VILNIUS UNIVERSITY

FACULTY OF MATHEMATICS AND INFORMATICS

DATA SCIENCE STUDY PROGRAMME

Master's thesis

Comparison of Applying Advanced Spatiotemporal Clustering Algorithms with Machine Learning Methods for Forecasting of Urban Mobility Demand

Pažangių erdvės ir laiko klasterizavimo algoritmų ir mašininio mokymosi metodų taikymo lyginamoji analizė miesto mobilumo paklausos prognozavimui

Anna Yasyreva

Supervisor : Prof. Dr. Jurgita Markevičiūtė

Vilnius
2026

Acknowledgements

The author is thankful to her family and dear ones for their support, understanding and patience during the challenging time of writing the thesis, so I could concentrate my time and energy on doing my research.

I am also grateful for my university professors for encouraging, inspiring and motivating me during the course of my studies.

Summary

Urban transportation networks generate vast spatiotemporal data streams that offer untapped potential for optimizing mobility flows and mitigating congestion. This thesis investigates the dynamics of New York City taxi trip data to develop predictive models that enhance urban transportation efficiency. By integrating temporal, spatial, and derivative variables, this study constructs a multi-level analytical framework employing advanced feature engineering, hybrid clustering, and machine learning methodologies.

The research pursues three primary objectives: (1) examining the efficacy of hybrid clustering algorithms in identifying demand hotspots; (2) benchmarking the predictive performance of SARIMA, XGBoost, and LSTM networks; and (3) evaluating the stability of these models across varying forecast horizons. Exploratory analysis reveals distinct spatiotemporal patterns corresponding to rush hours and weekly cycles, which were successfully segmented using K-means and HDBSCAN clustering.

Comparative evaluation indicates that machine learning approaches significantly outperform the baseline SARIMA model. Specifically, the XGBoost model achieved the highest accuracy, reducing the Root Mean Square Error (RMSE) for one-day forecast by 89.4% compared to SARIMA and 39.1% compared to LSTM. For the three-day forecast, RMSE reduced by 90.3% compared to SARIMA and 40.3% compared to LSTM, and for seven-day forecast RMSE reduced by 87.8% compared to SARIMA and 36.4% compared to LSTM. Furthermore, analysis of the forecasting horizon demonstrates that XGBoost exhibits slower performance degradation over longer time windows than LSTM for mean absolute error (MAE) and mean absolute percentage error (MAPE) evaluation metrics, suggesting superior robustness for operational planning and stability for long-horizon forecasting. These findings provide actionable insights for dynamic fleet management and urban mobility optimization.

Keywords: spatiotemporal data, urban mobility, clustering algorithms, demand forecasting, urban mobility, K-means, HDBSCAN, SARIMA, XGBoost, LSTM

Santrauka

Miesto transporto tinklai generuoja didelius erdvės ir laiko (angl. spatiotemporal) duomenų srautus, turinčius neišnaudotą potencialą optimizuoti judumo srautus ir mažinti spūstis. Šiame magistro darbe tiriama Niujorko taksi kelionių duomenų dinamika, siekiant sukurti prognozavimo modelius, galinčius padidinti miesto transporto efektyvumą. Integruojant laikinius, erdvinius ir išvestinius kintamuosius, tyrime sudaroma daugiapakopė analitinė sistema, kurioje taikoma pažangi požymių inžinerija, hibridinis klasterizavimas ir mašininio mokymosi metodikos.

Tyrimas orientuotas į tris pagrindinius tikslus: (1) ištirti hibridinių klasterizavimo algoritmų efektyvumą identifikuojant paklausos židinius; (2) palyginti SARIMA, XGBoost ir ilgosios trumposios atminties (angl. Long Short-Term Memory, LSTM) tinklų prognozavimo našumą; bei (3) įvertinti šių modelių stabilumą kintant prognozavimo horizontui. Tiriamoji duomenų analizė atskleidė ryškius erdvėlaikinius dėsningumus, susijusius su piko valandomis ir savaitiniais ciklais, kurie buvo sėkmingai segmentuoti naudojant K-means ir HDBSCAN klasterizavimo algoritmus.

Lyginamoji analizė rodo, kad mašininio mokymosi metodai reikšmingai pranoksta bazinį SARIMA modelį. Konkrečiai, XGBoost modelis pasiekė didžiausią tikslumą, sumažindamas vienos dienos prognozės šaknies vidurkio kvadratinę paklaidą (RMSE) 89.4% lyginant su SARIMA ir 39.1% lyginant su LSTM. Trijų dienų prognozės RMSE sumažėjo 90.3% lyginant su SARIMA modeliu ir 40.3% lyginant su LSTM, bei septynių dienų prognozės RMSE sumažėjo 87.8% lyginant su SARIMA ir 36.4% lyginant su LSTM. Be to, prognozavimo horizonto analizė rodo, kad ilgėjant laiko langams XGBoost pasižymi lėtesniu tikslumo mažėjimu nei LSTM lyginant vidutinės absoliučios paklaidos (MAE) ir vidutinės absoliučios procentinės paklaidos (MAPE) metrikas, o tai indikuoja didesnį šio modelio patikimumą operaciniam planavimui bei stabilumą atliekant ilgojo horizonto prognozes. Šie rezultatai suteikia praktinių įžvalgų dinaminiam automobilių parko valdymui ir miesto judumo optimizavimui

Raktiniai žodžiai: erdvės ir laiko duomenys, miesto judumas, klasterizavimo algoritmas, paklausos prognozavimas, K-means, HDBSCAN, SARIMA, XGBoost, LSTM

List of Figures

List of Tables

Contents

List of abbreviations

VAR	Vector autoregression statistical model
SARIMA	Seasonal ARIMA
XGBoost	eXtreme Gradient Boosting - ensemble tree-based algorithm.
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long short-term memory recurrent neural network
ConvLSTM	Convolutional Long short-term memory recurrent neural network
DBSCAN	Density-based spatial clustering of applications with noise
ST-DBSCAN	SpatioTemporal DBSCAN
HDBSCAN	Hierarchical version of DBSCAN
ADF	Augmented Dickey-Fuller test
ACF	Autocorrelation Function
PACF	Partial Autocorrelation Function

Introduction

The intensification of urbanization and the increase in vehicle ownership have precipitated considerable challenges for city transportation networks, influencing congestion levels, travel efficiency, and environmental impact. Although technological advancements have enabled the continuous collection and archiving of urban mobility data, significant uncertainty remains in terms of leveraging such data for tangible improvements in cities' transportation systems.

It might be expected that the availability of large spatiotemporal mobility datasets would allow optimal traffic flow predictions and resource allocations; however, research in this field has produced differentiated findings on the effectiveness and universality of existing modelling techniques. The ability to forecast demand, mitigate congestion, and allocate transport efficiently is crucial not only for data scientists but also for city planners and mobility service providers. More robust analytical frameworks are needed to address recurrent shortcomings, such as underperformance in generalizing across varying urban districts and time periods and insufficient integration of external influences.

The aim of this thesis is to examine whether the application of combined clustering algorithms with different data science approaches - such as classical statistical Seasonal ARIMA (SARIMA) models, advanced machine learning XGBoost algorithm and modern deep-learning Long Short-Term Memory (LSTM) network - to forecast trips demand can produce actionable insights in urban mobility. This research will test several assumptions regarding the improvement of predictive accuracy through elaborate feature engineering and clustering, as well as the potential for data-driven allocation strategies to reinforce urban transportation planning. Specifically, the objectives include quantifying spatial and temporal transportation demand, developing scalable models to forecast travel frequency and evaluating the impact of methodological choices on practical deployment in the New York metropolitan context.

Research objectives:

- Examine whether the application of combined clustering algorithms with different forecasting models can produce actionable insights in urban mobility;
- Apply three models with different approaches to compare the prediction capabilities - SARIMA, XGBoost and LSTM;
- Compare the forecasting horizon for each model to evaluate which works best and deteriorates less in which conditions.

The significance of this research is underscored by the pressing need for data-driven urban transportation solutions. Effective modelling and forecasting of mobility demand are indispensable for reducing congestion, promoting sustainable economic activity, and delivering improved urban experiences. By integrating large-scale real-world data with advanced analytical techniques, this thesis aims to bridge the observed gap in the existing studies and contribute operationally relevant knowledge for the optimization of city mobility systems, i.e. combining advanced spatiotemporal analytics with real-world transportation data, proposing insights with concrete relevance for operational and policy planning in large urban contexts.

The innovativeness of this research lies in the application of combined hierarchical clustering algorithms for temporal and spatial features, as well as the investigation of prediction on different time horizons, which is not usually done in such studies.

1 Related work

The prediction of taxi demand and flow among city districts has evolved substantially across different methodological paradigms: classical statistical approaches, machine learning-based techniques, and constantly evolving deep learning architectures.

Although previous research has explored spatial and temporal variation in urban taxi demand, many gaps persist in the predictive accuracy, interpretability, and adaptability of models used in real-world urban environments. Existing studies often struggle to generalize across districts and periods or do not quantify the impact of feature engineering on predictive success. In this section, the methodological landscape on spatiotemporal modelling is reviewed, and the research gaps are identified together with discussion of their strengths and limitations, providing the theoretical foundation and motivation for the development of advanced and more robust and sophisticated modelling approaches that are required for the forecasting of this type of complex data.

1.1 Classical statistical approaches - Vector Autoregression and Seasonal ARIMA

Traditional methodologies in transportation forecasting have primarily relied on statistical and econometric techniques. Classical approaches consist of descriptive statistical methods for traffic pattern analysis, ARIMA and Seasonal ARIMA models for temporal forecasting as well as regression-based techniques. Those methods have provided foundational and grounding insights into urban mobility patterns. This approach usually treats spatial and temporal dimensions separately.

However, most of the current SARIMA models are unable to model and forecast multiple time series at the same time and can only be calculated one by one, which is a disadvantage from a computational point of view. Besides, it cannot capture the correlation between multiple time series, which may negatively affect its modelling effectiveness as is discussed in the work of Li et al. [23]. Also, it assumes linear relationships and stationary time series, which taxi demand data frequently violates due to non-linear behavioural patterns and external shocks.

Vector Autoregression (VAR) models represent a fundamental multivariate time series approach that captures linear relationships among multiple time series simultaneously [12]. However, VAR models face significant limitations when applied to taxi demand prediction across multiple geographic zones. As demonstrated by Faghih et al. [12], when k geographic regions are modelled with VAR of order p, the model requires estimation of $k^2 \cdot p$ parameters using only $k \cdot T$ observations, where T represents the temporal length [35]. This high-dimensionality problem becomes an issue as the number of zones increases, leading to overfitting and poor generalization. Moreover, traditional VAR models treat spatial and temporal dimensions separately, failing to capture the critical spatial autocorrelation inherent in taxi demand patterns. For those reasons the VAR model might not be the best option for the forecasting of the current data, which is why it was not selected for this research.

To sum up, SARIMA type of models are good for their interpretability and simplicity - their interpretability is substantially higher than black-box machine learning methods, the coefficients directly represent influence strength, making them explainable to transportation practitioners. These models

are computationally efficient, requiring minimal computational resources for real-time applications, which makes them a good baseline model to compare to.

1.2 Machine learning models - eXtreme Gradient Boosting

Machine learning methods have substantially advanced taxi demand prediction by introducing non-linear transformations and ensemble learning principles. Among these, eXtreme Gradient Boosting (XGBoost) has emerged as a prominent technique, offering superior performance over traditional linear models while maintaining computational tractability compared to deep learning approaches.

XGBoost is an optimized gradient boosting framework that constructs an ensemble of decision trees sequentially, where each new tree corrects errors made by previous trees [9]. For taxi demand prediction, XGBoost typically processes concatenated demand matrices from previous n time intervals to predict future demand, effectively capturing temporal patterns through feature engineering. A comparative analysis integrating Fourier Transform features with XGBoost demonstrated noticeable improvements in prediction accuracy compared to traditional time series baselines, with Mean Absolute Percentage Error (MAPE) reduced to 11.6% [33]. The method's strength lies in its capacity to automatically identify non-linear feature interactions and its inherent feature importance ranking, which provides interpretability regarding which temporal and external features most strongly influence demand.

In the context of taxi demand prediction, XGBoost has been effectively combined with preprocessing techniques such as density-based hotspot identification - researchers have employed XGBoost and DBSCAN clustering combination to first identify high-demand zones, then apply XGBoost for zone-specific demand prediction [2]. Vanichrujee et al. developed an ensemble model combining LSTM, GRU, and XGBoost for Bangkok taxi demand, with XGBoost showing competitive performance particularly for capturing demand variations across different zone function types (residential, commercial, airport, etc.) [41].

XGBoost offers several practical advantages. It demonstrates robustness to outliers and missing values, requires minimal data preprocessing compared to neural networks, and provides built-in regularization that mitigates overfitting. Training time is relatively fast compared to deep learning architectures. However, significant limitations persist - XGBoost exhibits high data dependency, its performance degrades substantially with limited historical data [33].

1.3 Advanced deep learning techniques - Long Short-Term Memory, Convolutional LSTM

Deep learning architectures have improved spatiotemporal prediction by explicitly modelling temporal dependencies through recurrent mechanisms and spatial dependencies through convolutional and graph-based operations. These approaches address fundamental limitations of classical and machine learning methods by capturing complex non-linear patterns and enabling end-to-end learning of feature representations.

Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) address the vanishing gradient problem inherent in basic Recurrent Neural Networks (RNNs) through gating mechanisms that enable learning of long-range temporal dependencies [11]. LSTM networks have demonstrated substantial improvements over statistical baselines, with reported accuracy rates exceeding 83% in comparative studies [24].

The integration of Convolutional Neural Networks (CNNs) with recurrent architectures addresses a fundamental limitation of pure LSTM approaches: their serial processing prevents efficient learning of spatial dependencies. Thus, CNNs and LSTMs are complementary in their modelling capabilities, as CNNs are good at reducing frequency variations and LSTMs are good at temporal modelling as stated in the research of Sainath et al. [36]. ConvLSTM, specifically, embeds convolutional operations within LSTM gates, enabling the model to simultaneously process spatial and temporal dimensions in a unified framework [39].

These architectures offer substantial advantages over classical and machine learning approaches. They automatically learn feature representations from raw data, eliminating manual feature engineering for temporal patterns [18]. Recurrent mechanisms explicitly model long-range temporal dependencies without requiring manual lag selection [3]. CNN-LSTM variants capture both spatial and temporal patterns in unified frameworks, enabling joint optimization [37].

However, critical limitations persist, as RNN variants suffer from sequential processing, preventing efficient parallelization on modern hardware, resulting in slow training despite architectural advantages [34]. Also, performance tends to degrade significantly with limited historical records [7]. These models are prone to overfitting, requiring careful regularization and early stopping. Interpretability is severely limited as well, it is difficult to understand why the model makes specific predictions [22]. Training instability complicates hyperparameter tuning. Additionally, these models often assume fixed spatial structures, which may not align with actual urban geography or administrative boundaries.

Furthermore, there are many other variants of Neural Networks that proved quite fit for the prediction of transport demand, such as Graph Neural Networks [6], Temporal Convolutional Networks [25], Spatial-Temporal Convolutional Transformer Networks etc. [19]. However, they require a separate study to analyse and investigate them in full, which is not the aim of this research.

1.4 Clustering algorithms

Spatial clustering algorithms serve critical roles in taxi demand analysis by discovering natural geographic zones of similar demand patterns, identifying demand hotspots and reducing dimensionality for prediction models. Different clustering approaches offer distinct trade-offs between interpretability, computational efficiency, and discovery of complex spatial structures.

K-Means is the simplest and most widely adopted clustering approach for taxi data analysis [30]. Studies on NYC and Beijing taxi data have demonstrated that K-Means effectively identifies high-demand geographic concentrations around transportation hubs, commercial centres, and residential districts.

K-Means applications in taxi demand lack methodological guidance on optimal cluster number

selection, integration with temporal information for discovering time-dependent zones, and incorporation of demand intensity (weighted by trip frequency) into cluster optimization. The assumption of equal-variance spherical clusters proves restrictive for real urban hotspot geometries. This study tries to cover those research gaps by applying hierarchical two-step clustering to address both temporal and spatial concentrations.

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) addresses K-Means limitations by discovering clusters of arbitrary shapes without requiring predefined cluster counts. Applications to taxi trajectory data have demonstrated DBSCAN's effectiveness in discovering passenger hotspots; Zhao and colleagues compared K-Means, agglomerative hierarchical clustering, and DBSCAN, finding DBSCAN superior in identifying realistic hotspot geometries [42].

Spatiotemporal extensions of DBSCAN incorporate time as an additional dimension in distance metrics. ST-DBSCAN explicitly handles spatiotemporal data by considering both spatial proximity and temporal proximity when forming clusters [32]. This enables discovery of demand hotspots that are specific to particular time periods, capturing time-varying demand patterns. Applications using ST-DBSCAN with NYC taxi data have identified distinct hotspot evolution patterns across different hours and days of the week, enabling time-conditional demand prediction models.

Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) extends DBSCAN by constructing a dendrogram of clusters at multiple density levels, then extracting a flat clustering based on cluster stability measures [26]. HDBSCAN advantages for taxi demand analysis include automatic parameter selection (requiring only MinPts, which is less sensitive than ε), discovery of variable-density clusters, and built-in hierarchy providing multi-scale perspectives on demand structure [1]. HDBSCAN applications in taxi demand remain limited despite apparent suitability. This study addresses this research gap, applying HDBSCAN in a combination with K-means.

2 Methodology

This section describes the techniques that were considered and actually used in experimental part.

The initial idea was to simply implement K-means spatial clustering to the dataset and then apply VAR, XGBoost and CNN models. However, the final architecture turned out to be more complex and hierarchical: first applying clustering on temporal features, and then applying the hybrid approach clustering on spatial characteristics - with HDBSCAN finding the clusters of arbitrary shape and then passing the found centroids coordinated to form a K-means clusters free of noise.

Secondly, the forecasting approaches were also moved from VAR to SARIMA as a baseline model, as it captures seasonality better and due to VAR application's curse of dimensionality. For deep learning, approaches were moved from CNN to LSTM network, as it is able to capture non-linear spatiotemporal patterns.

2.1 Clustering algorithms

The first naive idea for clustering consisted of only the spatial feature, meaning simply finding the agglomerations of the naturally dense trip pickup locations, using **K-means** clustering which assigns points to the nearest centroid and iteratively updates centroids until convergence [40], with its optimization problem described as follows in equations 1, 2, 3 and 4:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{k} r_{ij} \left\| \mathbf{p}_i - \boldsymbol{\mu}_j \right\|^2 \tag{1}$$

where:

N = total number of taxi trips

k = number of clusters (predetermined)

$\mathbf{p}_i = (lat_i, lon_i)$ = pickup location coordinates for trip i

$\boldsymbol{\mu}_j$ = centroid of cluster j

r_{ij} = binary indicator variable where $r_{ij} = 1$ if point $\mathbf{p}_i$ is assigned to cluster j, and $r_{ij} = 0$ otherwise

where r_{ij} variable :

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_{1 \le j \le k} \left\| \mathbf{p}_i - \boldsymbol{\mu}_j \right\|^2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

and update step: recomputing cluster centroids as the mean of assigned points:

$$\boldsymbol{\mu}_j^{(t+1)} = \frac{\sum_{i=1}^{N} r_{ij}^{(t)} \mathbf{p}_i}{\sum_{i=1}^{N} r_{ij}^{(t)}} \tag{3}$$

until convergence criterion is met:

$$\left\| \boldsymbol{\mu}^{(t)} - \boldsymbol{\mu}^{(t-1)} \right\|^2 < \epsilon \quad \text{or} \quad t > T_{max} \tag{4}$$

The limitation of this approach is that the predetermined number of clusters has to be provided, meaning it can be biased by this selected parameter. The next, far more significant, limitation can be described as clusters being static in time, which means that for every temporal moment the clusters would be the same - the point later proved to be wrong during the exploratory data analysis.

Therefore, more advanced techniques were selected and tested to find the best approach to the problem.

DBSCAN - Density-Based Spatial Clustering of Applications with Noise is a density-based clustering algorithm designed to discover clusters of arbitrary shape and identify noise points in spatial databases, which was first introduced by Ester et al. in their article [10]. The algorithm operates on two fundamental parameters: ε, which defines the radius of a neighbourhood around a point, and MinPts, which specifies the minimum number of points required within that neighbourhood to form a dense region.

The ε-neighbourhood of a point $\mathbf{p}_i$ is defined as:

$$N_\varepsilon(\mathbf{p}_i) = \{\mathbf{q} \in D : \|\mathbf{p}_i - \mathbf{q}\| \leq \varepsilon\} \tag{5}$$

where D is the dataset of pickup locations and $\|\cdot\|$ denotes the Euclidean distance metric.

A point **q** is *directly density-reachable* from a core point **p** if **q** belongs to the ε-neighbourhood of **p**. Formally:

$$\text{DDR}(\mathbf{p}, \mathbf{q}) \equiv \mathbf{q} \in N_\varepsilon(\mathbf{p}) \wedge |N_\varepsilon(\mathbf{p})| \geq \text{MinPts} \tag{6}$$

The concept of *density-reachability* extends this through chains of core points. A point **p** is *density-reachable* from a point **q** if there exists a chain of points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n$ such that $\mathbf{p}_1 = \mathbf{q}$, $\mathbf{p}_n = \mathbf{p}$, and each $\mathbf{p}_{i+1}$ is directly density-reachable from $\mathbf{p}_i$:

$$\text{DR}(\mathbf{p}, \mathbf{q}) \equiv \exists \text{ chain } \mathbf{p}_1, \ldots, \mathbf{p}_n : \mathbf{p}_1 = \mathbf{q} \wedge \mathbf{p}_n = \mathbf{p} \wedge \bigwedge_{i=1}^{n-1} \text{DDR}(\mathbf{p}_{i+1}, \mathbf{p}_i) \tag{7}$$

Density-connectivity provides a symmetric relation connecting border points of the same cluster. A point **p** is *density-connected* to a point **q** if there exists an object **o** such that both **p** and **q** are density-reachable from **o**:

$$\text{DC}(\mathbf{p}, \mathbf{q}) \equiv \exists \mathbf{o} \in D : \text{DR}(\mathbf{p}, \mathbf{o}) \wedge \text{DR}(\mathbf{q}, \mathbf{o}) \tag{8}$$

A *cluster* C with respect to ε and MinPts is defined as a maximal set of density-connected points satisfying: (1) maximality—if a point $\mathbf{p} \in C$ and a point **q** is density-reachable from **p**, then $\mathbf{q} \in C$; (2) connectivity—all points in C are pairwise density-connected. Points not assigned to any cluster are classified as *noise*.

D = dataset of all pickup locations

$\mathbf{p}_i = (lat_i, lon_i)$ = pickup location coordinates for trip i

ε = neighbourhood radius parameter

MinPts = minimum number of points to define a core point (hyperparameter)

$N_\varepsilon(\mathbf{p}_i)$ = ε-neighbourhood of point $\mathbf{p}_i$

DDR = directly density-reachable relation

DR = density-reachable relation

DC = density-connected relation

C = a cluster in the final partition

In the context of taxi pickup locations, DBSCAN identifies natural agglomerations without requiring a predetermined number of clusters, in contrast to K-means. However, a critical limitation emerges when applying DBSCAN to spatiotemporal taxi data: the algorithm produces static clusters that remain invariant across temporal dimensions. A given location's cluster assignment remains identical regardless of time of day, day of week, or other temporal factors. This temporal rigidity makes DBSCAN unsuitable for capturing time-dependent variations in taxi demand hotspots, as pickup patterns exhibit significant fluctuations between morning rush hours and late evening, or between weekdays and weekends.

ST-DBSCAN - Spatiotemporal Density-Based Clustering - extends the DBSCAN framework to simultaneously consider spatial and temporal dimensions, enabling the discovery of clusters whose characteristics vary across time [4]. Rather than using a single distance parameter ε, ST-DBSCAN employs two distance parameters: ε_1 for spatial dimensions and ε_2 for temporal dimensions.

For taxi trip data with spatial coordinates and temporal attributes, the spatiotemporal $(\varepsilon_1, \varepsilon_2)$-neighbourhood of a point $\mathbf{p}_i$ is defined as the intersection of spatial and temporal neighbourhoods:

$$N_{\varepsilon_1,\varepsilon_2}(\mathbf{p}_i) = \left\{ \mathbf{q} \in D : \left\| \mathbf{p}_i^{\text{spatial}} - \mathbf{q}^{\text{spatial}} \right\| \leq \varepsilon_1 \wedge |t_i - t_q| \leq \varepsilon_2 \right\} \tag{9}$$

where $\mathbf{p}_i^{\text{spatial}} = (lat_i, lon_i)$ denotes the spatial coordinates of pickup location i, and t_i denotes its temporal attribute.

The spatial distance component is computed using Euclidean distance:

$$\left\| \mathbf{p}_i^{\text{spatial}} - \mathbf{q}^{\text{spatial}} \right\| = \sqrt{(lat_i - lat_q)^2 + (lon_i - lon_q)^2} \tag{10}$$

The temporal distance $|t_i - t_q|$ is measured in appropriate temporal units, in this case hours.

The core concepts of density-reachability and density-connectivity are preserved in ST-DBSCAN but now incorporate both spatial and temporal constraints. A point **p** is *directly density-reachable* from a core point **q** with respect to ε_1, ε_2, and MinPts if:

$$\text{DDR}_{\text{ST}}(\mathbf{p}, \mathbf{q}) \equiv \mathbf{p} \in N_{\varepsilon_1,\varepsilon_2}(\mathbf{q}) \wedge |N_{\varepsilon_1,\varepsilon_2}(\mathbf{q})| \geq \text{MinPts} \tag{11}$$

Density-reachability extends through chains in the spatiotemporal neighbourhood space:

$$\text{DR}_{\text{ST}}(\mathbf{p}, \mathbf{q}) \equiv \exists \text{ chain } \mathbf{p}_1, \ldots, \mathbf{p}_n : \mathbf{p}_1 = \mathbf{q} \wedge \mathbf{p}_n = \mathbf{p} \wedge \bigwedge_{i=1}^{n-1} \text{DDR}_{\text{ST}}(\mathbf{p}_{i+1}, \mathbf{p}_i, \varepsilon_1, \varepsilon_2) \tag{12}$$

To address the problem of clusters with different local densities, ST-DBSCAN introduces a *density factor* parameter. The *density factor* of a cluster C is computed as:

$$\text{DF}(C) = \frac{1}{\frac{1}{|C|}\sum_{\mathbf{p}\in C} \text{density\_distance}(\mathbf{p})} \tag{13}$$

where *density_distance* for a point **p** is the ratio of maximum to minimum distances within its spatiotemporal neighbourhood:

$$\text{density\_distance}(\mathbf{p}) = \frac{\max\{\|\mathbf{p}-\mathbf{q}\| : \mathbf{q} \in N_{\varepsilon_1,\varepsilon_2}(\mathbf{p})\}}{\min\{\|\mathbf{p}-\mathbf{q}\| : \mathbf{q} \in N_{\varepsilon_1,\varepsilon_2}(\mathbf{p})\}} \tag{14}$$

A fourth parameter, Δ^*, prevents the merging of adjacent clusters that should remain separate based on temporal characteristics. When assigning a new point to an existing cluster, the algorithm compares the point's temporal attribute with the cluster's average temporal value. If the absolute difference exceeds Δ^*, the point is not assigned to the cluster:

$$\text{Assign } \mathbf{p} \text{ to } C \text{ only if } |t_p - \bar{t}_C| \leq \Delta^* \tag{15}$$

where $\bar{t}_C = \frac{1}{|C|}\sum_{\mathbf{q}\in C} t_q$ is the mean temporal attribute of cluster C.

D = dataset of all pickup locations

$\mathbf{p}_i = (lat_i, lon_i)$ = pickup location coordinates for trip i

$\mathbf{p}_i^{\text{spatial}}$ = spatial component of point $\mathbf{p}_i$

t_i = temporal attribute of point $\mathbf{p}_i$ (hour, day of week, etc.)

ε_1 = spatial neighbourhood radius parameter

ε_2 = temporal neighbourhood radius parameter

MinPts = minimum number of points to define a core point (hyperparameter)

$N_{\varepsilon_1,\varepsilon_2}(\mathbf{p}_i)$ = spatiotemporal $(\varepsilon_1, \varepsilon_2)$-neighborhood of point $\mathbf{p}_i$

DDR_{ST} = directly density-reachable relation in spatiotemporal space

DR_{ST} = density-reachable relation in spatiotemporal space

$\text{DF}(C)$ = density factor of cluster C

Δ^* = temporal separation threshold to prevent cluster merging

$\bar{t}_C$ = mean temporal attribute of cluster C

C = a cluster in the final partition

For taxi pickup locations, ST-DBSCAN identifies demand hotspots that are localized in both space and time. A cluster discovered during morning rush hours at a transit hub represents a distinct spatiotemporal entity different from evening pickups at the same location. The algorithm captures these temporal variations by requiring both spatial proximity and temporal proximity for points to be density-reachable. This capability makes ST-DBSCAN well-suited for analysing intra-day patterns and weekly variations in taxi demand dynamics.

The time complexity of ST-DBSCAN remains $\mathcal{O}(n \log n)$ on average (where n is the number of points) when spatial-temporal index structures (such as R*-trees or modified quadtrees) are employed for efficient neighbourhood queries. The modifications to DBSCAN require careful selection of the two distance parameters ε_1 and ε_2, as well as the temporal separation threshold Δ^*.

HDBSCAN - extends DBSCAN to handle clusters of varying densities [26]. Applied to taxi pickup locations, it automatically determines the number of clusters without requiring predetermined pa-

rameters like the number of clusters. The algorithms works as described in the equations 16, 17, 18, 19 below.

For each pickup location $\mathbf{p}_i$, the core distance is defined as the distance to the m-th nearest neighbour:

$$d_{\text{core}}(\mathbf{p}_i) = \text{distance to the } m\text{-th nearest neighbour of } \mathbf{p}_i \tag{16}$$

The parameter m represents the minimum number of samples in a neighbourhood and effectively incorporates local density information into the algorithm.

The mutual reachability distance between two pickup locations $\mathbf{p}_i$ and $\mathbf{p}_j$ is computed as:

$$d_{\text{mreach}}(\mathbf{p}_i, \mathbf{p}_j) = \max\left\{d_{\text{core}}(\mathbf{p}_i), d_{\text{core}}(\mathbf{p}_j), \|\mathbf{p}_i - \mathbf{p}_j\|\right\} \tag{17}$$

where $\|\mathbf{p}_i - \mathbf{p}_j\|$ is the Euclidean distance between points. The mutual reachability distance emphasizes density-based relationships by ensuring that points in low-density regions are pushed further apart.

HDBSCAN constructs a Minimum Spanning Tree (MST) from all N pickup locations using the mutual reachability distances as edge weights:

$$\text{MST} = \{(i,j) : d_{\text{mreach}}(\mathbf{p}_i, \mathbf{p}_j) \text{ is among the smallest } N-1 \text{ edges}\} \tag{18}$$

From the MST, a hierarchical dendrogram is constructed by iteratively merging the two clusters connected by the smallest mutual reachability distance. At each merge step, a new cluster hierarchy level is created, representing different density thresholds.

The stability of a cluster is computed as the sum of the differences between the distance at which points enter and leave the cluster throughout the hierarchy:

$$\text{Stability}(C) = \sum_{\mathbf{p}_i \in C} \left(\lambda_{\text{death}}(\mathbf{p}_i) - \lambda_{\text{birth}}(\mathbf{p}_i)\right) \tag{19}$$

where $\lambda_{\text{birth}}(\mathbf{p}_i)$ is the density level (inverse of mutual reachability distance) at which point $\mathbf{p}_i$ enters cluster C, and $\lambda_{\text{death}}(\mathbf{p}_i)$ is the density level at which it leaves the cluster.

Clusters are selected from the hierarchy based on their stability. Starting from leaf nodes, the algorithm works upward through the dendrogram. A cluster is selected as a final cluster if its stability is greater than the sum of its children's stabilities:

$$\text{Select } C \text{ if } \text{Stability}(C) > \sum_{C' \in \text{children}(C)} \text{Stability}(C') \tag{20}$$

All points not assigned to any selected cluster are labelled as noise.

N = total number of taxi trips

$\mathbf{p}_i = (lat_i, lon_i)$ = pickup location coordinates for trip i

m = minimum number of samples defining a neighbourhood (hyperparameter)

$d_{\text{core}}(\mathbf{p}_i)$ = core distance of point $\mathbf{p}_i$

$d_{\text{mreach}}(\mathbf{p}_i, \mathbf{p}_j)$ = mutual reachability distance between points $\mathbf{p}_i$ and $\mathbf{p}_j$
MST = minimum spanning tree of all pickup locations
$\lambda_{\text{birth}}(\mathbf{p}_i)$ = density level at which point $\mathbf{p}_i$ enters a cluster
$\lambda_{\text{death}}(\mathbf{p}_i)$ = density level at which point $\mathbf{p}_i$ leaves a cluster
C = a cluster in the hierarchy

The final approach that was selected for this work was a combination of temporal clustering using described above K-means algorithm with prior optimization of number of clusters, then applying Hierarchical DBSCAN clustering algorithm within each of those temporal clusters to find natural clusters' centroids of spatial data and reusing those as initial starting points in K-means algorithm's later refinement of clusters, resulting in a hybrid approach that can be described as innovative and experimental.

2.2 Forecasting models

2.2.1 SARIMA: Seasonal Autoregressive Integrated Moving Average

The Seasonal Autoregressive Integrated Moving Average (SARIMA) model serves as a robust baseline approach for time series forecasting, particularly when dealing with data exhibiting both trend and seasonal components. SARIMA extends the standard ARIMA framework by incorporating seasonal parameters, making it well-suited for taxi demand data which demonstrates strong intra-day and weekly patterns [17]. The SARIMA model is denoted as:

$$ARIMA(p, d, q) \times (P, D, Q)_s \tag{21}$$

where:
p = order of non-seasonal autoregressive component
d = degree of non-seasonal differencing
q = order of non-seasonal moving average component
P = order of seasonal autoregressive component
D = degree of seasonal differencing
Q = order of seasonal moving average component
s = seasonal period length
The general mathematical formulation without a constant term is expressed as:

$$\Phi(B^s)\phi(B)\nabla_s^D\nabla^d y_t = \Theta(B^s)\theta(B)\varepsilon_t \tag{22}$$

where:
y_t = time series observation at time t
B = backshift operator such that $By_t = y_{t-1}$
$\nabla^d = (1 - B)^d$ = ordinary differencing operator applied d times
$\nabla_s^D = (1 - B^s)^D$ = seasonal differencing operator applied D times
s = seasonal period (e.g., 24 for hourly taxi data with daily seasonality)
ε_t = white noise error term at time t

The non-seasonal autoregressive polynomial is defined as:

$$\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \tag{23}$$

The non-seasonal moving average polynomial is:

$$\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q \tag{24}$$

The seasonal autoregressive polynomial is:

$$\Phi(B^s) = 1 - \Phi_1 B^s - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps} \tag{25}$$

The seasonal moving average polynomial is:

$$\Theta(B^s) = 1 + \Theta_1 B^s + \Theta_2 B^{2s} + \cdots + \Theta_Q B^{Qs} \tag{26}$$

Parameter identification for SARIMA models relies on autocorrelation function (ACF) and partial autocorrelation function (PACF) analysis. The seasonal components (P, D, Q) are identified by examining autocorrelations at seasonal lags (e.g., lags 24, 48, 72 for hourly data with daily seasonality), while non-seasonal components (p, d, q) are identified from non-seasonal lags. The parameter d (ordinary differencing) is determined by visual inspection or unit root tests to ensure stationarity, while D (seasonal differencing) addresses seasonal non-stationarity.

For taxi demand forecasting, SARIMA captures the hierarchical nature of temporal patterns: morning rush hour effects repeat with approximately 24-hour periodicity, weekly patterns manifest in day-of-week variations. The multiplicative form of seasonal and non-seasonal components (Equation 22) allows these patterns to interact naturally. For example, morning peak intensity may scale proportionally with overall demand trends, which SARIMA accommodates through the multiplicative structure.

The primary limitation of SARIMA as a baseline model is its assumption of linear relationships between past values and future observations. Taxi demand dynamics often exhibit non-linear patterns: demand response to weather conditions, special events, or other factors. Additionally, SARIMA cannot directly incorporate exogenous variables, such as weather or event indicators, without extensions like SARIMAX, and lacks the capacity to capture complex spatiotemporal interactions across multiple pickup locations.

2.2.2 XGBoost: Extreme Gradient Boosting for Time Series

XGBoost is an efficient implementation of gradient boosting that has become widely adopted for regression and classification tasks, including time series forecasting [5]. When applied to taxi demand forecasting, XGBoost operates on lagged features constructed from the time series, transforming the temporal prediction problem into a supervised learning task. The model learns non-linear mappings from past observations to future demand values.

The XGBoost ensemble model uses K additive decision trees to produce predictions:

$$\hat{y}_i = \varphi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{27}$$

where $\mathcal{F} = \{f(x) = w_{q(x)}\}$ denotes the space of regression trees with $q : \mathbb{R}^m \to \{1, \ldots, T\}$ representing tree structure and $w \in \mathbb{R}^T$ representing leaf weights.

The regularized objective function to be minimized is:

$$\mathcal{L}(\varphi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{28}$$

where the regularization term is:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|\mathbf{w}\|^2 \tag{29}$$

Here l is a differentiable convex loss function, T is the number of leaves in tree f_k, γ is the leaf complexity penalty, and λ is the L2 regularization parameter on leaf weights.

Model training proceeds iteratively, at each iteration t, the objective function to minimize is:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{30}$$

Using second-order Taylor approximation of the loss function around the previous prediction $\hat{y}^{(t-1)}$:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{31}$$

where the first and second order gradient statistics are:

$$g_i = \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}}, \quad h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial (\hat{y}^{(t-1)})^2} \tag{32}$$

Removing constant terms and defining $I_j = \{i : q(x_i) = j\}$ as the instance set of leaf j, the simplified objective becomes:

$$\tilde{\mathcal{L}}^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T \tag{33}$$

For a fixed tree structure, the optimal leaf weight is computed analytically as:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \tag{34}$$

The gain from splitting a leaf into left and right child nodes is quantified as:

$$\text{Gain} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{35}$$

n = number of training examples
m = number of input features
K = number of boosting iterations (trees)
$\mathcal{F}$ = hypothesis space of regression trees
$q(x)$ = tree structure mapping input to leaf index
T = number of leaves in a single tree
w = vector of leaf weights/scores
$l(\hat{y}, y)$ = loss function (e.g., squared error for regression)
γ = complexity penalty coefficient for tree depth/leaf count
λ = L2 regularization coefficient for leaf weights
g_i, h_i = first and second order gradient statistics for sample i
I_j = set of sample indices in leaf j

When applied to taxi demand forecasting, XGBoost constructs feature matrices from historical demand sequences. For instance, a sliding window of the previous 6 hours of taxi pickups becomes input features, with the next hour's demand as the target. XGBoost then learns complex non-linear decision boundaries separating demand patterns. Its advantages include automatic handling of non-linearities without explicit feature engineering, as well as robustness to feature scaling, built-in regularization preventing overfitting and computational efficiency even with large sets. However, XGBoost is fundamentally a non-temporal model: it does not maintain internal state and must have temporal dependencies explicitly engineered as lagged features, potentially losing implicit temporal patterns that recurrent architectures can capture.

2.2.3 LSTM: Long Short-Term Memory Networks for Spatiotemporal Forecasting

Long Short-Term Memory (LSTM) networks represent a specialized recurrent neural network architecture designed to capture non-linear temporal dependencies and spatiotemporal patterns in time series data. LSTMs address the vanishing gradient problem inherent in standard recurrent neural networks, enabling effective learning across long sequences with time lags exceeding hundreds of steps [18].

An LSTM layer processes sequential data through memory cells and gating mechanisms that regulate information flow. At each time step t, the network maintains a cell state $\mathbf{c}_t$ (long-term memory) and a hidden state $\mathbf{h}_t$ (short-term memory). Three multiplicative gates control information flow: the forget gate determines which information to discard from the previous cell state, the input gate regulates which new information enters the cell, and the output gate controls which information is exposed to the next layer or time step.

The forget gate activation at time t is computed as:

$$\mathbf{f}_t = \sigma(W_f \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f) \tag{36}$$

The input gate activation is:

$$\mathbf{i}_t = \sigma(W_i \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i) \tag{37}$$

A candidate cell state is generated by:

$$\tilde{\mathbf{c}}_t = \tanh(W_c \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c) \tag{38}$$

The cell state is updated as:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{39}$$

The output gate activation is:

$$\mathbf{o}_t = \sigma(W_o \cdot [\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{40}$$

Finally, the hidden state (and layer output) is computed as:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{41}$$

where σ denotes the sigmoid activation function, tanh is the hyperbolic tangent function, $\odot$ represents element-wise multiplication, W_f, W_i, W_c, W_o are learnable weight matrices, $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o$ are bias vectors, $\mathbf{h}_{t-1}$ is the hidden state from the previous time step, and $\mathbf{x}_t$ is the input at the current time step.

t = current time step
$\mathbf{x}_t \in \mathbb{R}^{d_{in}}$ = input vector at time t
$\mathbf{h}_t \in \mathbb{R}^H$ = hidden state vector (short-term memory)
$\mathbf{c}_t \in \mathbb{R}^H$ = cell state vector (long-term memory)
$\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t \in (0,1)^H$ = forget, input, output gate activations
$\tilde{\mathbf{c}}_t \in (-1,1)^H$ = candidate cell state
$W_f, W_i, W_c, W_o \in \mathbb{R}^{H \times (H+d_{in})}$ = learnable weight matrices
$\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_c, \mathbf{b}_o \in \mathbb{R}^H$ = bias vectors
H = hidden state dimension (hyperparameter)
σ = sigmoid activation function: $\sigma(x) = 1/(1 + e^{-x})$
tanh = hyperbolic tangent activation: $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$
T = sequence length

The key innovation addressing the vanishing gradient problem lies in the architecture: the cell state $\mathbf{c}_t$ maintains constant error flow (the forget gate acts multiplicatively on previous state, not on gradients). This allows gradients to propagate backward through many time steps without exponential decay, enabling the network to learn long-term dependencies. When forget gate activations remain close to 1.0, errors flow unchanged through the cell state; when they approach 0.0, irrelevant information is discarded.

For taxi demand forecasting, LSTM architectures typically stack multiple layers: an input layer processes raw temporal sequences (e.g., hourly pickup counts for the past 72 hours), multiple LSTM layers extract hierarchical spatiotemporal patterns, and a dense output layer produces predictions. The architecture can incorporate spatial information by maintaining separate LSTM sequences for different taxi zones or clusters discovered by the earlier clustering phase. An encoder-decoder struc-

ture may be employed where encoder LSTMs summarize the input sequence context and decoder LSTMs generate multi-step-ahead forecasts.

The computational complexity of LSTM is $\mathcal{O}(T \cdot H^2)$ per training epoch, where T is sequence length and H is hidden dimension size. Training employs backpropagation through time (BPTT), computing gradients by unrolling the network across time steps and applying the chain rule backward. Common challenges include (1) *gradient clipping* to prevent exploding gradients during backpropagation; (2) *dropout regularization* applied to inputs and recurrent connections to prevent overfitting; and (3) *early stopping* based on validation loss to avoid memorizing training data.

In the context of this thesis, LSTMs are employed as the primary deep learning baseline, leveraging their demonstrated capacity to capture non-linear spatiotemporal dynamics. Their strength lies in discovering implicit temporal patterns from data without manual feature engineering, contrasting with XGBoost's requirement for explicit lagged feature construction. The hybrid clustering-forecasting architecture proposed in this work positions LSTM models to operate on cluster-specific demand sequences, potentially improving performance by decomposing complex city-wide patterns into more regular cluster-level behaviours.

In the current study LSTM architecture comprises 2 LSTM layers with reduced hidden units (256 to 128), the equation for which can be seen in equation 43. Following Mishra and Murthy [27], this progressive compression enables the model to extract hierarchical spatiotemporal patterns characteristic of urban taxi dynamics. Two dense layers further compress the recurrent representations before generating cluster probability outputs. The first LSTM layer comprises 256 hidden units, equal to the 256-dimensional input space (PCA-reduced features). The progressive reduction through subsequent layer creates a bottleneck that forces the model to extract the most informative features for demand prediction, following the information bottleneck principle [38].

$$\text{units}_i = \frac{\text{lstm\_units}}{2^i}, \quad i = 0, 1, \ldots, N-1 \tag{42}$$

Input shape consists of sequence of 168 (representing hourly prediction for one week) and dimensionality of 256 after PCA reduction, discussed further in optimization section. Output shape dimension is 533, which is equal to predicted clusters count. Output values represent cluster probability distributions for demand prediction. Variational dropout is applied within LSTM layers (recurrent_dropout=0.2) based on the studies by Gal & Ghahramani [14] and Chen et al. [8].

$$\text{h}_t = \text{LSTM}(x_t; W, b) \odot m \tag{43}$$

2.2.4 Evaluation criteria

Described below are the evaluation criteria for the forecasting of time-series models, and the selection of metrics that were used in the current research.

MAE is Mean Absolute Error - an interpretable metric in original units that is robust to outliers. The mathematical formulation expresses it as the average absolute deviation from actual values 44.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{44}$$

where:

- y_i denotes the actual (observed) value at time step i
- $\hat{y}_i$ denotes the predicted value at time step i
- n is the total number of observations

Since MAE is expressed in the same units as the original target variable - the count of taxi demand - it is easy to interpret and compare. And because this metric treats all errors with equal weight, it makes it less sensitive to outliers. On the other hand, those can be seen as disadvantages, since it does not penalise the larger errors more severely.

MSE is Mean Squared Error, which computes the average of squared differences between predicted and actual values. The equation for its calculation is below in 45. By squaring the errors, metric gives higher weight to larger deviations [13].

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{45}$$

RMSE - Root Mean Squared Error - is the square root of MSE as can be seen in 46. It returns the metric to its original units of measurement increasing the interpretability, while assigning a large influence to large residuals [31].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{46}$$

MAPE is Mean Absolute Percentage Error - it expresses the average prediction error as a percentage of actual values. This enables comparison of forecast accuracy across different scales and magnitudes, as it is more sensible to relative variations rather than absolute. [29]

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{47}$$

R^2 - the coefficient of determination, is the proportion of the variation in the dependent variable that is predictable from the independent variable(s), defined as:

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{48}$$

where the sum of squares of residuals is:

$$\text{SS}_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{49}$$

and the total sum of squares is:

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{50}$$

Here $\bar{y}$ represents the mean of actual values:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{51}$$

Coefficient of determination indicates how well the model fits the observed data - ranges from 0 to 1, where 1 indicates a perfect fit and 0 suggests that the model does not explain the variance in the dependent variable [15].

The comparative table of evaluation metrics is provided in Table 1

Table 1 *Comparative Summary of Evaluation Metrics*

Metric	Units	Scale-Free	Outlier Sensitive	Interpretability
MAE	Original	No	Low	Very High
MSE	Squared	No	Very High	Low
RMSE	Original	No	High	High
MAPE	Percentage	Yes	Medium	High
R^2	Dimensionless	Yes	Low	High

For spatiotemporal prediction purpose such as current demand forecasting, employing multiple complementary metrics provides the most comprehensive assessment. A combination of MAE, MSE, RMSE, MAPE and R^2, as well as residual analysis, was selected because it offers sufficient coverage of performance.

3 Experimental setup

In this section the full process is described from receiving the data to the final models outputs evaluation.

3.1 Dataset, cleaning and EDA

This research employs the NYC Yellow Taxi Trip Data dataset compiled by Vishesh Mittal and made available through the Kaggle platform [28]. The dataset represents a curated subset of New York City yellow taxi trip records originally sourced from the NYC Taxi & Limousine Commission (TLC).

The dataset consists of approximately 46.2 million trip records. The temporal coverage is limited to specific months: January 2015, January-March 2016. This imposes one limitation on the current study, as the data covers only winter months, so no yearly seasonality can be explored if any.

It represents an earlier version of TLC data that retains precise GPS coordinates for pickup and drop-off locations, unlike more recent TLC releases that use aggregated taxi zone identifiers for privacy protection, thus enables fine-grained spatial analysis. The data dictionary is described in Table 2.

Table 2 *Dataset Field Descriptions*

Category	Field Name	Description
Identification	VendorID	Code indicating the TPEP provider (1 = Creative Mobile Technologies, 2 = VeriFone Inc.)
Temporal	tpep_pickup_datetime	Timestamp when the taximeter was engaged
	tpep_dropoff_datetime	Timestamp when the taximeter was disengaged
Spatial	Pickup_longitude	Longitude coordinate where meter was engaged
	Pickup_latitude	Latitude coordinate where meter was engaged
	Dropoff_longitude	Longitude coordinate where meter was disengaged
	Dropoff_latitude	Latitude coordinate where meter was disengaged
Trip Characteristics	Passenger_count	Number of passengers (driver-entered value)
	Trip_distance	Elapsed trip distance in miles reported by taximeter
	RateCodeID	Final rate code in effect at trip end
Operational	Store_and_fwd_flag	Flag indicating whether trip record was held in vehicle memory (”Y” = store and forward, ”N” = direct transmission)
	Payment_type	Numeric code for payment method
Economic	Fare_amount	Time-and-distance fare calculated by meter
	Extra	Miscellaneous extras and surcharges ($0.50 and $1 rush hour/overnight charges)
	MTA_tax	$0.50 MTA tax automatically triggered based on metered rate
	Tip_amount	Tip amount (automatically populated for credit card tips only; cash tips excluded)
	Tolls_amount	Total amount of all tolls paid during trip
	Improvement_surcharge	$0.30 improvement surcharge assessed at flag drop (began 2015)
	Total_amount	Total amount charged to passengers (excludes cash tips)

Since not all provided columns are relevant for the aim of this research (such as economic, operational type of information and partly trip characteristics), a number of them were removed from the dataset during and after the cleaning stage, as those are not providing any additional information for demand forecasting.

However, before abandoning those columns, they were part of the data cleaning process. More specifically, some values from economic variables, as well as their derivative calculated features, were used in data cleaning to get rid of outliers and to remove other illogical values (for example, Fare amount < 0, Average speed > 120 km/h, unreasonably high prices per km and so on) to ensure the erratic and inconsistent trips data are not present in the models' input dataset. Figure 1 below shows the boxplot distribution of average speed values. It can be seen how it contained unreasonable values before, and how the distribution changed after capping it to physically plausible 120 km/h ceiling.

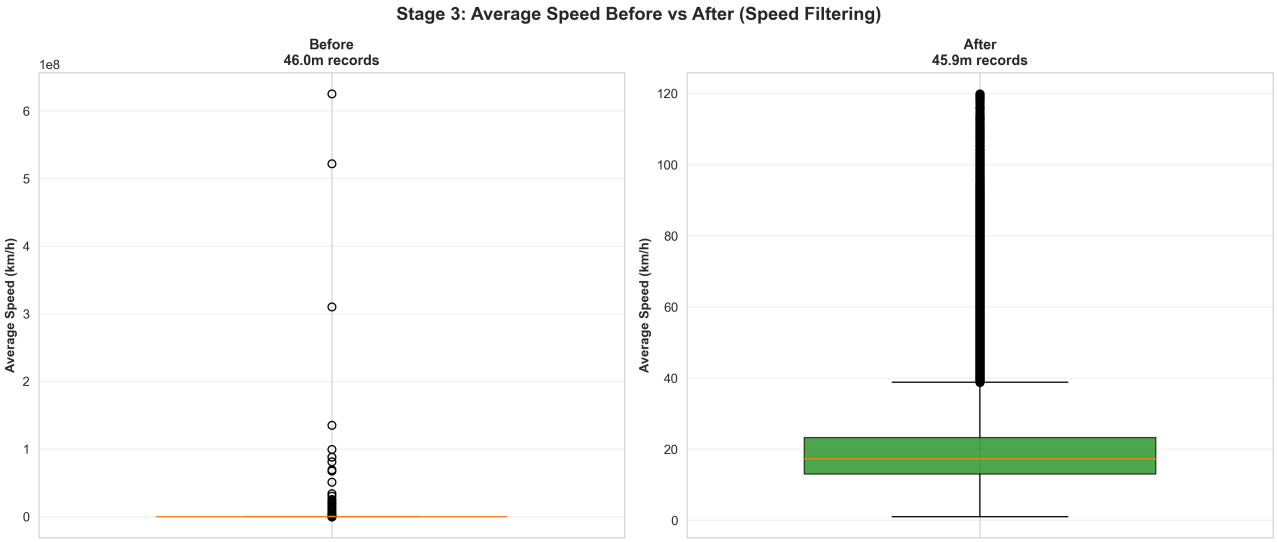

Figure 1 *Average speed boxplots*

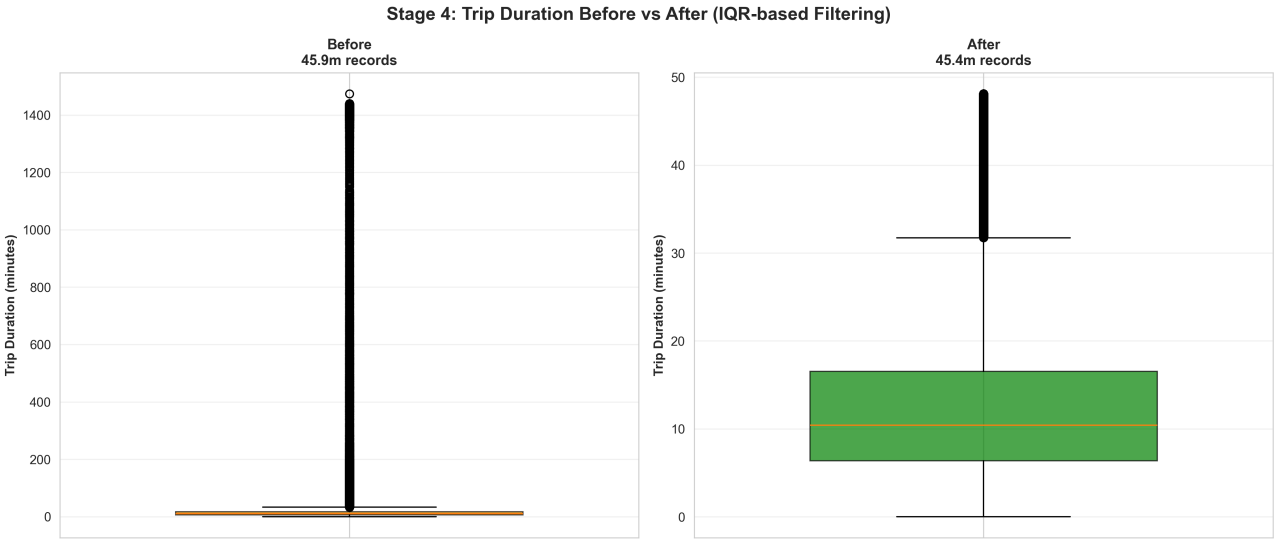

Figure 2 *Trip duration boxplots*

The same can be seen in Figure 2 for trip duration variable and in Figure 3 for trip distance - they

were reasonably capped by IQR-based filtering at 50 minutes duration and 15 kilometres distance, respectively, in order to capture only intra-city traffic movements and demand, which are relevant for this research.

For the additional layer of sanity check, some derivative fields were calculated, such as price per kilometre and price per minute, and the similar sanity filter was applied, limiting the maximum values to feasible threshold. The results can be seen in Figure 4 and Figure 5 respectively.

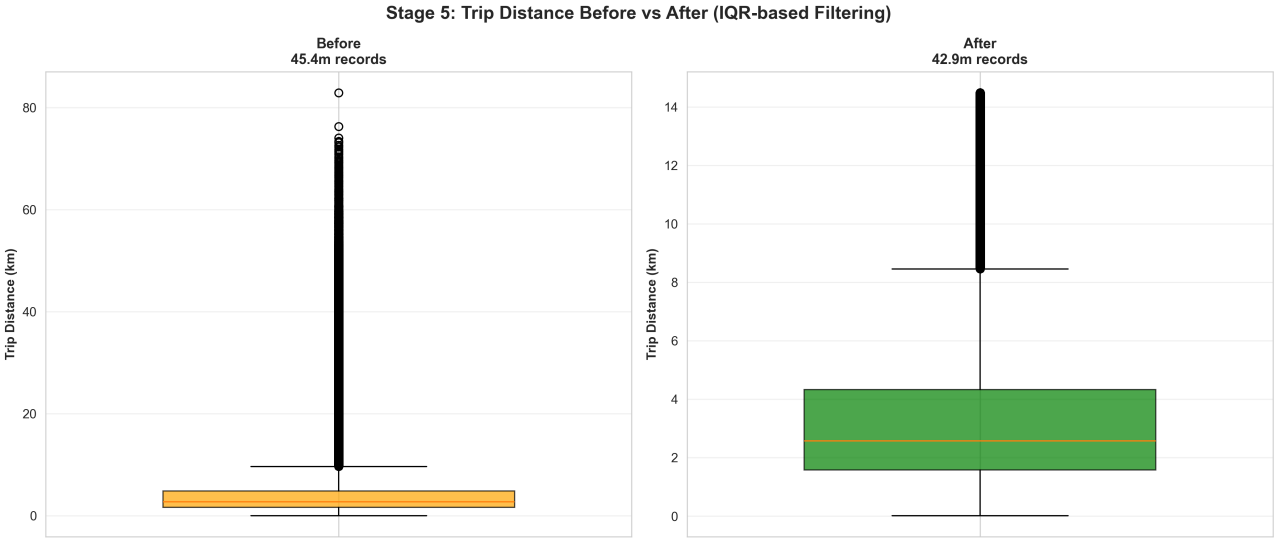

Figure 3 *Trip distance boxplots*

Figure 4 *Price per km boxplots*

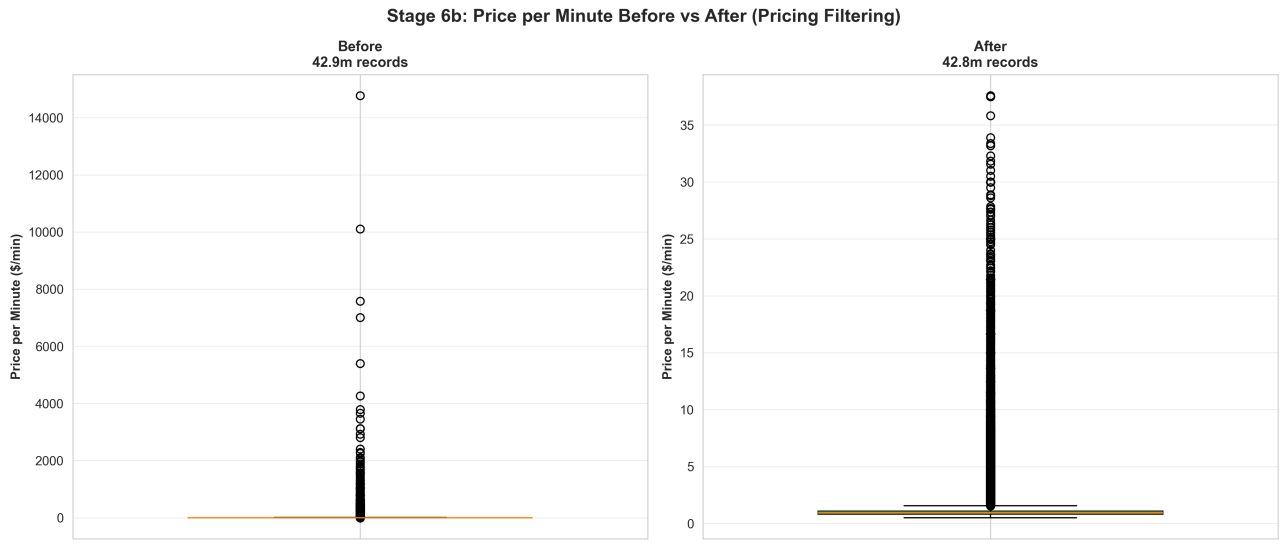

Figure 5 *Price per min boxplots*

Additional data preparation included filtering out the trips with pickup coordinates not included in the rectangle formed by New York City boundaries, i.e. pickup longitude between -74.05 and -73.75 and pickup latitude between 40.63 and 40.86. Some rows contained the exact same pickup and drop off coordinates, which were also successfully removed from the dataset.

Frequency distribution of some features (including calculated) before the cleaning can be seen in Figure 6, while the features' frequency distributions after implementing all cleaning and outliers removals can be seen in charts below in Figure 7.

It must be noted that there are some limitations to the data. For instance, the concentration of data in January-March may not adequately represent annual mobility patterns, seasonal variations, or the full spectrum of urban events that influence taxi demand throughout a year.

After the clustering algorithms discussed further were applied, the next step included the conversion of the data from initial format. The data was aggregated to form a demand matrix with 1 hour window aggregated demand per each hierarchical cluster resulting in the chronologically sorted initial dataset that can be used as an input for the models.

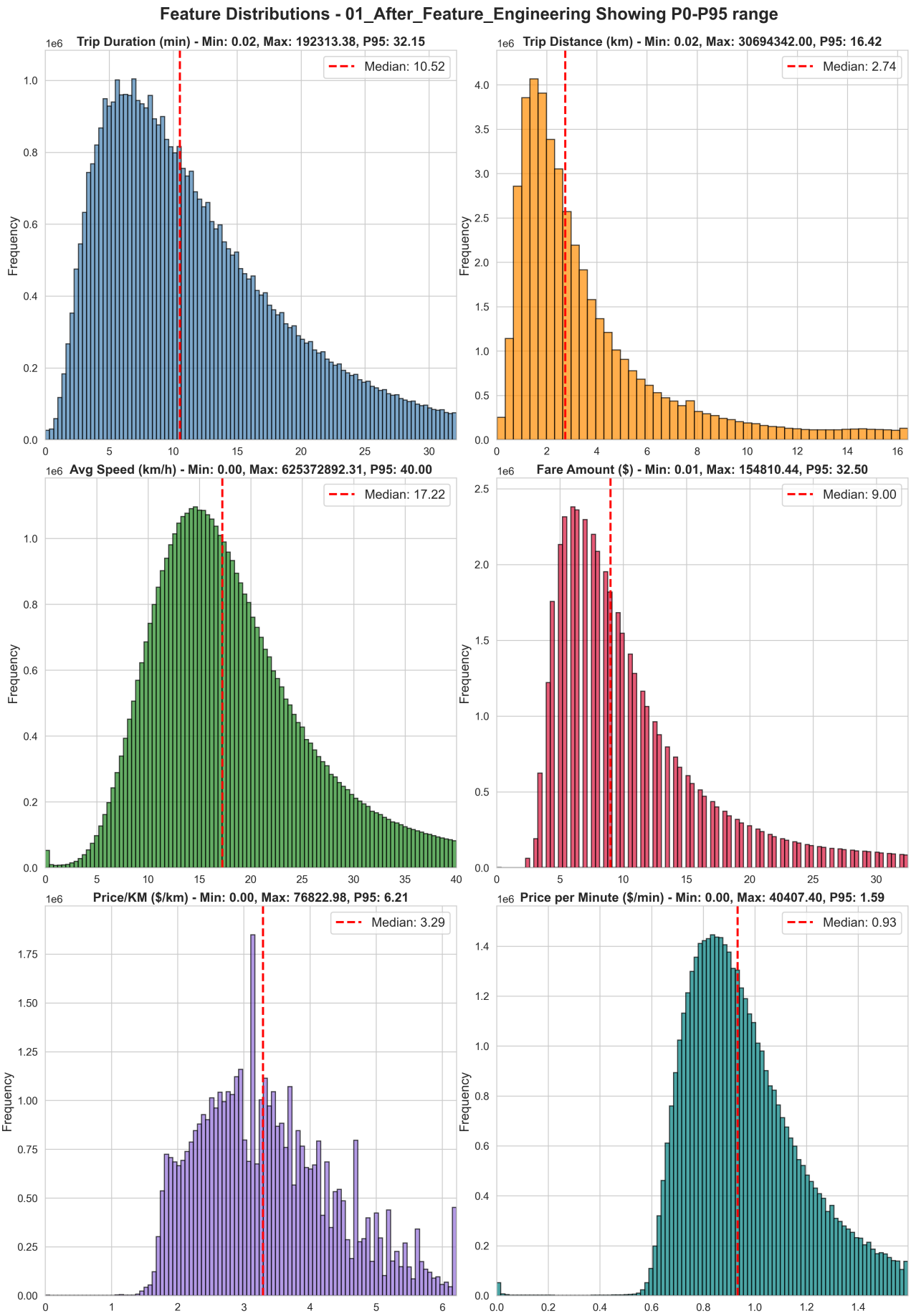

Figure 6 *Distribution of some features before cleaning and removing outliers*

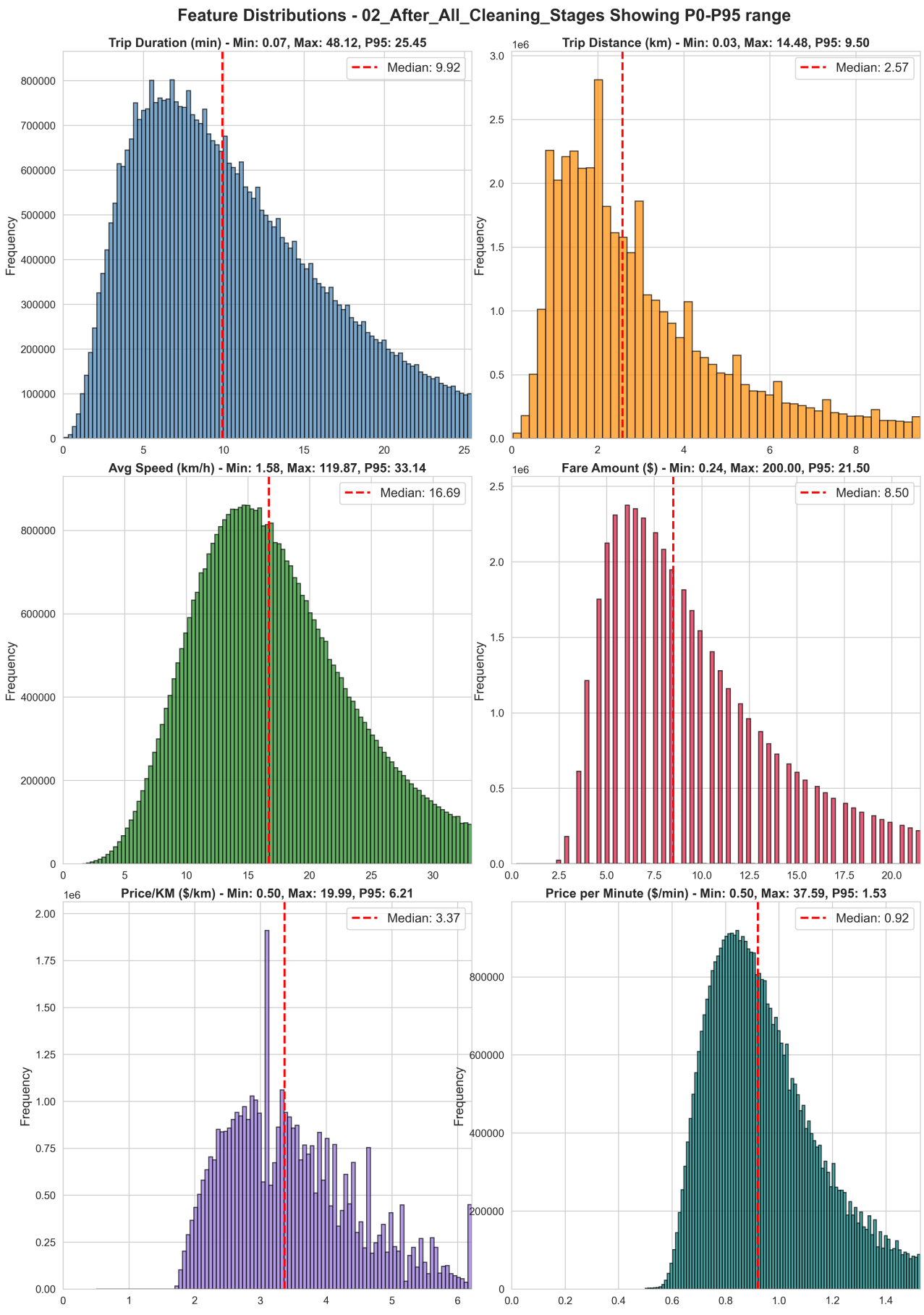

Figure 7 *Frequency distribution after cleaning*

3.2 Feature engineering

In order to provide XGBoost and LSTM models with enough input, feature engineering was applied, which is described in this section.

Transforming Temporal Features

First, simple derivative features were designed from the pickup time - hour, day of week, day of month, month - to help establish the connection with demand volumes.

Next, instead of using raw numerical values (e.g. Hour $\in [0, 24)$, Day $\in [0, 6]$), it was essential to create two new features for each temporal dimension by projecting them onto a unit circle, as was suggested in the work of Khazem and Kanso [21] . This allowed to preserve temporal cyclical proximity (e.g. 23:59 $\approx$ 00:01, and 0=Monday having the same distance to 1=Tuesday as it has from the 6=Sunday at the same time).

The formulas to transform the pickup time (converted to a continuous value representing hours and fractional minutes), as well as the day of the week, to a circle are provided below.

$$\text{Hour}_{\text{sin}} = \sin\left(\frac{2\pi \times \text{hour}}{24}\right) \tag{52}$$

$$\text{Hour}_{\text{cos}} = \cos\left(\frac{2\pi \times \text{hour}}{24}\right) \tag{53}$$

- hour $\in [0, 24)$: Pickup hour with minute fractional component
- $\text{Hour}_{\text{sin}}, \text{Hour}_{\text{cos}} \in [-1, 1]$: Sine and cosine projections on unit circle

$$\text{Day}_{\text{sin}} = \sin\left(\frac{2\pi \times \text{day}}{7}\right) \tag{54}$$

$$\text{Day}_{\text{cos}} = \cos\left(\frac{2\pi \times \text{day}}{7}\right) \tag{55}$$

- day $\in [0, 6]$: Pickup hour with minute fractional component
- $\text{Day}_{\text{sin}}, \text{Day}_{\text{cos}} \in [-1, 1]$: Sine and cosine projections on unit circle

Other temporal features engineering included calculation of two boolean flags, indicating whether it was a weekend, a rush hour (defined as pickup hour between 7-9 a.m. or 5-7 p.m.).

Lags calculation

The next stage of preparing the features consisted of calculating the lags for each temporal feature. It was selected to make 168 lag range due to weekly (168 hours) cyclicity of the data. Furthermore, additional rolling statistics were added for each feature, namely rolling mean of 6 hours, rolling standard deviation of 6 hours and rolling mean of 24 hours to help capture the temporal patterns.

3.3 Clustering algorithms

This subsection describes what ended up being the final technique that was applied to define the both temporal and spatial distinctive patterns - K-means algorithm combination with HDBSCAN.

To better understand the patterns, exploratory data analysis was performed on the data from calculated demand matrix. It showed that the temporal structure in the dataset is quite different throughout the week. This is illustrated in the heatmap graph in Figure 8, with a big concentration of demand on Thursday, Friday and Saturday evenings and downtime during night and early mornings on working days. In addition, frequency distribution plot in Figure 9 further proves different pickup hour demand patterns within weekdays and weekends.

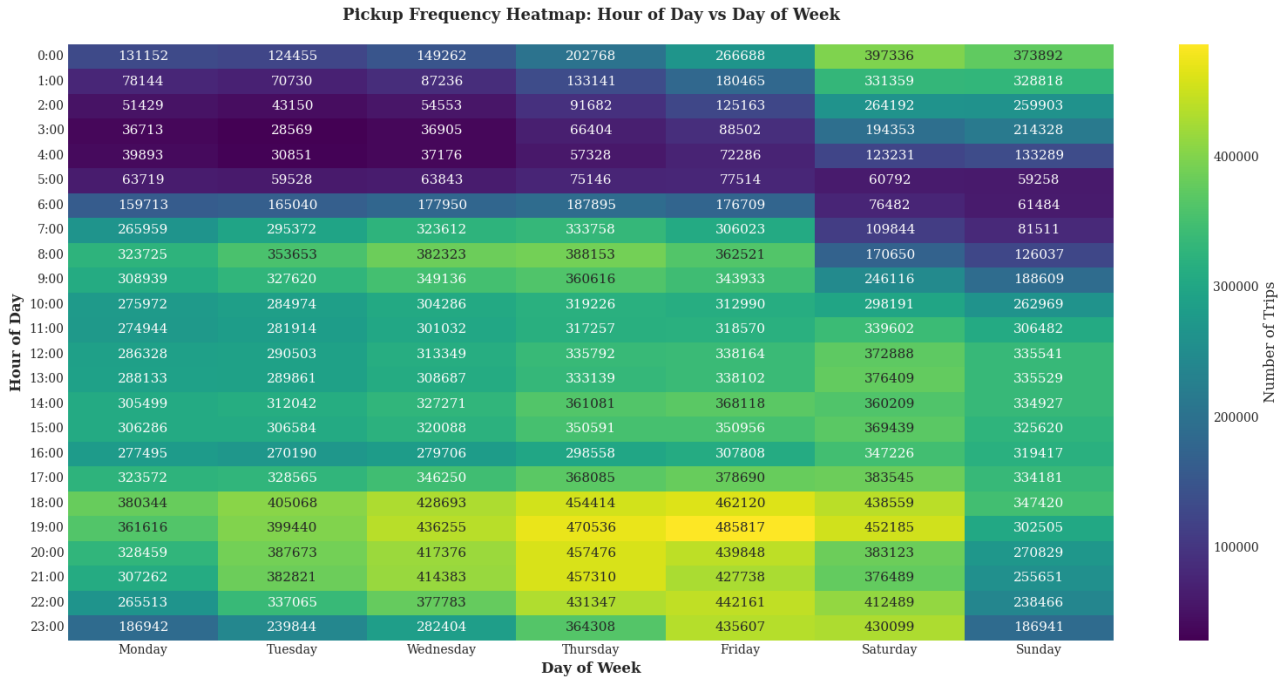

Figure 8 *Pickup time frequency heatmap*

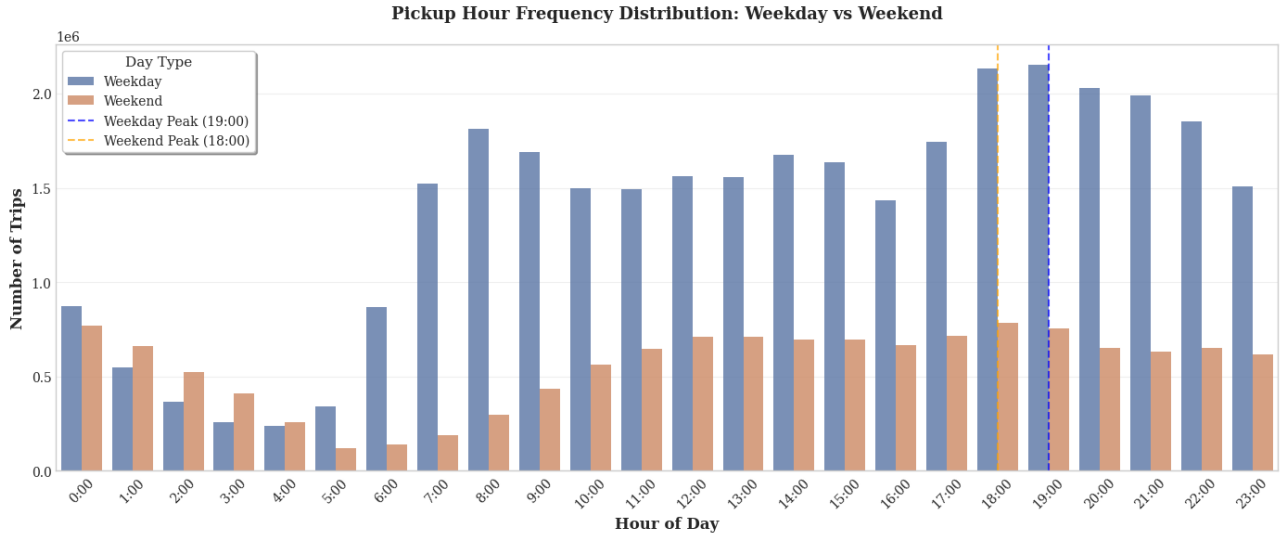

Figure 9 *Pickup time frequency distribution by weekdays*

With all this in mind, the temporal clustering seems essential to capture the behaviour during different times of week.

The initial idea was to combine both temporal and spatial features into unified spatiotemporal distance metric to be used as an input in ST-DBCSAN algorithm. However, it was not efficient due to very high density of the data, with the algorithm not able to distinguish any patterns correctly. Therefore, the combined approach was constructed to use K-means clustering first to identify temporal patterns and then apply the hybrid K-means and customised HDBSCAN to cluster on spatial factor.

Therefore, the first step, i.e. temporal clustering, was selected to be K-means algorithm with the finite number of clusters, reflecting different temporal patterns. As for the temporal clustering optimization, it is described in detail in next section together with spatial clustering optimization.

The K-means clustering on temporal data resulted in 6 temporal clusters. The temporal patterns can be described for each cluster, as can be seen in the Figure 10 summary, as well as in the Figure 14 graph.

```
TEMPORAL PATTERNS:
------------------------------------------------
T0: Weekday Morning Rush (256,226 trips, 17.08%)
T1: Weekend/Night (256,347 trips, 17.09%)
T2: Weekday Evening Rush (288,297 trips, 19.22%)
T3: Weekday Evening Rush (249,130 trips, 16.61%)
T4: Weekend/Night (203,030 trips, 13.54%)
T5: Weekday Morning Rush (246,970 trips, 16.46%)
```

Figure 10 *Temporal patterns descriptions*

It can be noticed that, for instance, the spatial division of the very southern financial district of Manhattan is different during T1 (weekend night) Figure 11 and during T0 (weekday morning rush) Figure 12. In the first case, the two spatial clusters form larger areas indicating lower demand, while in the second case, the division is finer, which can be translated as a data-driven actionable insight for a dispatcher to allocate more taxis to more hotspots during that time.

The next stage included clustering on the spatial data within each of the temporal clusters. It was selected to apply the hybrid algorithm, first to find the initial centroids of clusters with HDBSCAN, and then to use those points as initial starting points in K-means algorithm. The result and detailed process of optimization is described in the Optimization section below.

3.4 Data Splitting

70% of data was used for the training part, 10% was put to validation set, and the last 20% reserved for testing.

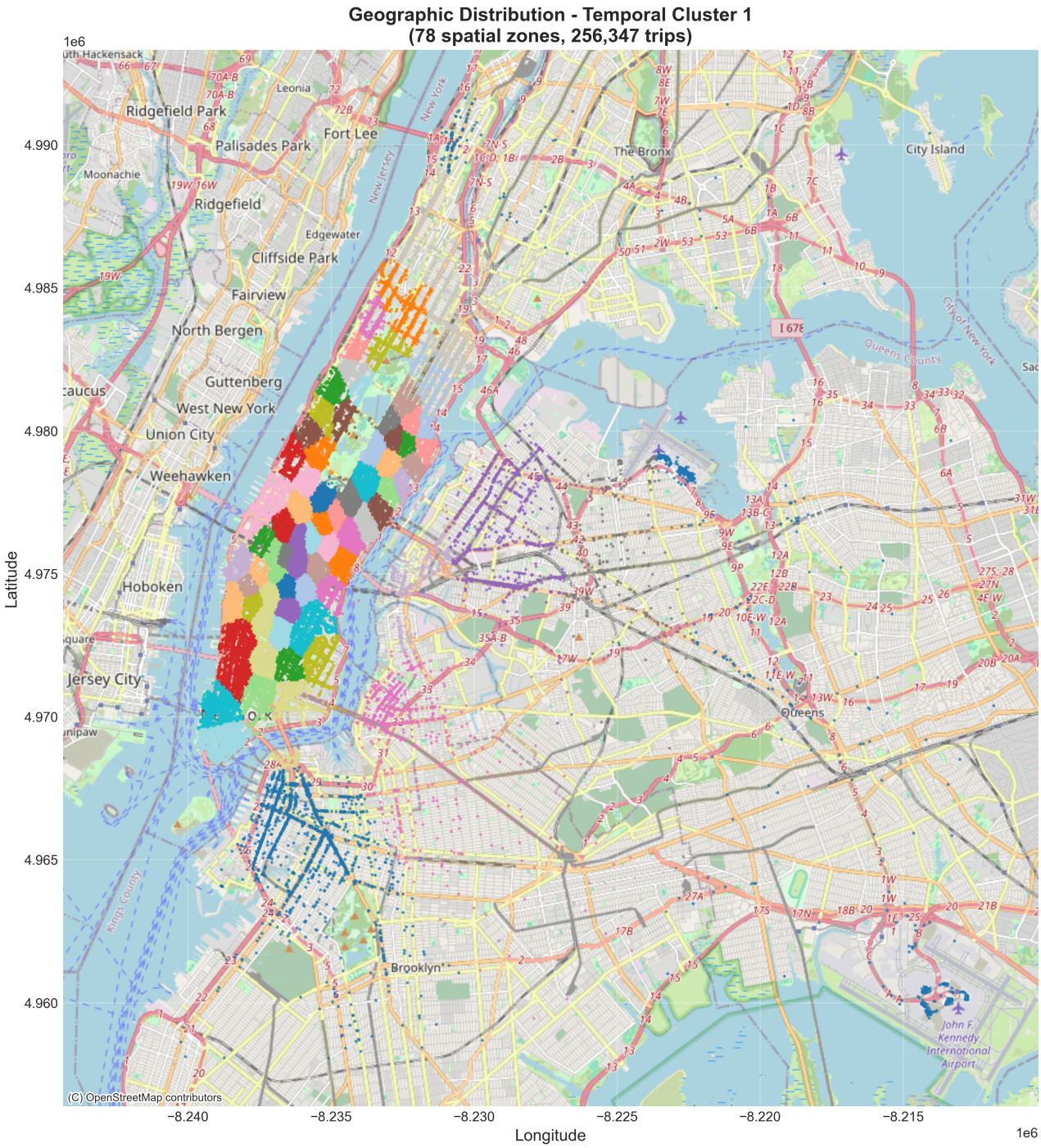

Figure 11 *Geographic distribution of cluster 1*

3.5 Models training

In the next step, the selected three forecasting models, namely SARIMA, XGBoost and LSTM, which were discussed in Methodology section, were trained on the prepared data, with the process described in detail in the below Optimization section for forecasting models. The SARIMA model used **SARIMAPredictor** class from **sarima_predictor** Python package, XGBoost was constructed as **XGBRegressor** from **xgboost** Python package with randomized grid search utilizing **RandomizedSearchCV** from **sklearn.model_selection** package.

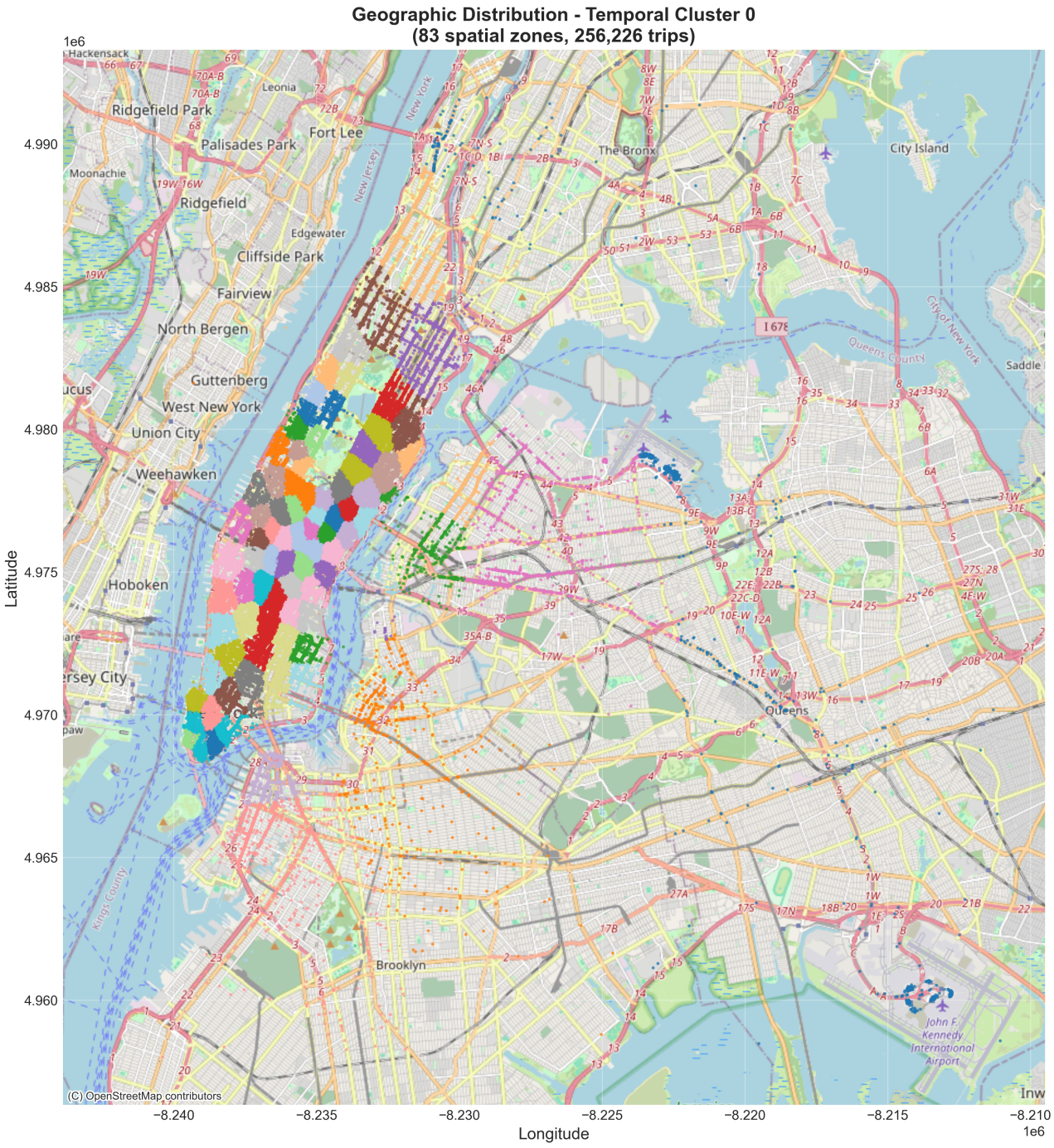

Figure 12 *Geographic distribution of cluster 0*

For LSTM model, as a part of preprocessing, the data was transformed with **sklearn.preprocessing** package's **StandardScaler**, PCA was conducted with **sklearn.decomposition** PCA tool and the model itself was built with **keras.layers.LSTM** using **keras.optimizers.Adam** optimizer and also utilizing **keras.callbacks.EarlyStopping** and **keras.callbacks.ReduceLROnPlateau** for early stopping of fitting. The custom weighted loss function was used, which is described in the Optimization section below.

3.6 Optimization

In this section, the applied optimization techniques are discussed for both clustering algorithms and forecasting models.

3.6.1 Optimization for temporal clustering

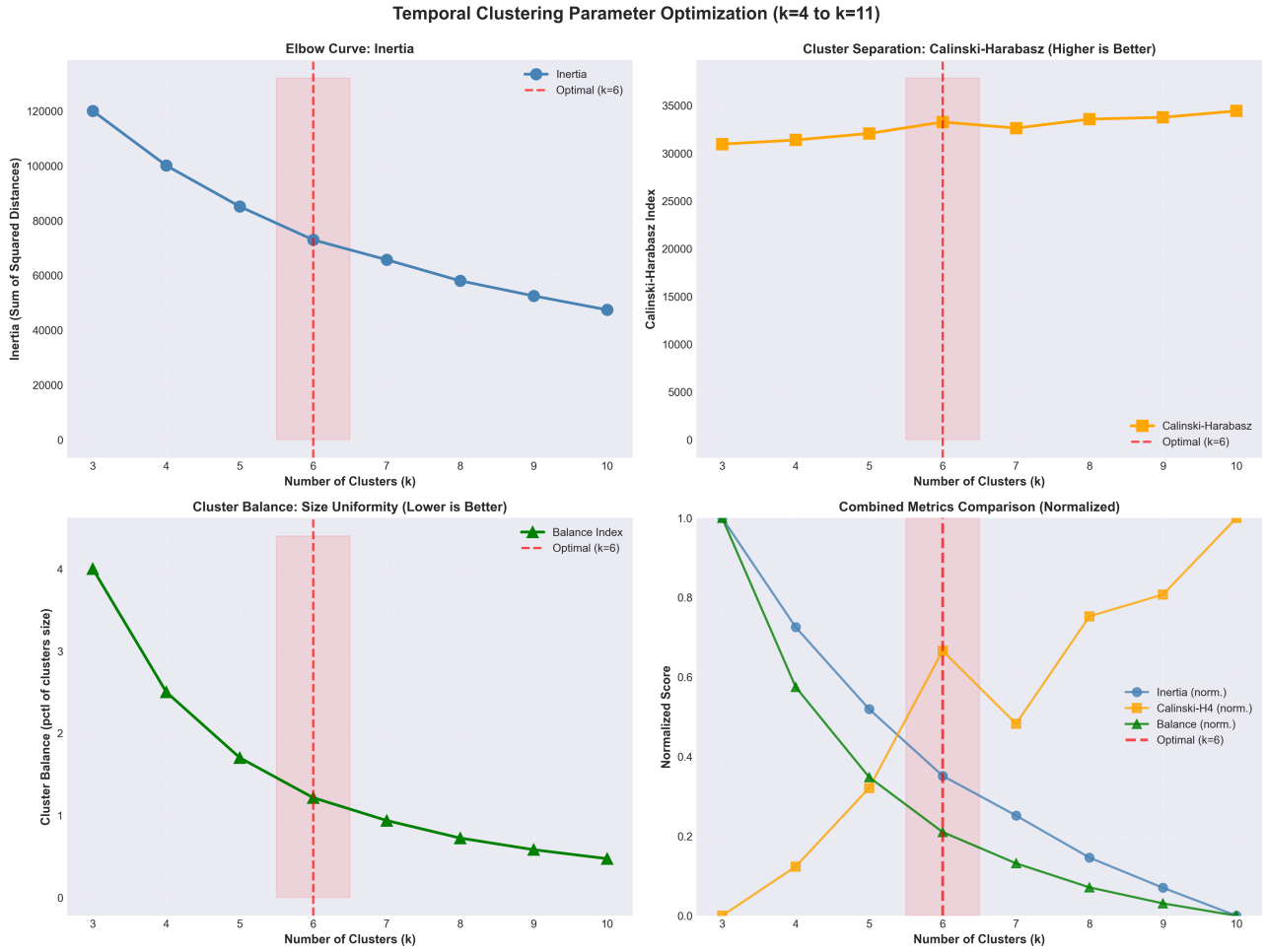

Figure 13 *Temporal clustering optimization*

For the first part of combined clustering, where K-means algorithm was applied to temporal data, it was needed to provide the finite number of clusters to capture all the patterns of time. For this, it was decided to run parameter optimization - test the clustering in the range between 4 and 11 clusters and smaller data sample of 500 000 trips to see how the metrics change. Instead of looking solely at the silhouette score (as it might be not so informative on such densely concentrated data), it was decided to evaluate how the change of numbers of cluster will change the inertia and clusters balance, as well as the calculated Calinski-Harabasz index. In Figure 13, it can be seen that, evaluating the elbow curve and finding the local maximum in the Calinski-Harabasz index, the optimal point was selected with the number of clusters being 6. The overview of the 1.5m trips sample of the final six temporal clusters can be seen in the graphs in Figure 14 - for example, the T0 cluster can be described as Weekday morning pattern with weekdays from Monday to Wednesday and pickup hours starting as

early as 5 a.m. and peaking at around 9 a.m., while the T4 cluster concentrates mostly on weekends - Saturday and Sunday evening and night, continuing into Monday early night until morning.

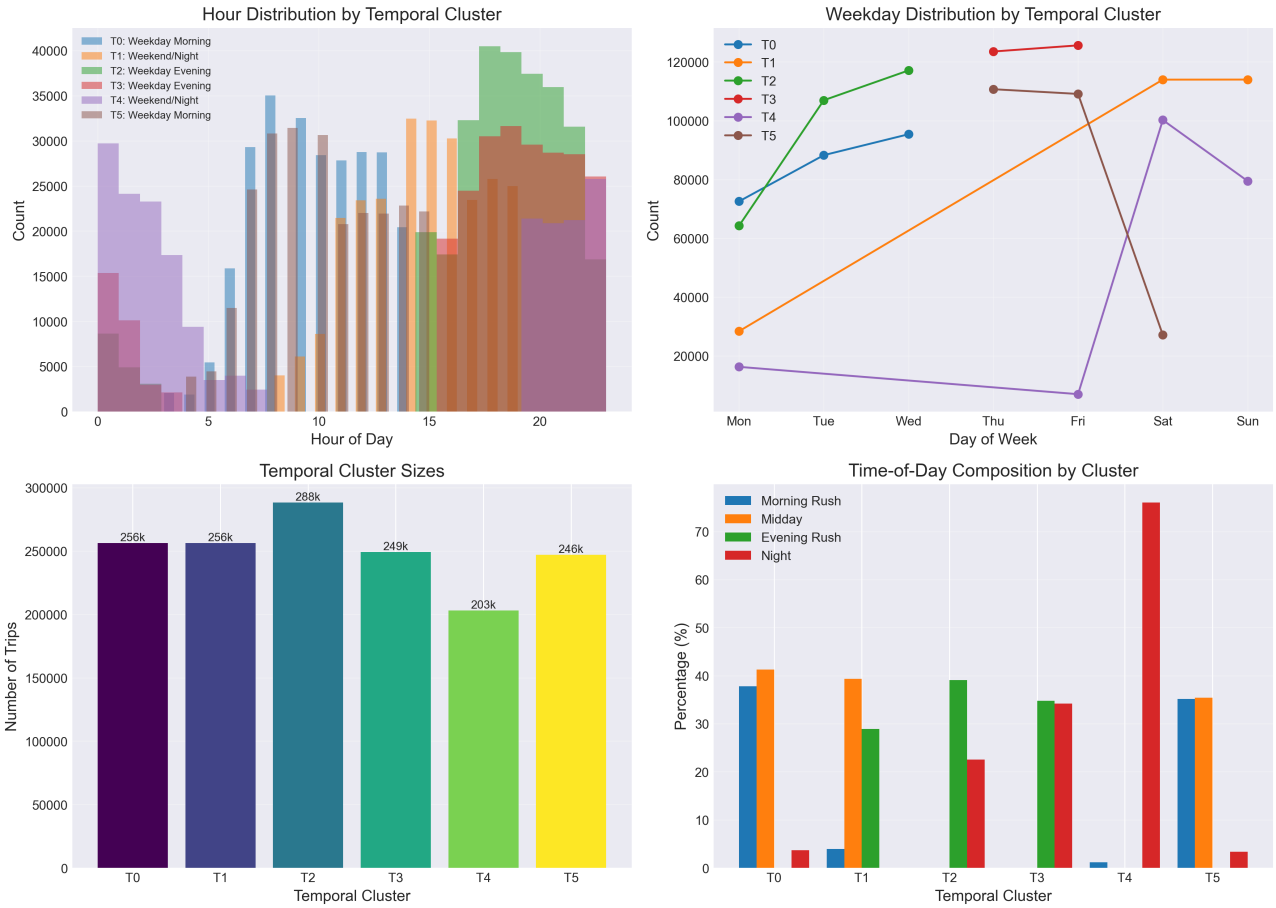

Figure 14 *Temporal clusters analysis*

3.6.2 Optimization for spatial clustering

For spatial HDBSCAN clustering for the first temporal cluster, the test run of different parameters, such as minimum cluster size and minimum samples, was also conducted. The results can be seen in the heat map in Figure 15, but that did not answer the question of what the best parameters for each model should be: due to different density of data points, clustering did not work as intended simultaneously with the same parameters on all temporal clusters.

Thus, it was decided to calculate the adjustable parameters based on the density of trips in a particular temporal cluster. The idea was to calculate the density score as 1 divided by median of pairwise distances between trips in that particular cluster, and apply different parameters for clusters of different densities. This solved the problem and made the algorithm perform well on different temporal clusters.

With this approach, it was possible to achieve similar size clusters distribution. This is illustrated in Figure 16, where in the top right graph, the top 20 clusters by size are of adequate comparable sizes. The trip characteristics for the top 15 clusters such as average distance and average fare also look reasonably similar, as can be seen in the bottom right graph. The lower left graph shows that the geographic spread in combination with cluster sizes is not that high among clusters.

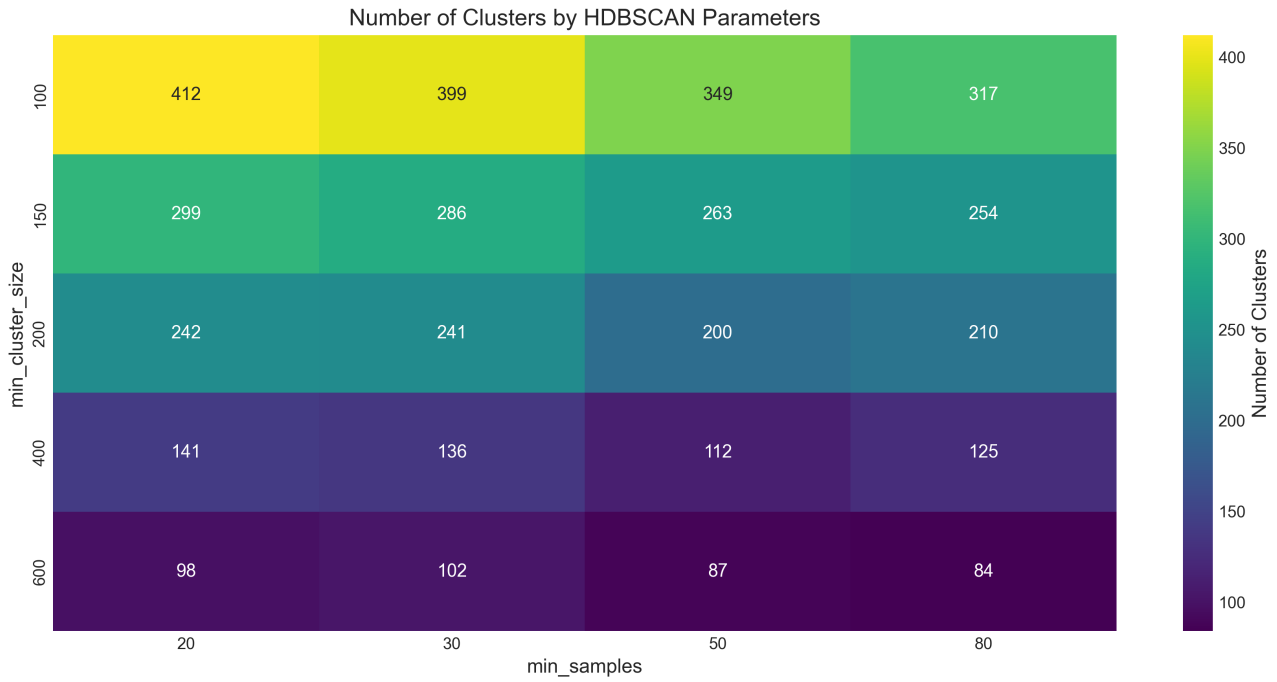

Figure 15 *Heatmap of grid search results on spatial clustering parameters*

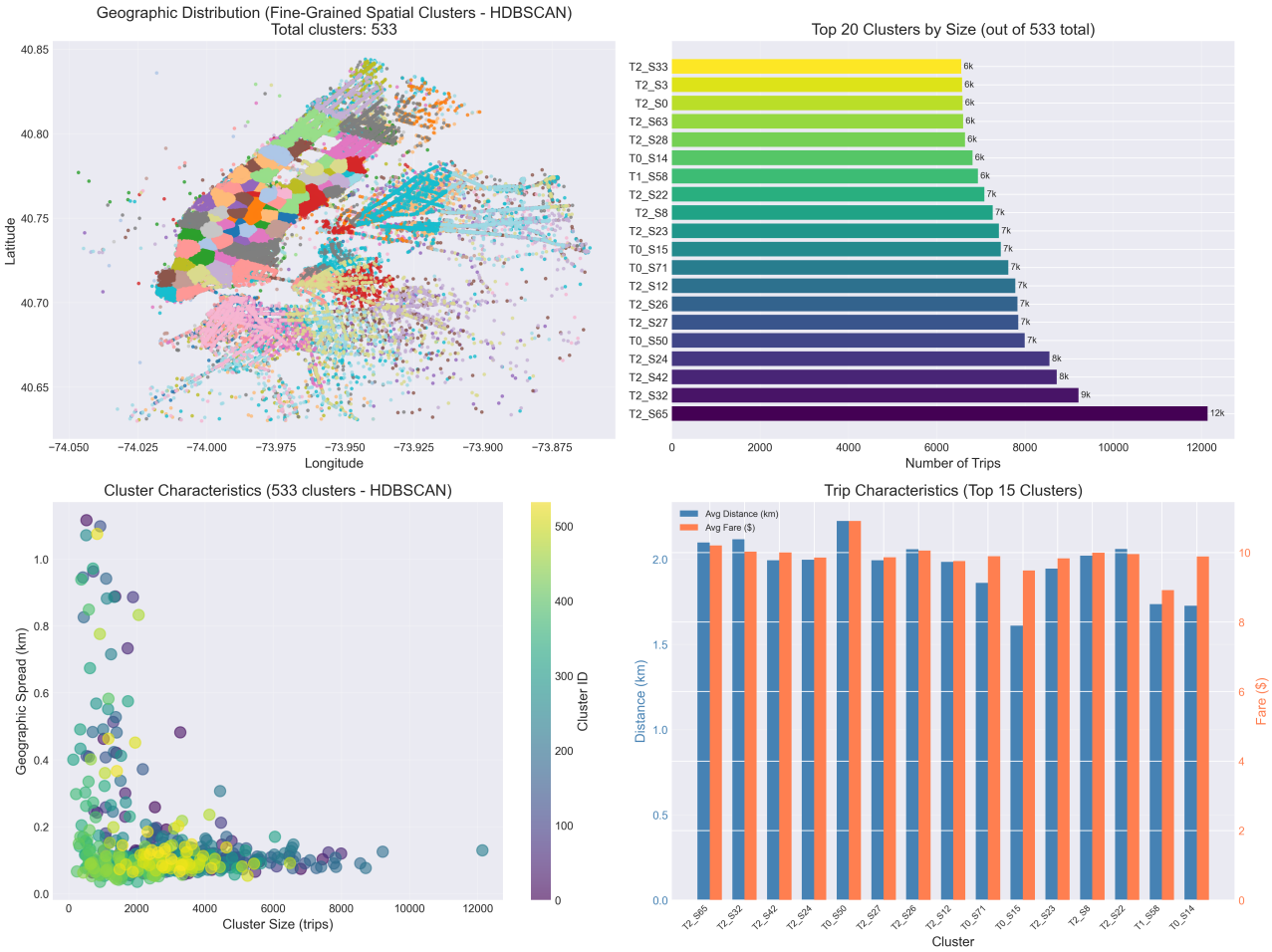

Figure 16 *Spatial clustering analysis*

3.6.3 Optimization for SARIMA

For SARIMA, a two-phase parameter optimization strategy employed.

First, to investigate the seasonality, ACF (autocorrelation function)/PACF (partial autocorrelation function) analysis was done. Augmented Dickey-Fuller (ADF) test was also performed to check stationarity on largest five clusters, get the visual representation of cyclicity of the time series and identify whether daily or weekly seasonality order seems to be the best fit. After visual inspection of spikes at lag 168, it was evident that seasonal lag of one week has the strongest signal. Also, after applying non-seasonal differencing d=1, spikes become less prominent and more noise appears, proving it to be a worse parameter than d=0. Thus, the best parameters for seasonality lag and non-seasonal differencing were selected as 168 and 0 respectively. The seasonal differencing parameter D was selected to be 1, as the data has strong seasonal patterns [16]. The graph of ACF/PACF analysis for one of the clusters can be seen in Figure 17. ADF test showed that all clusters data is stationary, thus further proving d=0 to be the better fit.

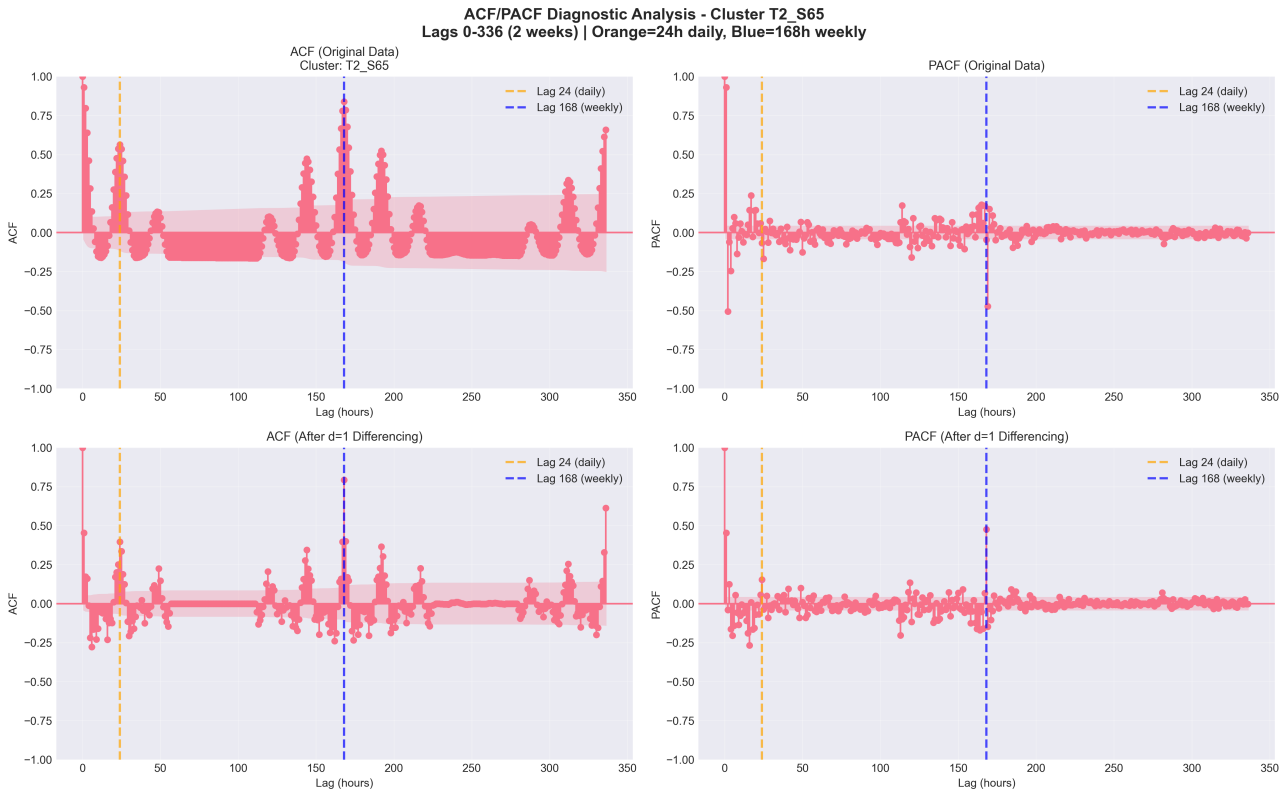

Figure 17 *ACF/PACF analysis on the top cluster*

Secondly, optimal polynomial and moving average SARIMA parameters were selected through grid search on the top three highest-demand clusters (T2_S65, T2_S42, T2_S32) for the last month of training data, evaluating 36 parameter combinations each ((p,q) $\in \mathbb{N}^2$, (P,Q) $\in \mathbb{N}^2$, with d=0, and D=1 fixed). The algorithm selected optimal SARIMA parameters per cluster using AIC criterion. The model minimizing AIC across these representative clusters - SARIMA(2, 0, 2)×(1, 1, 0, 168) from cluster T2_S65 (AIC=3784.47) - was applied uniformly to all 533 demand clusters.

3.6.4 Optimization for XGBoost

In this research, I assume that the urban mobility demand generating process has common underlying dynamics across different zones of the city. Therefore, the structural complexity of the model, which is required in order to capture these dynamics, should be stable across the spatiotemporal clusters, and transfer learning of hyperparameters can be implemented. Tuning on the most data-rich cluster ensures that the signal would be most robustly captured, avoiding overfitting to noise in smaller, sparser clusters.

With this approach, only the selected top 5 clusters by hourly demand were put into lightweight hyperparameter grid search tuning for XGBoost.

The following parameters combinations were taken:

'n_estimators': [100, 200, 300],
'max_depth': [3, 5, 7],
'learning_rate': [0.05, 0.1, 0.2],
'min_child_weight': [1, 3],
'subsample': [0.8, 1.0],
'colsample_bytree': [0.8, 1.0]

For the rest of XGBoost models the mode of the best parameters from top 5 clusters was used to create a robust set of hyperparameters. The assumption was that the parameters that work best for the top 5 heavy-traffic clusters will likely work very well for the smaller ones too.

The best combination of hyperparameters found by grid search was as follows:

'n_estimators': 300,
'max_depth': 3,
'learning_rate': 0.1,
'min_child_weight': 3,
'subsample': 0.8,
'colsample_bytree': 0.8

3.6.5 Optimization for LSTM

When applying LSTM model, a number of optimizations was implemented to increase the performance of the model.

First, the dimensionality reduction technique, namely Principal Components Analysis (PCA), was applied due to high number of features. Contrary to XGBoost, LSTM does not benefit from high number of input features, thus their number needs to be decreased for the model to work properly. PCA reduced the number of features from more than 91 thousand to 256 components, effectively capturing more than 98% of variance, as can be seen in Figure 18. This validates the assumption that NYC taxi demand exhibits strong spatial autocorrelation - neighbouring zones show correlated patterns. Even with the number of components at 100, the variance explained would already be a sufficient 95%, but in computational terms, it did not add much to the training time to capture more, which is why it was decided to select a larger number of components (256).

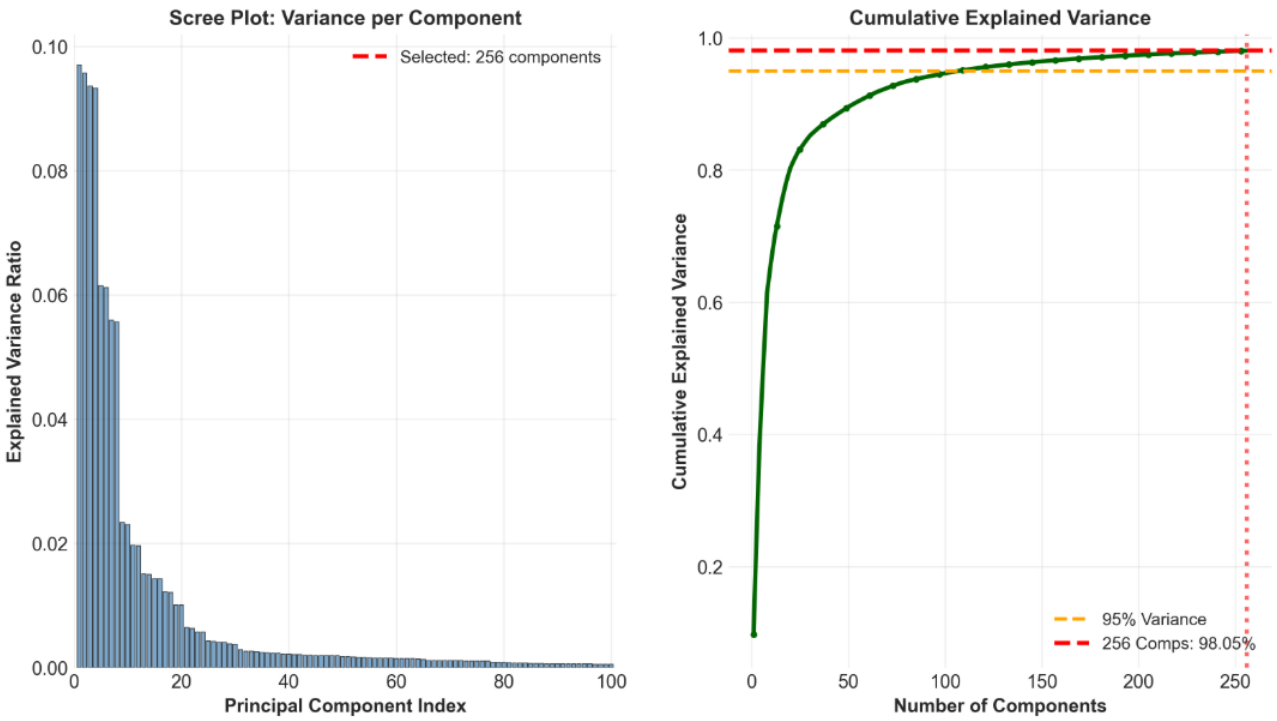

Figure 18 *PCA for LSTM model training*

Standard MSE loss treats all prediction errors equally, which is suboptimal for taxi demand forecasting. Therefore, for better peak prediction, LSTM architecture employs a weighted mean squared error loss function that emphasizes peak demand prediction errors with 3× weight compared to off-peak periods, as can be seen in 56. The peaks are defined as points with actual demand value > 75th percentile. This reflects the practical importance of accurately forecasting taxi demand during rush hours. The mixed-precision (FP16) training enables convergence within 60-100 epochs via early stopping with validation loss monitoring. This helped penalize the loss during peaks, allowing the model to predict the patterns more accurately.

$$\mathcal{L}_{\text{weighted}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot (y_i - \hat{y}_i)^2 \tag{56}$$

Where the weight w_i is determined by: $w_i = \begin{cases} 3.0 & \text{if } y_i > Q_{75} \\ 1.0 & \text{otherwise} \end{cases}$

3.7 Models evaluation and comparison

In this section, the comparison of the models is done, using the evaluation metrics described in the Methodology section. It presents the empirical evaluation of the optimized SARIMA, XGBoost, and LSTM models applied to the 533 spatiotemporal clusters derived from the NYC Yellow Taxi dataset. The models were assessed using MAE, RMSE, and MAPE metrics across 1-day (24h), 3-day (72h), and 7-day (168h) forecasting horizons on the held-out test set, enabling direct comparison of predictive accuracy and degradation patterns.

Figure 19 summarizes model performance aggregated across all clusters. XGBoost achieves the lowest errors across clusters (e.g., RMSE=8.65 at 24h), outperforming LSTM (RMSE=14.20) by 39% and SARIMA (RMSE=81.39) by 89%, consistent with gradient boosting's efficacy on engineered lag features for non-linear spatiotemporal series. Meanwhile, SARIMA exhibits substantially higher errors due to its linear assumptions failing to capture non-linear demand volatility.

	RMSE	MAE	MAPE	R2
SARIMA	102.6718	51.1196	486.9517	-1.1844
XGBoost	12.9721	4.3073	53.7989	0.9527
LSTM	59.4643	26.2994	227.2102	0.1973

Figure 19 *Model performance summary across all clusters*

Figure 20, Figure 21 and Figure 22 illustrate performance degradation over extended horizons, where XGBoost and LSTM demonstrate slower growth of RMSE and MAE from 1 to 7 days and even improvement of MAPE from 1-day to 7-day forecasting for XGBoost, confirming its robustness for multi-step forecasting as corroborated by comparisons in taxi demand literature [33].

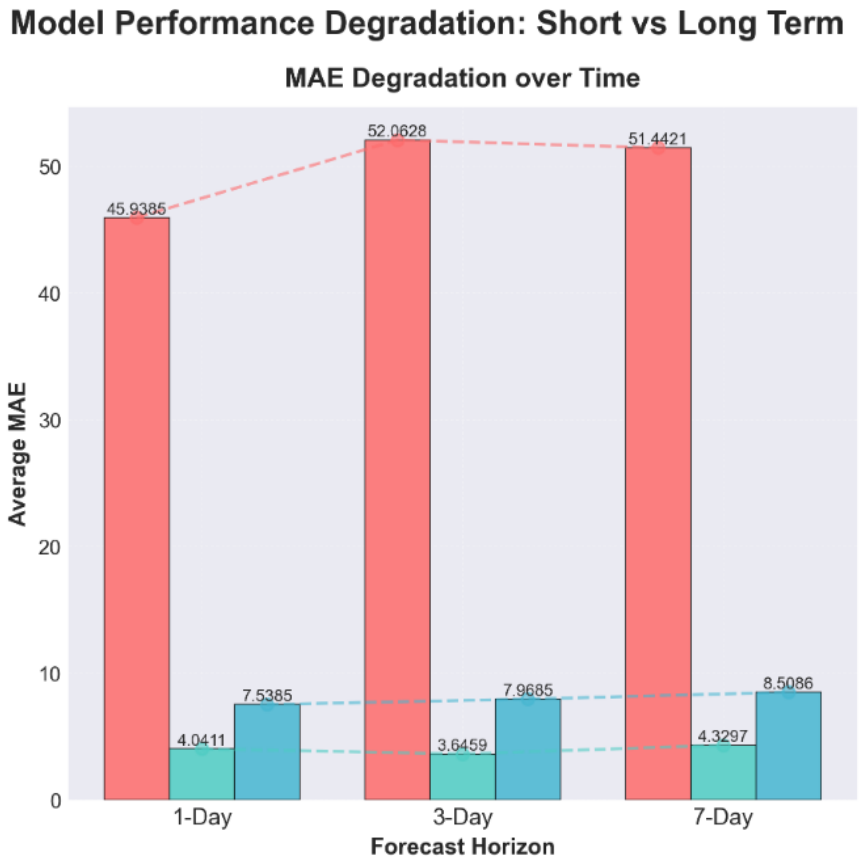

Figure 20 *Model performance degradation - MAE*

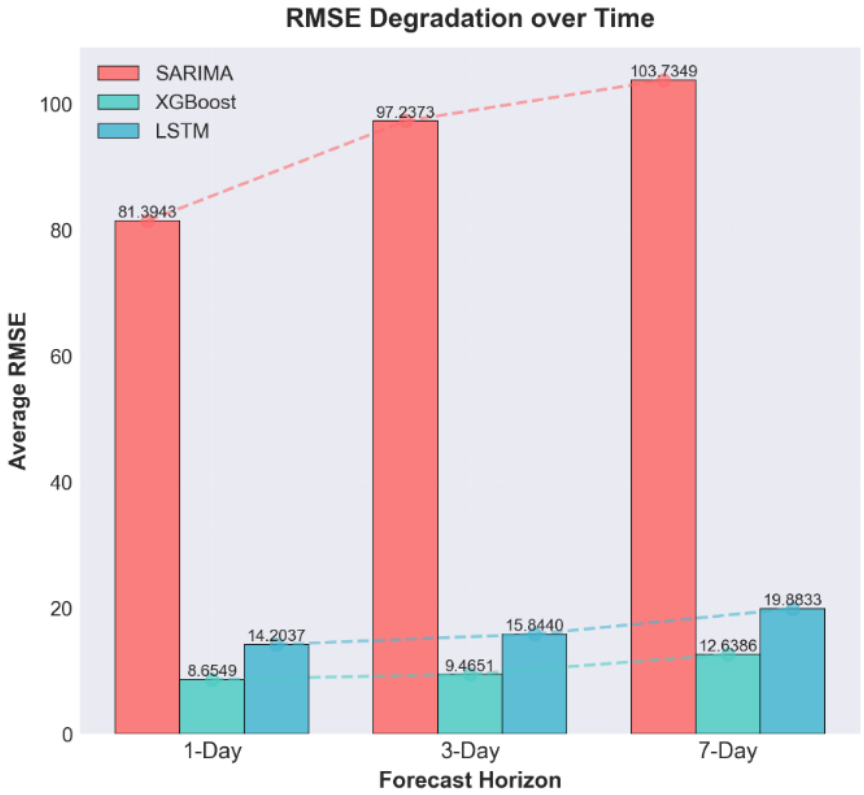

Figure 21 *Model performance degradation - RMSE*

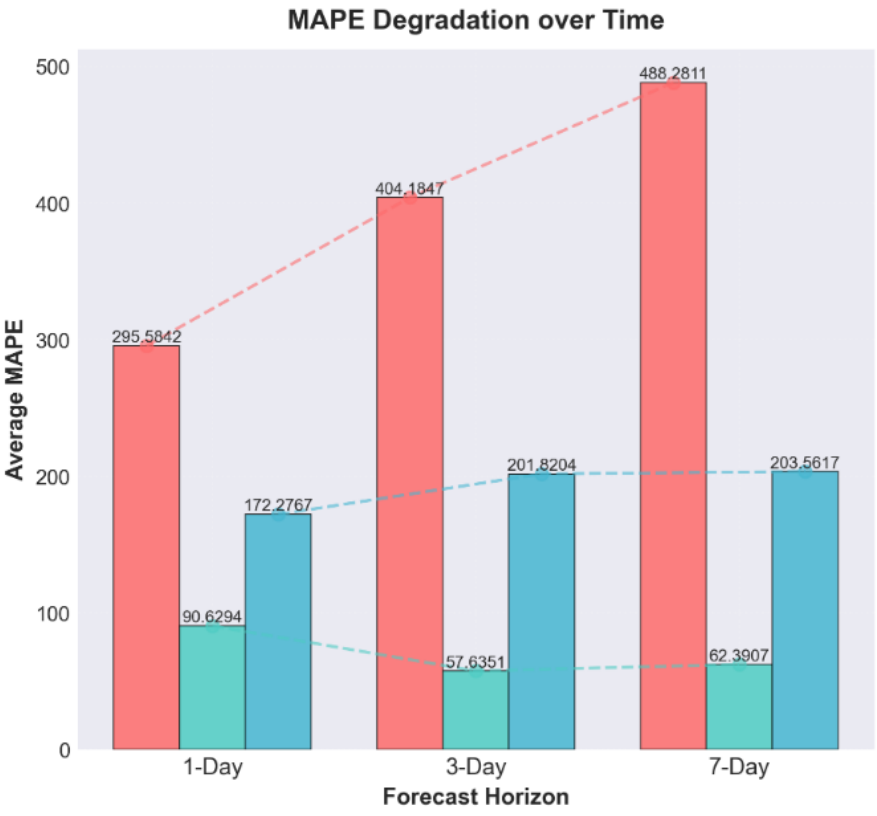

Figure 22 *Model performance degradation - MAPE*

Representative forecast visualizations in Figure 23 (cluster T3_S25) and Figure 24 (T2_S65) highlight practical alignment: XGBoost closely tracks actual peaks, LSTM captures temporal dynamics, but overpredicts baselines, and SARIMA shows phase misalignment characteristic of autoregressive limitations on non-stationary series. These results validate the hybrid clustering preprocessing, as cluster-specific modelling reduces noise and enhances non-linear methods' generalization, aligning with tensor-based NYC taxi analyses emphasizing spatiotemporal decomposition for improved forecasting stability.

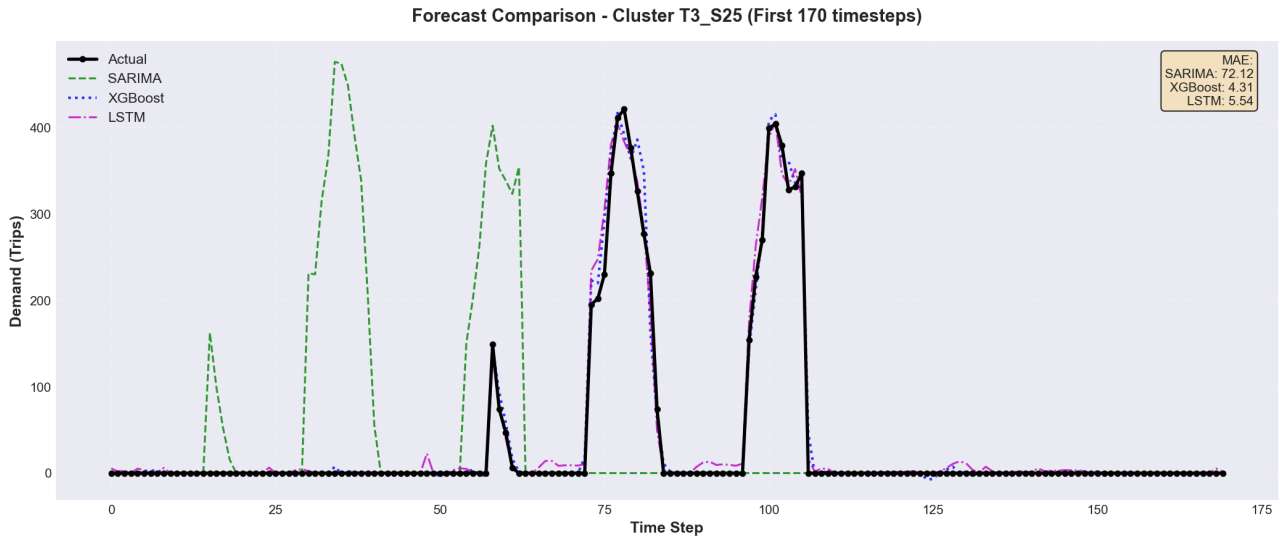

Figure 23 *Forecast comparison for cluster T3_S25*

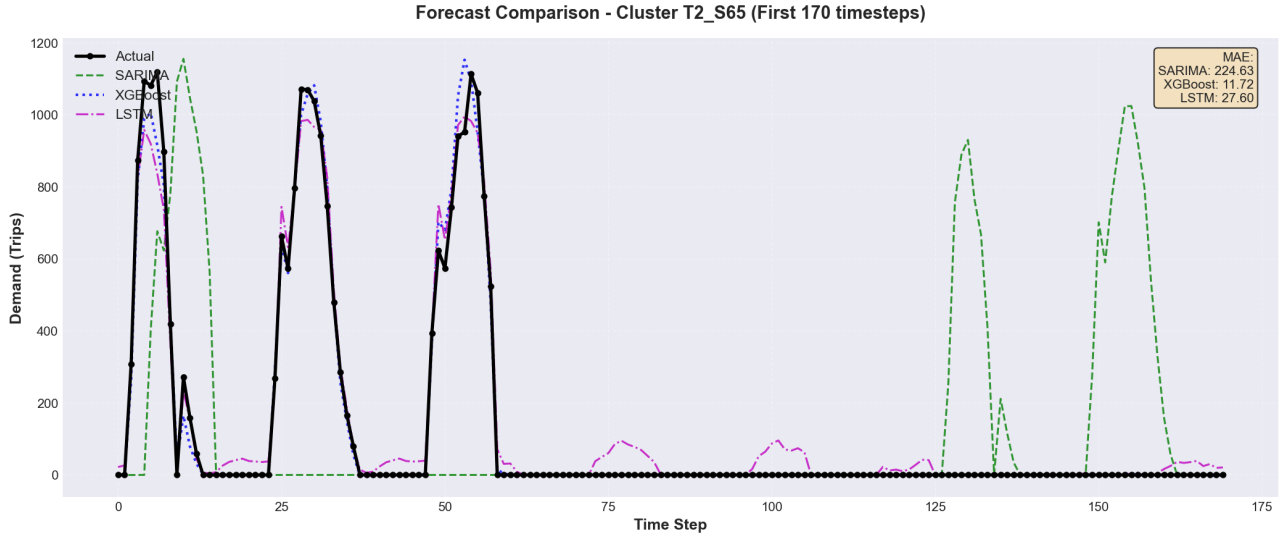

Figure 24 *Forecast comparison for cluster T2_S65*

4 Results and conclusions

Here, the overview of results is provided, with the linkage to the aim and goals raised for this paper.

4.1 Results

This section reports on the aim and three objectives of the paper. Below is the summary of the results achieved during the research:

- Hybrid clustering proved to be a working technique, decomposing taxi demand data into behaviourally coherent clusters.
- SARIMA model was not able to model and forecast demand properly, due to high amount of volatility in the data combined with sparsely or zero inflated data and its non-linearity - its multi-step forecasting displays a distinct phase misalignment and was not able to predict peaks effectively in time.
- Both XGBoost and LSTM models, when tuned and optimized, proved to be robust and showed better performance in forecasting the taxi demand among different clusters compared to baseline SARIMA.
- SARIMA model, despite rigorous hyperparameter optimization via grid search and analysing ACF/PACF plots to determine optimal stationarity transformations, exhibited significant performance degradation compared to the Machine Learning (XGBoost) and Deep Learning (LSTM) approaches.

The reported values in the table below represent performance across all 533 clusters, computed on the test set, and thus summarise the global behaviour of each model family.

Table 3 *Forecasting performance of SARIMA, XGBoost and LSTM models across different horizons, evaluated on the held-out test period (averaged across all demand clusters).*

Model	Horizon	RMSE	MAE	MAPE (%)	R^2
SARIMA	1 day (24 h)	81.39	45.94	295.58	-0.41
	3 days (72 h)	97.24	52.06	404.18	-24.37
	7 days (168 h)	103.73	51.44	488.28	-1.08
XGBoost	1 day (24 h)	8.65	4.04	90.63	0.62
	3 days (72 h)	9.47	3.65	57.64	0.91
	7 days (168 h)	12.64	4.33	62.39	0.96
LSTM	1 day (24 h)	14.20	7.54	172.28	0.57
	3 days (72 h)	15.84	7.97	201.82	0.75
	7 days (168 h)	19.88	8.51	203.56	0.89

The numerical results in Table 3 confirm that both XGBoost and LSTM substantially outperform the SARIMA baseline across all forecast horizons, particularly beyond the 24-hour horizon where

SARIMA's phase misalignment and error accumulation become dominant. For the 1-day ahead forecasts, XGBoost achieves an RMSE of approximately 8.65 and MAE of 4.04 while LSTM attains 14.20 and 7.54 respectively, compared to SARIMA's considerably higher errors of 81.39 for RMSE and 45.94 for MAE. This pattern is mirrored in MAPE and R^2, where SARIMA yields a relatively low explained variance of around -0.41 over 24 hours, whereas LSTM and XGBoost reach R^2 values in the range of 0.57-0.62, indicating that non-linear methods capture a substantially larger share of the underlying demand variability.

As the horizon extends to 3 and 7 days, all models exhibit some degradation in accuracy, but the magnitude of this degradation differs markedly. SARIMA's RMSE increases from 81.39 at 1 day to 97.24 at 3 days and 103.73 at 7 days, with a simultaneous drop in R^2 to near -1.08, confirming that the linear specification struggles to maintain temporal alignment under recursive multi-step forecasting. In contrast, XGBoost and LSTM degrade more gracefully: for example, XGBoost's RMSE grows only from 8.65 to 12.64 and LSTM from 14.2 to 19.88, while maintaining R^2 above 0.8 even at the 7-day horizon. This behaviour suggests that the non-linear models learn more robust representations of the weekly demand cycle and its local deviations, which is consistent with prior findings on taxi demand forecasting.

The horizon-wise degradation plots in Figure 20, Figure 21 and Figure 22 visually reinforce these quantitative differences. The MAE and RMSE curves demonstrate that SARIMA's error grows almost linearly with horizon length, while XGBoost and LSTM exhibit a slower, sub-linear increase. XGBoost consistently has lower MAPE than LSTM across all horizons, despite the weighted loss. LSTM's MAPE is higher, suggesting the weighting did not yield the relative accuracy advantage on percentage error.

4.2 Discussion and possible further directions

In this part, it is discussed what further changes and improvements can be implemented or tried in experimental setup to improve forecasting performance of the models.

For SARIMA, it is possible to adjust the models with data-driven order selection per cluster, as the same parameters for clusters with different dynamics might be too rigid. Also, current approach uses multi-step forecast without refitting - another possibility would be to shorten forecasting horizon or make it in blocks, meaning first forecast first from 24 to 48 hours ahead, and use newly predicted values as additional input for retraining the model further. This should in theory reduce the recursive error accumulation and increase each model's ability to forecast over longer horizons.

For LSTM model, it is possible to switch from Many-to-One prediction mode to Sequence-to-Sequence, but that would require building a separate forecasting pipeline branch for that approach, as the processing differs. Also, for LSTM it is possible to apply another variant CNN-LSTM which is a hybrid neural network combining CNN for spatial features with LSTM for temporal ones into one efficient one, as discussed in researches like Sainath et al. [36].

All in all, the empirical results of the current study demonstrate that the hierarchical two-stage clustering, combining temporal K-means with density-based spatial refinement via HDBSCAN, provides a useful decomposition of the NYC taxi demand data into behaviourally coherent clusters. By first segmenting the time axis into six prototypical temporal regimes (weekday morning, weekend

night, etc.) and then discovering spatial hotspots within each regime, the approach avoids the overly rigid assumption of static spatial clusters that is implicit in purely spatial methods. This design is particularly important for high-density central areas such as the southern part of Manhattan, where the same physical locations participate in distinct demand patterns depending on the time of week - the observed difference between coarse weekend clusters and finer weekday rush-hour clusters exemplifies this phenomenon. From a modelling perspective, this hierarchical clustering serves as a form of structured dimensionality reduction: instead of forecasting a single city-wide aggregate series or thousands of raw GPS cells, the models operate on 533 clusters that each correspond to spatio-temporally homogeneous demand regimes. This reduces noise, increases stationarity within each series, and allows both XGBoost and LSTM to focus on learning regular patterns. The fact that shared hyperparameters transfer well from the largest clusters to smaller ones, suggests that the demand-generation mechanisms are sufficiently similar across zones once the temporal regime is correctly identified, supporting the assumption of shared underlying dynamics.

At the same time, several limitations of the experimental setup should be acknowledged. First, the dataset covers only winter months, which constrains the analysis to intra-weekly and short-term seasonal effects and prevents explicit modelling of annual seasonality or longer-term structural changes in mobility patterns. Thus, the conclusions about model performance are valid for short-horizon demand in winter conditions, but may not fully correspond to periods with different weather regimes, tourism intensity, or other events.

Second, despite the use of sophisticated non-linear models, the feature engineering remains essentially endogenous: no exogenous covariates such as weather, holidays, or other are incorporated, even though such variables can improve predictive performance as can be seen in the results of other studies [20].

These limitations indicate that the current architecture captures a large portion of the spatiotemporal structure, but does not exploit all available sources of information, leaving room for future extensions.

The findings of this study are consistent with recent literature comparing classical time-series models and machine learning approaches for urban mobility forecasting. Similar to studies that reported the underperformance of ARIMA-type models with non-linear patterns, the SARIMA baseline here fails to track shifting peak times and exhibits rapid error growth over multi-day horizons. However, the present work differs from other existing approaches in two important aspects. First, the hierarchical clustering step explicitly disentangles temporal patterns before spatial clustering, whereas most prior clustering-based methods rely either on static spatial clusters or on generic spatiotemporal distance metrics that can struggle in very dense urban cores, such as NYC data. Second, instead of designing a single deep architecture that simultaneously learns clustering and forecasting, this study deliberately separates into an unsupervised clustering phase and a supervised forecasting phase. This modular design is closer to operational practice and enhances interpretability, as clusters can be analysed and visualised independently of the forecasting models.

From an applied perspective, the results suggest that the proposed architecture can support several decision-making tasks in urban mobility. In particular, the cluster-level forecasts generated

by XGBoost can be directly used for proactive taxi re-allocation for fleet operators, especially during weekend evenings and weekday rush hours. The ability to distinguish between temporally specific hotspots, such as weekday morning commuting versus late-night leisure districts, enables targeted interventions that are more precise than the city-wide ones. Moreover, the hybrid clustering provides a compact yet interpretable spatial index over the city that could be reused as an intermediate representation in other mobility analytics tasks. Because the clustering is learned from historical trip data alone, the same methodology can be transferred to other cities or datasets with minimal adjustments, provided that sufficient GPS-level trip records are available. For municipal planners, such transferable techniques and forecasting models offer scalable options towards more data-driven, demand-responsive transport systems.

Overall, the study contributes to filling several gaps identified in the literature review. Existing clustering-based analyses of NYC taxis often either ignore temporal heterogeneity or treat time only as a covariate rather than as a primary dimension for structuring the clustering process, whereas the present work demonstrates that explicitly separating temporal and spatial clustering yields more nuanced hotspot definitions that better reflect operational realities. Likewise, while many forecasting studies compare a small number of models on aggregated series, this thesis evaluates three distinct modelling paradigms across hundreds of clusters and multiple horizons, providing a more fine-grained picture of how model performance depends on spatial context and forecast length. Finally, by documenting the full pipeline - from aggressive data cleaning and feature engineering through hierarchical clustering, PCA-based dimensionality reduction, and model-specific optimisation - the thesis offers a reproducible blueprint for large-scale spatiotemporal forecasting using publicly available datasets. This end-to-end perspective is often missing in more narrowly focused methodological papers, yet it is crucial for practitioners who need to deploy such models under real computational and data constraints.

4.3 Conclusions

The visual analysis suggests that while SARIMA successfully identified the existence of a seasonal pattern, it failed to accurately localize it temporally, reflecting a fundamental limitation of how ARIMA-class models process temporal information through differencing and filtering. For NYC hourly taxi demand data, it appears that the model was able to learn the shape of the demand cycles, however, losing the precise temporal registration of actual events and fails to capture these dynamic shifts. The temporal increases (i.e. peaks) were caught because the weekly seasonal component was detected correctly, but the non-seasonal differencing causes the forecast to lag. SARIMA is a linear model and cannot capture those non-linear events in highly non-linear NYC Taxi data without external regressors. This explains why, in comparison, LSTM and XGBoost track peaks more accurately: they implicitly learn non-linear relationships through hidden layers and tree ensembles and map non-linear relationships effectively, as was discussed in Gifty and Li's work [15]. This indicates that the strict periodic assumption ($m = 24$ or $m = 168$) was insufficient for capturing the dynamic variance in peak demand times for SARIMA, as in reality it might drift slightly (e.g. rush hours appear not at 5 p.m. but at 6 p.m. on Friday) and SARIMA cannot adapt to this change dynamically because

of the fixed coefficients, while the non-parametric nature of XGBoost and the adaptive weights of the LSTM handled this flexibility with greater precision.

Another point is error accumulation due to recursive multi-step forecasting - with more than 24 steps, step $t+2$ relies on the prediction of step $t+1$, not the actual value. If step $t+1$ is slightly off, that error propagates and amplifies. By step $t+20$, the model might drift completely out of phase, as can be seen in the results of this research.

What can also be seen from conducted ACF/PACF analysis is that the autocorrelation with immediate past (lags 1-3) is stronger, while the correlation with seasonal lag (24 or 168 hours) can be noisy. SARIMA struggles to balance these long-term dependencies compared to LSTM mechanisms.

Overall, the results provide empirical evidence that while SARIMA remains a robust baseline for low-volatility, strictly periodic data, it is not well-suited for the high-variance, non-stationary nature of granular NYC taxi demand compared to modern non-linear architectures.

In summary, the main conclusions of this thesis can be articulated as follows:

- Hierarchical temporal–spatial clustering using K-means and HDBSCAN yields behaviourally meaningful demand clusters that reflect distinct weekly regimes and spatial hotspots, and provides a compact representation suitable for downstream forecasting.
- Classical SARIMA models, even when carefully tuned using ACF/PACF analysis and grid search, struggle with zero-inflated, highly volatile cluster-level series and exhibit pronounced phase misalignment and error accumulation over multi-day horizons.
- Non-linear models, specifically XGBoost and LSTM, deliver substantially better accuracy across all horizons, with LSTM performing well on peak periods when combined with a peak-weighted loss function and PCA-based dimensionality reduction.
- The proposed pipeline demonstrates that large-scale, fine-grained taxi demand forecasting is feasible using only historical trip records and can be implemented efficiently on commodity hardware, making it attractive for both academic and industrial applications.
- Future work should extend the framework by incorporating exogenous variables (weather, events, holidays), additional seasons of data, and alternative deep architectures such as CNN-LSTM hybrids or attention-based models, to further improve peak prediction and robustness.

References and sources

[1] S. D. et al. “A Hierarchical Density-Based Clustering Method Applied to Mixed-Mail in Austria.” In: (2024). URL: https://www.bvl.de/lore/all-volumes--issues/volume-17/issue-1/a-hierarchical-density-based-clustering-method-applied-to-mixed-mail-in-austria.

[2] P. S. Benarji, P. S. Bharadwaj, B. Neeha, D. Srikanth, V. Ankitha. “Taxi Demand Prediction using Machine Learning.” In: *International Research Journal of Engineering and Technology (IRJET)* 10 (2023). URL: https://www.irjet.net/archives/V10/i5/IRJET-V10I5234.pdf.

[3] Y. Bengio, P. Simard, P. Frasconi. “Learning long-term dependencies with gradient descent is difficult.” In: *IEEE Transactions on Neural Networks* 5.2 (1994), pages 157–166. https://doi.org/10.1109/72.279181. URL: https://www.comp.hkbu.edu.hk/~markus/teaching/comp7650/tnn-94-gradient.pdf.

[4] D. Birant, A. Kut. “ST-DBSCAN: An algorithm for clustering spatial-temporal data.” In: *Data Knowl. Eng.* 60 (2007), pages 208–221. URL: https://api.semanticscholar.org/CorpusID:22091296.

[5] T. Chen, C. Guestrin. “XGBoost: A Scalable Tree Boosting System.” In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. ACM, 2016, pages 785–794. https://doi.org/10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[6] Z. e. a. Chen. “Multitask Learning and GCN-Based Taxi Demand Prediction for a Traffic Road Network.” In: 2020. https://doi.org/doi:10.3390/s20133776. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7374365/.

[7] Z. Chen. “Machine Learning for Smart Cities: LSTM Model-Based Taxi OD Demand Forecasting in New York.” In: 2021. URL: https://www.scitepress.org/Papers/2024/132054/132054.pdf.

[8] G. Cheng, V. Peddinti, V. Manohar, S. Khudanpur, Y. Yan. “An Exploration of Dropout with LSTMs.” In: 2017, pages 1586–1590. https://doi.org/10.21437/Interspeech.2017-129.

[9] D. Correa, C. Moyano. “Analysis and Prediction of New York City Taxi and Uber Demands.” In: *Journal of Applied Research and Technology* 21 (2023), page 1. https://doi.org/10.22201/icat.24486736e.2023.21.5.2074.

[10] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. “A density-based algorithm for discovering clusters in large spatial databases with noise.” In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pages 226–231. URL: https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf.

[11] S. S. Faghih. *Understanding and Modeling Taxi Demand Using Time Series Models*. 2019. URL: https://academicworks.cuny.edu/cc_etds_theses/881.

[12] S. S. Faghih, A. Safikhani, B. Moghimi, C. Kamga. *Predicting Short-Term Uber Demand Using Spatio-Temporal Modeling: A New York City Case Study*. 2018. URL: https://arxiv.org/abs/1712.02001.

[13] J. Frost. "Mean Squared Error (MSE)." In: (). URL: https://statisticsbyjim.com/regression/mean-squared-error-mse/.

[14] Y. Gal, Z. Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2016. URL: https://arxiv.org/abs/1506.02142.

[15] A. Gifty, D. Y. Li. "A Comparative Analysis of LSTM, ARIMA, XGBoost Algorithms in Predicting Stock Price Direction." In: *Engineering And Technology Journal* 9.8 (2024), pages 4978–4986. https://doi.org/10.47191/etj/v9i08.50. URL: https://everant.org/index.php/etj/article/view/1495.

[16] R. Hyndman, G. Athanasopoulos. "Forecasting: principles and practice (3rd ed.) OTexts, Chapter 9.7 on Seasonal ARIMA." In: URL: https://otexts.com/fpp3/seasonal-arima.html.

[17] R. J. Hyndman. "Seasonal ARIMA models." In: (). URL: https://robjhyndman.com/talks/RevolutionR/10-Seasonal-ARIMA.pdf.

[18] S. Hochreiter, J. Schmidhuber. "Long Short-Term Memory." In: *Neural Computation* 9.8 (1997), pages 1735–1780. ISSN: 0899-7667. https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://www.bioinf.jku.at/publications/older/2604.pdf.

[19] B. Yu, H. Yin, Z. Zhu. "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting." In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. IJCAI-2018. International Joint Conferences on Artificial Intelligence Organization, 2018, pages 3634–3640. https://doi.org/10.24963/ijcai.2018/505. URL: http://dx.doi.org/10.24963/ijcai.2018/505.

[20] H. Yu, X. Chen, Z. Li, G. Zhang, P. Liu, J. Yang, Y. Yang. "Taxi-Based Mobility Demand Formulation and Prediction Using Conditional Generative Adversarial Network-Driven Learning Approaches." In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pages 3888–3897. https://doi.org/10.1109/TITS.2019.2923964. URL: https://yangzzzy.github.io/PDF/tits19_taxi.pdf.

[21] S. Khazem, H. Kanso. "Cyclical Temporal Encoding and Hybrid Deep Ensembles for Multistep Energy Forecasting." In: (2025). URL: https://arxiv.org/abs/2512.03656.

[22] F. e. a. Li. "A hierarchical temporal attention-based LSTM encoder-decoder model for individual mobility prediction." In: volume 403. 2020, pages 153–166. https://doi.org/doi:10.1016/j.neucom.2020.03.080. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7252178/.

[23] Z. Li, J. Liu, X. Yang. "New York city taxi demand forecast based on ARIMA model. Applied and Computational Engineering." In: 68 (2025), pages 143–149. URL: https://doi.org/10.54254/2755-2721/68/20241416.

[24] L. Linthoi, M. K. G, M. Rallapalli. “TAXI DEMAND PREDICTION IN REAL TIME.” In: *International Research Journal of Engineering and Technology (IRJET)* 10 (2023). URL: https://www.irjet.net/archives/V10/i9/IRJET-V10I915.pdf.

[25] C. Ma. “A short-time traffic flow prediction model based on TCN-LSTM with causal convolutional layer.” In: 2021. URL: https://global-sci.org/index.php/jics/article/download/15450/30654/31884.

[26] C. Malzer, M. Baum. “A Hybrid Approach To Hierarchical Density-based Cluster Selection.” In: (2020), pages 223–228. https://doi.org/10.1109/mfi49285.2020.9235263. URL: https://arxiv.org/pdf/1911.02282.

[27] S. Mishra, T. S. Murthy. *A Predictive and Optimization Approach for Enhanced Urban Mobility Using Spatiotemporal Data*. 2024. URL: https://arxiv.org/abs/2410.05358.

[28] V. Mittal. *NYC Yellow Taxi Trip Data*. URL: https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data.

[29] A. de Myttenaere, B. Golden, B. Le Grand, F. Rossi. “Mean Absolute Percentage Error for regression models.” In: *Neurocomputing* 192 (2016), pages 38–48. ISSN: 0925-2312. https://doi.org/10.1016/j.neucom.2015.12.114. URL: https://apiacoa.org/publications/2016/demyttenaeregoldenetal2016mean-absolute.pdf.

[30] L. Pengshun, C. Jiarui, Z. Yi, Z. Yi. “Multi-zone prediction analysis of city-scale travel order demand.” In: *PLOS ONE* 16.3 (2021), pages 1–23. https://doi.org/10.1371/journal.pone.0248064. URL: https://doi.org/10.1371/journal.pone.0248064.

[31] R. G. Pontius, O. Thontteh, H. Chen. “Components of information for multiple resolution comparison between maps that share a real variable.” In: *Environmental and Ecological Statistics* 15 (2008), pages 111–142. ISSN: 1573-3009. https://doi.org/10.1007/s10651-007-0043-y. URL: https://commons.clarku.edu/cgi/viewcontent.cgi?params=/context/faculty_geography/article/1769/&path_info=GeogFacWorks_Pontius_Components_2006.pdf.

[32] K. Qin, Q. Zhou, T. Wu, Y. Q. Xu. “HOTSPOTS DETECTION FROM TRAJECTORY DATA BASED ON SPATIOTEMPORAL DATA FIELD CLUSTERING.” In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W7 (2017), pages 1319–1325. https://doi.org/10.5194/isprs-archives-XLII-2-W7-1319-2017. URL: https://isprs-archives.copernicus.org/articles/XLII-2-W7/1319/2017/.

[33] S. R., Suprapto, S. A. “Optimizing urban mobility: A comparative analysis of taxi demand prediction models.” In: *Ingénierie des Systèmes d’Information* 29 (2024). https://doi.org/https://doi.org/10.18280/isi.290522.

[34] “RNN vs LSTM vs GRU vs Transformers.” In: URL: https://www.geeksforgeeks.org/deep-learning/rnn-vs-lstm-vs-gru-vs-transformers/.

[35] A. Safikhani, D. Fahrangnassab, M. Shoja. “Spatio-temporal modeling of yellow taxi demands in New York City using STARMA and LASSO penalization.” In: *arXiv preprint* arXiv:1711.10090 (2017). URL: https://arxiv.org/pdf/1711.10090.pdf.

[36] T. N. Sainath, O. Vinyals, A. Senior, H. Sak. “Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks.” In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pages 4580–4584. https://doi.org/10.1109/ICASSP.2015.7178838. URL: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/43455.pdf.

[37] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. Woo. *Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting*. 2015. URL: https://arxiv.org/abs/1506.04214.

[38] R. Shwartz-Ziv, N. Tishby. *Opening the Black Box of Deep Neural Networks via Information*. 2017. URL: https://arxiv.org/abs/1703.00810.

[39] A. Silfver. “Short-Term Forecasting of Taxi Demand using a two Channelled Convolutional LSTM network.” In: *Linköping University | Department of Computer and Information Science* (2019). URL: https://liu.diva-portal.org/smash/get/diva2:1431037/FULLTEXT01.pdf.

[40] T. P. S. University. “Applied Data Mining and Statistical Learning - Cluster Analysis - K-Means.” In: (). URL: https://online.stat.psu.edu/stat857/node/110/.

[41] U. VANICHRUJEE. “TAXI DEMAND PREDICTION USING ENSEMBLE MODEL AND TAXI GPS DATA ANALYSIS.” In: 2017. URL: https://ethesisarchive.library.tu.ac.th/thesis/2017/TU_2017_5922040588_8707_6838.pdf.

[42] P. X. Zhao, K. Qin, Q. Zhou, C. K. Liu, Y. X. Chen. “DETECTING HOTSPOTS FROM TAXI TRAJECTORY DATA USING SPATIAL CLUSTER ANALYSIS.” In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* II-4/W2 (2015), pages 131–135. https://doi.org/10.5194/isprsannals-II-4-W2-131-2015. URL: https://isprs-annals.copernicus.org/articles/II-4-W2/131/2015/.

Appendix 1. Python Code

The code, written in Python using Positron IDE, which was used for data processing and clustering algorithms and the forecasting pipeline using the models, can be found in the GitHub repository, together with the instructions to reproduce the results, under the following link:
GitHub repository

The execution of the mentioned code was made on the following hardware setup: AMD Ryzen 7 9800X3D processor and 32GB DDR5 RAM.

Appendix 2. Using AI tools

An AI tool, in particular Perplexity AI, was used as an aid when writing this thesis, for the purposes as listed: finding the relevant literature sources, giving ideas and advices on the topic and techniques implementations and caveats associated with them, improving the writing style, enhancing the code to produce required visualizations, optimization of code computational performance to ensure code re-usage, and adding logging and performance monitoring layers to maintain understandable running code.