),
         Integer(10, 512, name='x2_'),
         Categorical(['relu', 'sigmoid', 'tanh'], name='x3_'),
         Real(0.0, 0.8, name='x4_'),
         Integer(10, 512, name='x5_'),
         Categorical(['relu', 'sigmoid', 'tanh'], name='x6_'),
         Real(0.0, 0.8, name='x7_'),
         Real(0.0001, 0.01, name='x8_'),
         Integer(32, 512, name='x9_')]
res = gp_minimize(Objective_, SPACE,
                  acq_func="LCB", n_calls=200,
                  kappa= np.random.default_rng().uniform(1.0, 2.0),
                  n_random_starts= np.random.randint(50, 150),
                  random_state=3632457)
res = forest_minimize(Objective_, SPACE,
                      acq_func="LCB", n_calls=200,
                      kappa= np.random.default_rng().uniform(1.0, 2.0),
                      n_random_starts= np.random.randint(50, 150),
                      base_estimator='RF', random_state=3612458)

# Tree-Structured Parzen Estimator
def Objective_(trial):
    # Variables Declaration
    x0_ = trial.suggest_int('x0_', 10, 170)
    x1_ = trial.suggest_int('x1_', 1, 24)
    x2_ = trial.suggest_int('x2_', 10, 512)
    x3_ = trial.suggest_categorical('x3_', ['relu', 'sigmoid', 'tanh'])
    x4_ = trial.suggest_uniform('x4_', 0.0, 0.8)
    x5_ = trial.suggest_int('x5_', 10, 512)
    x6_ = trial.suggest_categorical('x6_', ['relu', 'sigmoid', 'tanh'])
    x7_ = trial.suggest_uniform('x7_', 0.0, 0.8)
    x8_ = trial.suggest_uniform('x8_', 0.0001, 0.01)
    x9_ = trial.suggest_int('x9_', 32, 512)
    # Test and Train Split
    x_train, x_test, y_train, y_test = datasplit_(np.array(Variables), np.array(Trips), x0_, x1_, 0.7018)
    x_train, x_test, y_train, y_test = np.array(x_train), np.array(x_test), np.array(y_train), np.array(y_test)
    # Training and Evaliation Split
    X_training, X_validation, y_training, y_validation = train_test_split(x_train, y_train, test_size=0.25, random_state=128)
    # Model Construction
```

```
    model = keras.Sequential()
    model.add(layers.GRU(units= x2_, input_shape=(np.shape(X_training)[1], np.shape(X_training)[2]),
                         activation= x3_, kernel_initializer='glorot_uniform',return_sequences=True))
    model.add(layers.Dropout(rate = x4_))
    model.add(layers.GRU(units= x5_, activation= x6_,
                         kernel_initializer='glorot_uniform', return_sequences=False))
    model.add(layers.Dropout(rate = x7_))
    model.add(layers.Dense(units=30, activation='relu'))
    callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
    model.compile(optimizer=optimizers.Adam(learning_rate= x8_), loss='mse')
    history = model.fit(X_training, y_training, epochs=100, callbacks=[callback], verbose=0,
                        validation_data=(X_validation, y_validation), batch_size=x9_)
    mse_train_ = np.round_(mean_squared_error(np.array(model.predict(X_training))*MaxDemand,
                                              np.array(y_training)*MaxDemand), decimals=6)
    mse_eval_ = np.round_(mean_squared_error(np.array(model.predict(X_validation))*MaxDemand,
                                             np.array(y_validation)*MaxDemand), decimals=6)
    mse_test_ = np.round_(mean_squared_error(np.array(model.predict(x_test))*MaxDemand,
                                             np.array(y_test)*MaxDemand), decimals=6)
    print(mse_train_, mse_eval_, mse_test_, mse_eval_+mse_eval_/mse_train_)
    return mse_eval_+mse_eval_/mse_train_
study = optuna.create_study(direction="minimize",
                            pruner=optuna.samplers.TPESampler(n_startup_trials=np.random.randint(50, 150),
                                                              n_ei_candidates=np.random.randint(20, 50),
                                                              multivariate=True, group=True, seed=3612895))
study.optimize(Objective_, n_trials=200)

# Genetic Algorithm
np.random.seed(1687493)
varbound = np.array([[10, 170], [1, 24], [10, 512], [0, 2], [0.0, 0.8],
                     [10, 512], [0, 2], [0.0, 0.8], [0.0001, 0.01], [32, 512]])
vartype = np.array([['int'], ['int'], ['int'], ['int'], ['real'],
                    ['int'], ['int'], ['real'], ['real'], ['int']])
population_ = np.random.randint(10, 30)
mutation_ = np.random.uniform(0.01, 0.5)
elit_ = np.random.uniform(0.0, 0.1)
crossover_ = np.random.uniform(0.2, 0.7)
parents_ = np.random.uniform(0.1, 0.5)
print('population_ %5.1f.' % population_,'mutation_ %5.4f.' % mutation_,'elit_ %5.4f.' % elit_,
      'crossover_ %5.4f.' % crossover_,'parents_ %5.4f.' % parents_)
algorithm_param = {'max_num_iteration': 21,   # Change to make it at least 200 Trials
                   'population_size': population_ ,
                   'mutation_probability': mutation_,
                   'elit_ratio': elit_,
                   'crossover_probability': crossover_,
                   'parents_portion': parents_,
                   'crossover_type':'uniform',
                   'max_iteration_without_improv':None}

model=ga(function=Objective_,
         dimension=10, variable_type_mixed=vartype,
         algorithm_parameters=algorithm_param,
         variable_boundaries=varbound,
         function_timeout = 30000)
model.run()

# Optimization of Eggholder Function
def Eggholder_(x):
    return -(x[1]+47)*np.sin(np.sqrt(abs(0.5*x[0]+x[1]+47)))-x[0]*np.sin(np.sqrt(abs(x[0]-x[1]-47)))

# Grid Search
x_ = np.arange(-486.4,512, 51.2)
y_ = np.arange(-486.4,512, 51.2)
x_, y_ = np.meshgrid(x_, y_)
x_, y_ = np.ravel(x_), np.ravel(y_)
gridspace = np.vstack((x_, y_,)).T
result = []
for i in range(len(gridspace)):
    result.append(Eggholder_(gridspace[i]))
```

```
# Random Search
x_, y_,result  = [], [], []
for i in range(400):
    x1_ = np.random.default_rng().uniform(-512.,512.)
    y1_ = np.random.default_rng().uniform(-512.,512.)
    x_.append(x1_)
    y_.append(y1_)
    result.append(Eggholder_([x1_ ,y1_]))

# Bayesian Optimization
SPACE = [Real(-512., 512., name='x_'), Real(-512., 512., name='y_')]
res = gp_minimize(Eggholder_, SPACE,
                  acq_func="LCB", n_calls=400,
                  kappa= np.random.default_rng().uniform(0.0, 2.0),
                  n_random_starts= np.random.randint(100, 300),
                  random_state=666)
res = forest_minimize(Eggholder_, SPACE,
                  acq_func="LCB", n_calls=400,
                  kappa= np.random.default_rng().uniform(0.0, 2.0),
                  n_random_starts= np.random.randint(100, 300),
                  base_estimator='RF', random_state=10306524)

# Genetic Algorithm
def Eggholder_(x):
    return -(x[1]+47)*np.sin(np.sqrt(abs(0.5*x[0]+x[1]+47)))-x[0]*np.sin(np.sqrt(abs(x[0]-x[1]-47)))
np.random.seed(1030216)
varbound=np.array([[-512, 512]]*2)
population_ = np.random.randint(5, 20)
mutation_ = np.random.uniform(0.01, 0.5)
elit_ = np.random.uniform(0.0, 0.1)
crossover_ = np.random.uniform(0.2, 0.7)
parents_ = np.random.uniform(0.1, 0.5)
print('population_ %5.1f.' % population_,
      'mutation_ %5.4f.' % mutation_,
      'elit_ %5.4f.' % elit_,
      'crossover_ %5.4f.' % crossover_,
      'parents_ %5.4f.' % parents_)
algorithm_param = {'max_num_iteration': 32,   # Change to make it at least 400 Trials
                   'population_size': population_ ,
                   'mutation_probability': mutation_,
                   'elit_ratio': elit_,
                   'crossover_probability': crossover_,
                   'parents_portion': parents_,
                   'crossover_type':'uniform',
                   'max_iteration_without_improv':None}
model=ga(function=Eggholder_,
         dimension=2,variable_type='real',
         algorithm_parameters=algorithm_param,
         variable_boundaries=varbound)
model.run()

# SVM for Digit Data Classification
digits = datasets.load_digits()
n_samples = len(digits.images)
data = digits.images.reshape((n_samples, -1))
data = data/16.0      # Normalization
# Split data into 50% train and 50% test subsets
X_train, X_test, y_train, y_test = train_test_split(
    data, digits.target, test_size=0.5, shuffle=False, random_state=1024)

def svm_digits(x):
  clf = svm.SVC(C= x[0], gamma=x[1], decision_function_shape='ovo')
  clf.fit(X_train, y_train)
  return -np.round_(metrics.accuracy_score(clf.predict(X_test), y_test)*100, decimals=4)

# Grid Search
x_ = np.arange(0.001+0.36, 10, 0.71)
y_ = np.arange(0.001+0.036, 1, 0.071)
x_, y_ = np.meshgrid(x_, y_)
x_, y_ = np.ravel(x_), np.ravel(y_)
```

```
gridspace = np.vstack((x_, y_,)).T
result = []
for i in range(len(gridspace)):
   result.append(svm_digits(gridspace[i]))

# Random Search
x_, y_, result  = [], [], []
for i in range(200):
   x1_ = np.random.default_rng().uniform(0.001, 10.)
   y1_ = np.random.default_rng().uniform(0.001, 1.)
   x_.append(x1_)
   y_.append(y1_)
   result.append(svm_digits([x1_ , y1_]))

# Bayesian Optimization GP-LCB and RF-LCB
SPACE = [Real(0.001, 10., name='x_'), Real(0.001, 1., name='y_')]
res = gp_minimize(svm_digits, SPACE,
              acq_func="LCB", n_calls=200,
              kappa= np.random.default_rng().uniform(0.0, 2.0),
              n_random_starts= np.random.randint(50, 150))
res = forest_minimize(svm_digits, SPACE,
                  acq_func="LCB", n_calls=200,
                  kappa= np.random.default_rng().uniform(0.0, 2.0),
                  n_random_starts= np.random.randint(50, 150), base_estimator='RF')

# Tree-Structured Parzen Estimator
def svm_digits(trial):
   x = trial.suggest_uniform('x', 0.001, 10.)
   y = trial.suggest_uniform('y', 0.001, 1.)
   clf = svm.SVC(C= x, gamma=y, decision_function_shape='ovo')
   clf.fit(X_train, y_train)
   return np.round_(metrics.accuracy_score(clf.predict(X_test), y_test)*100, decimals=4)
study = optuna.create_study(direction="maximize",
                      pruner=optuna.samplers.TPESampler(n_startup_trials=np.random.randint(50, 150),
                                          n_ei_candidates=np.random.randint(20, 50),
                                          multivariate=True, group=True))
study.optimize(svm_digits, n_trials=200)

# Genetic Algorithm
varbound = np.array([[0.001, 10.], [0.001, 1.]])
population_ = np.random.randint(5, 30)
mutation_ = np.random.uniform(0.01, 0.5)
elit_ = np.random.uniform(0.0, 0.1)
crossover_ = np.random.uniform(0.2, 0.7)
parents_ = np.random.uniform(0.1, 0.5)
print('population_ %5.1f.' % population_,'mutation_ %5.4f.' % mutation_,'elit_ %5.4f.' % elit_,
     'crossover_ %5.4f.' % crossover_,'parents_ %5.4f.' % parents_)
algorithm_param = {'max_num_iteration': 24,   # Change to make it at least 200 Trials
              'population_size': population_ ,
              'mutation_probability': mutation_,
              'elit_ratio': elit_,
              'crossover_probability': crossover_,
              'parents_portion': parents_,
              'crossover_type':'uniform',
              'max_iteration_without_improv':None}
model=ga(function=svm_digits,
       dimension=2,variable_type='real',
       algorithm_parameters=algorithm_param,
       variable_boundaries=varbound)
model.run()

# IDT-E
def initial_random(param1, param2, Numberinit_):
   y1_ = np.random.choice(np.ravel(param1), size=Numberinit_)
   y2_ = np.random.choice(np.ravel(param2), size=Numberinit_)
   initial_random = np.vstack((y1_, y2_)).T
   return initial_random
def decision_tree(var_list, num_initials, num_top_tree, max_iteration):
   var_names_ = list(var_list.keys());
   length_ = [len(list(var_list.items())[0][1]), len(list(var_list.items())[1][1])]
```

```
    param_initial = initial_random(list(var_list.items())[0][1], list(var_list.items())[1][1], num_initials)
    param_list = param_initial.copy();
    objective_list = [];
    for i in range(len(param_initial)):
        objective_list.append(svm_digits(param_initial[i]))

    for j in range(len(param_initial), 1000):  # Total Number of Iteration

        result_ = pd.DataFrame(param_list, columns=var_names_)
        result_['Output_'] = objective_list
        result_ = result_.sort_values(by=['Output_'], ascending=True)

        tree_ = tree.DecisionTreeRegressor()
        tree_ = tree_.fit(np.array(result_[var_names_]), np.array(result_[['Output_']]))

        for k in range(num_top_tree):   # Total Number of Top Tree to explore
            top_tree_k = result_[k:k+1]
            update_param = np.zeros((2, len(length_)))

            for l in range(len(length_)):  # Optimize variable (l) One by One
                vars_l = np.zeros((length_[l], len(length_)))
                vars_l[:,0] = np.array(top_tree_k)[:,0]
                vars_l[:,1] = np.array(top_tree_k)[:,1]
                vars_l[:,l] = list(var_list.items())[l][1]
                result_var_l = pd.DataFrame(vars_l, columns=var_names_)
                result_var_l['tree_k'] = np.round_(tree_.predict(np.array(result_var_l)), decimals=5)
                vars_l_new = result_var_l[result_var_l['tree_k']==float(np.array(top_tree_k)[:,2])]
                update_param[:,l] = [np.array(vars_l_new)[:,l].min(), np.array(vars_l_new)[:,l].max()]
            leaf_0 = np.array(update_param)[:,0]
            leaf_1 = np.array(update_param)[:,1]
            leaf_0, leaf_1 = np.meshgrid(leaf_0, leaf_1)
            leaf_0, leaf_1 = np.ravel(leaf_0), np.ravel(leaf_1)
            update_paramnew = np.vstack((leaf_0, leaf_1)).T

            new_array = [tuple(row) for row in update_paramnew] # Remove redundant row
            update_paramnew = np.unique(new_array, axis=0)

            repeat_param = [];
            for m in range(len(update_paramnew)):  # Remove the Repeated Candidate
                for n in range(len(param_list)):
                    if np.array_equal(param_list[n], update_paramnew[m]) == True:
                        repeat_param.append(m)
            candi_param = np.delete(update_paramnew, repeat_param, 0)
            if len(candi_param) == 0: break
            param_list = np.vstack([param_list, candi_param])
            for o in range(len(candi_param)):
                objective_list.append(svm_digits(candi_param[o]))
            if len(objective_list) > max_iteration : break
        if len(objective_list) > max_iteration : break
    return param_list, objective_list

np.random.seed(1036487);
var_list = {'x_0': np.arange(0.001, 10.001, 0.001),
            'x_1': np.arange(0.001, 1.001, 0.001)}
num_initials = np.random.randint(50, 150);  # Number of Initials
num_top_tree = np.random.randint(2, 6);      # Number of Top Tree to be minimized
max_iteration = 200;
X_, Y_ = decision_tree(var_list, num_initials, num_top_tree, max_iteration)

# IDT-R
def initial_random(param0, param1, Numberinit_):
    b0_ = np.random.choice(np.ravel(param0), size=Numberinit_)
    b1_ = np.random.choice(np.ravel(param1), size=Numberinit_)
    initial_random = np.vstack((b0_, b1_)).T
    return initial_random

def decision_tree(var_list, num_initials, num_top_tree, random_in_tree, max_iteration):
    var_names_ = list(var_list.keys());
    length_ = [len(list(var_list.items())[0][1]), len(list(var_list.items())[1][1])]
    param_initial = initial_random(list(var_list.items())[0][1], list(var_list.items())[1][1], num_initials)
```

```
    param_list = param_initial.copy();
    objective_list = [];
    for i in range(len(param_initial)):
        objective_list.append(svm_digits(param_initial[i]))

    for j in range(len(param_initial), 1000):  # Total Number of Iteration

        result_ = pd.DataFrame(param_list, columns=var_names_)
        result_['Output_'] = objective_list
        result_ = result_.sort_values(by=['Output_'], ascending=True)

        tree_ = tree.DecisionTreeRegressor()
        tree_ = tree_.fit(np.array(result_[var_names_]), np.array(result_[['Output_']]))

        for k in range(num_top_tree):   # Total Number of Top Tree to explore
            top_tree_k = result_[k:k+1]
            update_param = np.zeros((random_in_tree, len(length_)))

            for l in range(len(length_)):  # Optimize variable (l) One by One
                vars_l = np.zeros((length_[l], len(length_)))
                vars_l[:,0] = np.array(top_tree_k)[:,0]
                vars_l[:,1] = np.array(top_tree_k)[:,1]
                vars_l[:,l] = list(var_list.items())[l][1]
                result_var_l = pd.DataFrame(vars_l, columns=var_names_)
                result_var_l['tree_k'] = np.round_(tree_.predict(np.array(result_var_l)), decimals=4)
                vars_l_new = result_var_l[result_var_l['tree_k']==float(np.array(top_tree_k)[:,2])]
                update_param[:,l] = np.random.choice(np.ravel(vars_l_new[var_names_[l]]), size=random_in_tree)
            new_array = [tuple(row) for row in update_param] # Remove redundant row
            update_param = np.unique(new_array, axis=0)
            repeat_param = [];
            for m in range(len(update_param)):  # Remove the Repeated Candidate
                for n in range(len(param_list)):
                    if np.array_equal(param_list[n], update_param[m]) == True:
                        repeat_param.append(m)
            candi_param = np.delete(update_param, repeat_param, 0)
            if len(candi_param) == 0: break
            param_list = np.vstack([param_list, candi_param])
            for o in range(len(candi_param)):
                objective_list.append(svm_digits(candi_param[o]))
            if len(objective_list) > max_iteration : break
        if len(objective_list) > max_iteration : break
    return param_list, objective_list

np.random.seed(7943158);
var_list = {'x_0': np.arange(0.001, 10.001, 0.001),
            'x_1': np.arange(0.001, 1.001, 0.001)}
num_initials = np.random.randint(50, 150);  # Number of Initials
num_top_tree = np.random.randint(2, 5);    # Number of Top Tree to be minimized
random_in_tree = np.random.randint(1, 3); # Number of Random Pickup from Optimal Tree
max_iteration = 200;
X_, Y_ = decision_tree(var_list, num_initials, num_top_tree, random_in_tree, max_iteration)
# Rebalancing Optimization
cost_km = 1;     # Transportation Cost per km
p_remainFE = 5;  # Penalty cost of each remaining Faulty E-scooter
p_remainLBE = 3; # Penalty cost of each remaining Low Battery E-scooter
p_Unmet = 2;     # Penalty cost of Unmet Demand
p_Excess = 1;  # Penalty cost of Exessive Number of E-Scooter
v_capacity = 35; # Capacity of Vehicles
# Sampling Approach
def sampling_tg(pred_tg, pred_std,first_zeros,num_scen):
    Trip_Gap = np.zeros((num_scen,first_zeros))
    for i in range(len(pred_tg)):       # Generate for 10 Nodes
        sampling_i = np.random.normal(0, 1, num_scen)
        sampling_i[sampling_i > 2] = 2     # Set bound of Resampling to be within 2 to -2
        sampling_i[sampling_i < -2] = -2
        scenario_tg = pred_tg[i] + pred_std[i]*sampling_i
        scenario_tg = scenario_tg.round(0).reshape(-1,1)
        Trip_Gap = np.hstack((Trip_Gap, scenario_tg))
    Trip_Gap = np.array(Trip_Gap).astype(int)
    return Trip_Gap
```

```
# Baseline as Daily and Weekly Historical Trip Gaps
def seasonal_ha(targ_date, all_df, first_zeros, seasonal):
    all_df_ = all_df.copy()
    sha_ = []
    sha_ = all_df_.drop(all_df_[all_df_.new_date >= targ_date].index)
    sha_ = np.array(sha_)
    sha_ = sha_[np.arange(len(sha_)%seasonal,len(sha_),seasonal), 1:]
    return np.hstack((np.zeros((len(sha_), first_zeros)), sha_)).astype(int)

# Rebalancing Optimziation by ILP
while ii_i < 30:
  i_i = to_runs[ii_i]  # List of Random Instance
  print('#######################')
  print(' Result of Group', i_i)
  ### Actual Data ###
  #Trip_Gap = np.array([list(Actual_trip[i_i])])
  ### TripGap Sampling ##
  np.random.seed(123)
  Trip_Gap = sampling_tg(predict_tg[i_i], predict_std[i_i], 3, 100)
  ### Trip Gas as Seasonal HA ## 24:Daily and 168:Weekly
  #np.random.seed(123)
  #Trip_Gap = seasonal_ha(my_date[i_i], b_b, 10, 24)
  n = len(dist_matrix)
  n_sce = len(Trip_Gap)
  # Boundary Construction
  Bound_FE,Bound_PLBE,Bound_DLBE,Bound_UE = [],[],[],[]
  for k in range(n):
    Bound_FE.append([0, Faulty_E[k]])
    Bound_PLBE.append([0, max(LowBat_E[k] - Num_Dock[k], 0)])
    Bound_DLBE.append([0, max(Num_Dock[k] - LowBat_E[k], 0)])
    Bound_UE.append([0, max(0, Usable_E[k] - Min_E[k])])
  # MILP Model
  model = pyEnv.ConcreteModel()
  model.node_ = pyEnv.RangeSet(0, n-1)
  model.node_1 = pyEnv.RangeSet(1, n-1)
  model.stoch = pyEnv.RangeSet(0, n_sce-1)
  # Decision Variables
  model.x = pyEnv.Var(model.node_, model.node_, within = pyEnv.Binary)
  model.u = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = (0, n-1))
  model.pfe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = Bound_FE)        # Pickup FE
  model.vfe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = (0, v_capacity))  # FE on Vehicle
  model.plbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = Bound_PLBE)      # Pickup LBE
  model.dlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = Bound_DLBE)      # DropOff LBE
  model.rlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = Bound_PLBE)      # Remaining LBE
  model.vlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = (0, v_capacity)) # LBE on Vehicle
  model.pue = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = Bound_UE)        # Pickup UE
  model.due = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = (0, v_capacity))  # DropOff FE
  model.vue = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers, bounds = (0, v_capacity))  # UE on Vehicle
  model.ud = pyEnv.Var(model.node_, model.stoch, within = pyEnv.NonNegativeIntegers)               # Unmet Demand
  model.ee = pyEnv.Var(model.node_, model.stoch, within = pyEnv.NonNegativeIntegers)               # Excessive E-scooter
  # Objective Function
  def obj_func(model):
    sum_distance = 0
    remain_fe = 0
    remain_lbe = 0
    unmet_demand = 0
    excess_ = 0
    pickup_cost = 0
    for i in range(n):
      remain_fe = remain_fe + Faulty_E[i] - model.pfe[i]
      remain_lbe = remain_lbe + model.rlbe[i]
      pickup_cost = pickup_cost+ 0.1*(model.pfe[i]+model.plbe[i]+model.pue[i])
      for j in range(n):
        sum_distance = sum_distance + model.x[i,j] * dist_matrix[i,j]
      for k in range(n_sce):
        unmet_demand = unmet_demand + model.ud[i,k]
        excess_ = excess_ + model.ee[i,k]
    return sum_distance*cost_km +pickup_cost+ remain_fe*p_remainFE + remain_lbe*p_remainLBE + (1/n_sce)*(unmet_demand*p_Unmet
+ excess_*p_Excess)
  model.objective = pyEnv.Objective(rule = obj_func, sense = pyEnv.minimize)
```

```
  # Constraint of Node Leave and Arrive
  model.constNode = pyEnv.ConstraintList()
  for i in range(n):
    model.constNode.add(sum(model.x[j, i] for j in model.node_) == 1)
    model.constNode.add(sum(model.x[i, j] for j in model.node_) == 1)
  # Constraints Subtour Elimination
  model.constSub = pyEnv.ConstraintList()
  model.constSub.add(model.u[0] == 0)
  for i in range(1, n):
    for j in range(1, n):
        if i != j:
            model.constSub.add(model.u[i] - model.u[j] + model.x[i,j]*n <= n-1)
        else :
            model.constSub.add(model.u[i] - model.u[j]  == 0)
  # Constraints Loading and Unloading FE
  model.constFE = pyEnv.ConstraintList()
  model.constFE.add(model.vfe[0] == 0)
  for i in range(n):
    for j in range(1, n):
        model.constFE.add(model.vfe[j] - model.vfe[i] - model.pfe[j] + 1000000*(1-model.x[i,j]) >= 0)
  for i in range(n):
    for j in range(n):
        model.constFE.add(model.vfe[i] - model.vfe[j] + model.pfe[j] + 1000000*(1-model.x[i,j]) >= 0)
  # Constraints Loading and Unloading LBE
  model.constLBE = pyEnv.ConstraintList()
  model.constLBE.add(model.vlbe[0] == 0)
  for i in range(n):
    for j in range(1, n):
        model.constLBE.add(model.vlbe[j] - model.vlbe[i] - model.plbe[j] + model.dlbe[j] + 1000000*(1-model.x[i,j]) >= 0)
  for i in range(n):
    for j in range(n):
        model.constLBE.add(model.vlbe[i] - model.vlbe[j] + model.plbe[j] - model.dlbe[j] + 1000000*(1-model.x[i,j]) >= 0)
  # Constraints Loading and Unloading UE
  model.constUE = pyEnv.ConstraintList()
  model.constUE.add(model.vue[0] == model.pue[0])
  for i in range(n):
    for j in range(1, n):
        model.constUE.add(model.vue[j] - model.vue[i] - model.pue[j] + model.due[j] + 1000000*(1-model.x[i,j]) >= 0)
  for i in range(n):
    for j in range(n):
        model.constUE.add(model.vue[i] - model.vue[j] + model.pue[j] - model.due[j] + 1000000*(1-model.x[i,j]) >= 0)
  # Constraints Remain LBE, Unmet Demand and Excessive
  model.constUDEE = pyEnv.ConstraintList()
  model.constUDEE.add(model.rlbe[0] == 0)
  for k in range(n_sce):
    model.constUDEE.add(model.ud[0,k] == 0)
    model.constUDEE.add(model.ee[0,k] == 0)
  for i in range(1, n):
    model.constUDEE.add( max(LowBat_E[i] - Num_Dock[i], 0) - model.plbe[i] - model.rlbe[i] == 0)
    for k in range(n_sce):
      model.constUDEE.add(Min_E[i]-Usable_E[i]+model.pue[i]-model.due[i]+max(0, Trip_Gap[k,i])-model.ud[i,k] <= 0)
      model.constUDEE.add(Faulty_E[i]-model.pfe[i]+model.rlbe[i]+Usable_E[i]-model.pue[i]+model.due[i]-Trip_Gap[k,i]-Max_E[i]-
model.ee[i,k] <= 0)
  # Constraints Vehicle Capacity
  model.constVall = pyEnv.ConstraintList()
  for i in range(1, n):
    model.constVall.add(model.vfe[i] + model.vlbe[i] + model.vue[i] <= v_capacity)
    # Model Sovler
  solver = pyEnv.SolverFactory("glpk") # glpk
  result = solver.solve(model,timelimit= 1200) # timelimit (15 nodes = 1200, 30 nodes = 2400, 60 nodes = 3600)
  try:
    print("Objective Value: ", pyEnv.value(model.objective))
    print("U-i: ", np.array(list(model.u.get_values().items()))[:,1].astype(int))
    print("Routing Result X-ij: ")
    #print(np.array(list(model.x.get_values().items()), dtype=object)[:,1].reshape(n,n).astype(int))
    print("PFE-i: ", np.array(list(model.pfe.get_values().items()))[:,1].astype(int))
    print("PLBE-i: ", np.array(list(model.plbe.get_values().items()))[:,1].astype(int))
    print("DLBE-i: ", np.array(list(model.dlbe.get_values().items()))[:,1].astype(int))
    print("PUE-i: ", np.array(list(model.pue.get_values().items()))[:,1].astype(int))
    print("DUE-i: ", np.array(list(model.due.get_values().items()))[:,1].astype(int))
```

```
    rout_seq = np.array(list(model.x.get_values().items()), dtype=object)[:,1].reshape(n,n).astype(int)
    x_seq = []
    for seq_ in range(len(rout_seq)):
      x_seq.append([seq_,np.argmax(rout_seq[seq_])])
    print(x_seq)
  except ValueError:
    print("No result Found! Please try again ...")
  ii_i += 1

# Rebalancing Optimziation by ACO-ILP
def route_length(x):
    r_r = np.append(np.array(x), [0])
    sum_distance_ = 0
    for i in range(1, len(r_r)):
        sum_distance_ = sum_distance_+  dist_matrix[r_r[i-1],r_r[i]]
    return np.round_(sum_distance_*cost_km, decimals=3)
def get_penalty_cost(x):
  r = np.array(x).astype(int)
  n = len(dist_matrix)
  n_sce = len(Trip_Gap)
  # Model Constrution
  model = pyEnv.ConcreteModel()
  model.node_ = pyEnv.RangeSet(0, n-1)
  model.stoch = pyEnv.RangeSet(0, n_sce-1)
  # Decision Variables
  model.pfe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers)  # Pickup FE
  model.vfe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers)  # FE on Vehicle
  model.plbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers) # Pickup LBE
  model.dlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers) # DropOff LBE
  model.rlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers) # Remaining LBE
  model.vlbe = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers) # LBE on Vehicle
  model.pue = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers)  # Pickup UE
  model.due = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers)  # DropOff FE
  model.vue = pyEnv.Var(model.node_, within = pyEnv.NonNegativeIntegers)  # UE on Vehicle
  model.ud = pyEnv.Var(model.node_, model.stoch, within = pyEnv.NonNegativeIntegers)   # Unmet Demand
  model.ee = pyEnv.Var(model.node_, model.stoch, within = pyEnv.NonNegativeIntegers)   # Excessive E-scooter
  # Objective Function
  def obj_func(model):
    remain_fe = 0
    remain_lbe = 0
    unmet_demand = 0
    excess_ = 0
    pickup_cost = 0
    for i in range(n):
      remain_fe = remain_fe + Faulty_E[r[i]] - model.pfe[r[i]]
      remain_lbe = remain_lbe + model.rlbe[r[i]]
      pickup_cost = pickup_cost+ 0.1*(model.pfe[r[i]]+model.plbe[r[i]]+model.pue[r[i]])
      for k in range(n_sce):
        unmet_demand = unmet_demand + model.ud[r[i],k]
        excess_ = excess_ + model.ee[r[i],k]
    return pickup_cost+ remain_fe*p_remainFE + remain_lbe*p_remainLBE + (1/n_sce)*(unmet_demand*p_Unmet + excess_*p_Excess)
  model.objective = pyEnv.Objective(rule = obj_func, sense = pyEnv.minimize)
  # Constraints Loading and Unloading FE
  model.const = pyEnv.ConstraintList()
  model.const.add(model.pfe[0] == 0)
  model.const.add(model.vfe[0] == 0)
  model.const.add(model.plbe[0] == 0)
  model.const.add(model.dlbe[0] == 0)
  model.const.add(model.vlbe[0] == 0)
  model.const.add(model.rlbe[0] == 0)
  model.const.add(model.due[0] == 0)
  model.const.add(model.pue[0] <= Usable_E[0])
  model.const.add(model.pue[0] == model.vue[0])
  model.const.add(model.vue[0] <= v_capacity)
  for i in range(n_sce):
    model.const.add(model.ud[0, i] == 0)
    model.const.add(model.ee[0, i] == 0)
  for k in range(1,n):
    model.const.add(model.vfe[r[k]] == model.pfe[r[k]] + model.vfe[r[k-1]])
    model.const.add(model.pfe[r[k]] <= Faulty_E[r[k]])
```

```
    model.const.add(model.vlbe[r[k]] == model.plbe[r[k]] - model.dlbe[r[k]] + model.vlbe[r[k-1]])
    model.const.add(max(LowBat_E[r[k]] - Num_Dock[r[k]], 0) >= model.plbe[r[k]])
    #model.const.add( model.dlbe[r[k]] <= model.vlbe[r[k-1]])
    model.const.add( model.dlbe[r[k]] <=  max(Num_Dock[r[k]] - LowBat_E[r[k]], 0))

    model.const.add(model.rlbe[r[k]] == max(LowBat_E[r[k]] - Num_Dock[r[k]], 0) - model.plbe[r[k]])
    model.const.add(model.vue[r[k]] == model.vue[r[k-1]] + model.pue[r[k]] - model.due[r[k]])
    model.const.add(model.pue[r[k]] <=  max(Usable_E[r[k]] - Min_E[r[k]], 0))
    #model.const.add(model.due[r[k]] <= model.vue[r[k-1]])
    model.const.add(model.vfe[r[k]] + model.vlbe[r[k]] + model.vue[r[k]]<= v_capacity)

    for j in range(n_sce):
      model.const.add(Min_E[r[k]]-Usable_E[r[k]]+model.pue[r[k]]-model.due[r[k]]+max(0, Trip_Gap[j, r[k]])-model.ud[r[k], j] <= 0)
      model.const.add(Usable_E[r[k]]-model.pue[r[k]]+model.due[r[k]]-Trip_Gap[j, r[k]]-Max_E[r[k]]-model.ee[r[k], j] <= 0)

  solver = pyEnv.SolverFactory("glpk")
  solver.solve(model, timelimit= 300)  # timelimit (15 nodes = 1200, 30 nodes = 2400, 60 nodes = 3600) tee = True,
  #print("PFE-i: ", np.array(list(model.pfe.get_values().items()))[:,1].astype(int))
  #print("PLBE-i: ", np.array(list(model.plbe.get_values().items()))[:,1].astype(int))
  #print("DLBE-i: ", np.array(list(model.dlbe.get_values().items()))[:,1].astype(int))
  #print("PUE-i: ", np.array(list(model.pue.get_values().items()))[:,1].astype(int))
  #print("DUE-i: ", np.array(list(model.due.get_values().items()))[:,1].astype(int))
  #print(np.array(list(model.ud.get_values().items()), dtype=object)[:,1].reshape(n,n_sce).astype(int))
  #service_level = np.array(list(model.ud.get_values().items()), dtype=object)[:,1].reshape(-1,1).astype(int)
  #print(len(service_level), len(service_level[service_level>0]))
  return np.round_(pyEnv.value(model.objective), decimals=5)
# Combine Driving Distance Cost and Penalty Cost for ACO
def Routing_(x):
    x = np.array(x).astype(int)
    total_cost = route_length(x) + float(get_penalty_cost(x))
    #print(np.round_(total_cost, decimals=3))
    return np.round_(total_cost, decimals=3)

for i_i in range(0, 30):
  np.random.seed(123)
  print('########################')
  print(' Result of Group', i_i)
  ### Actual Data ###
  #Trip_Gap = np.array([list(Actual_trip[i_i])])
  ### TripGap Sampling ##
  Trip_Gap = sampling_tg(predict_tg[i_i], predict_std[i_i], 5, 100)
  ### Trip Gas as Seasonal HA ## 24:Daily and 168:Weekly
  #Trip_Gap = seasonal_ha(my_date[i_i], b_b, 5, 168)
  aca = ACA_TSP(Routing_, n_dim=35, size_pop=90, max_iter=20, distance_matrix=dist_matrix)
  best_x, best_y = aca.run()
  print(' best_x:', best_x, '\n', 'best_y:', best_y)
  print(np.min(aca.generation_best_Y))
```

BIOGRAPHY

Name:	Narith Saum
Date of Birth:	October 05, 1992
Education:	2015: Bachelor of Engineering (Civil Engineering) Institute of Technology of Cambodia. 2017: Master of Engineering (Transportation Engineering) Chulalongkorn University.

International Journal

Saum, N., Sugiura, S., & Piantanakulchai, M. (2022). Hyperparameter Optimization Using Iterative Decision Tree (IDT), *IEEE Access*, vol. 10, pp. 106812-106827, https://doi.org/10.1109/ACCESS.2022.3212387.

International Conference

Saum, N., Sugiura, S., & Piantanakulchai, M. (2020). Short-Term Demand and Volatility Prediction of Shared Micro-Mobility: a case study of e-scooter in Thammasat University. *Proceedings of Forum on Integrated and Sustainable Transportation Systems* (*FORUM ISTS 2020*), November 3-5, 2020, Delft, The Netherlands. https://doi.org/10.1109/FISTS46898.2020.9264852.

Saum, N., & Piantanakulchai, M. (2019). A Review on an Emerging New Mode of Transport: The Shared Dockless Electric Scooter. *Proceedings of 13th International Conference of Eastern Asia Society for Transportation Studies* (*EASTS 2019*), September 9-12, 2019, Colombo, Sri Lanka.

Saum, N., & Rudjanakanoknad, J. (2017). How to Make Safer Commuting of Garment and Footwear Workers in Phnom Penh based on Stakeholders' Opinions. *Proceedings of 12th International Conference of Eastern Asia Society for Transportation Studies* (*EASTS 2017*), September 18-21, 2017, Ho Chi Minh City, Vietnam.