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 : Assoc. Prof. Dr. Jurgita Markevičiūtė

Reviewer :

Vilnius
2025

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.

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

Summary

(PAPILDYTI)

Keywords: statiotemporal data, clustering algorithm, demand forecasting, urban mobility, K-means, HDBSCAN, SARIMA, VAR, XGBoost, LSTM

Santrauka

(PAPILDYTI)

Raktiniai žodžiai: Erdvėlaikio duomenys, klasterizavimo algoritmas, paklausos prognozavimas, K-means, HDBSCAN, SARIMA, VAR, XGBoost, LSTM

List of Figures

Figure 1 Average speed boxplots 19
Figure 2 Trip duration boxplots 20
Figure 3 Trip distance boxplots 20
Figure 4 Distribution of some features before cleaning and removing outliers 21
Figure 5 Frequency distribution after cleaning 23
Figure 6 Pickup time frequency heatmap 24
Figure 7 Pickup time frequency distribution by weekdays 24
Figure 8 Temporal clustering optimization 26
Figure 9 Temporal clusters analysis 27
Figure 10 Heatmap os spacial clustering parameters 28
Figure 11 Spacial clustering analysis 28

List of Tables

Table 1 Comparative Summary of Evaluation Metrics 17
Table 2 Dataset Field Descriptions 18

Contents

Summary . . . 3

Santrauka . . . 4

List of Figures . . . 5

List of Tables . . . 6

List of abbreviations . . . 8

Introduction . . . 9

1 Related work . . . 11
1.1 Classical statistical approaches - Vector Autoregression and Seasonal ARIMA . . . 11
1.2 Machine learning models - XGBoost . . . 11
1.3 Advanced deep learning techniques - CNN, LSTM, ConvLSTM . . . 11
1.4 Clustering algorithms . . . 11

2 Methodology . . . 13
2.1 Clustering algorithms . . . 13
2.2 Forecasting models . . . 15
2.2.1 Evaluation criteria . . . 15

3 Experimental setup . . . 18
3.1 Dataset . . . 18
3.2 Feature engineering with cleaning and EDA . . . 22
3.3 Clustering algorithms . . . 24
3.4 Data Splitting . . . 25
3.5 Model comparison . . . 25
3.6 Optimization . . . 25
3.6.1 Optimization for temporal clustering . . . 26
3.6.2 Optimization for spacial clustering . . . 27
3.6.3 Optimization for SARIMA . . . 29
3.6.4 Optimization for XGBoost . . . 29
3.6.5 Optimization for LSTM . . . 29
3.7 Evaluating . . . 29

4 Results and conclusions . . . 30
4.1 Results . . . 30
4.2 Discussion . . . 30
4.3 Conclusions . . . 30

Appendix 1. Python Code . . . 32

Appendix 2. Using AI tools . . . 33

List of abbreviations

VAR	Vector autoregression statistical model
SARIMA	Seasonal ARIMA
XGBoost	Long short-term memory recurrent neural network
CNN	Convolutional 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

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 as regards how to leverage 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 modeling 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, insufficient integration of multi-modal influences, and reliance on superficial feature construction.

The aim of this thesis is to examine whether the application of combined clustering algorithms with different data science approaches - such as classical statistical Vector Autoregression (VAR) and 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 hypotheses 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:

- To examine whether the application of combined clustering algorithms with different forecasting models can produce actionable insights in urban mobility;
- Applying 3 different 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 modeling 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 literature and contribute operationally relevant knowledge for the optimization of city mobility systems. This research fills a methodological gap in combining advanced spatiotemporal analytics with real-world transportation data, proposing scalable solutions with concrete relevance for operational and policy planning in large urban contexts.

The innovation of the research can be justified by applying combined hierarchical clustering algorithms for temporal and spacial features, as well as the investigation of prediction on different time horizons, which is not usually done in such researches.

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. In this section methodological landscape on spatiotemporal modeling is reviewed and the research gaps each of the approaches leaves are identified, providing the theoretical foundation for advanced and more robust modeling approaches, that are required for this type of complex data.

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.

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, regression-based techniques. Those methods have provided foundational and grounding insights into urban mobility patterns. This approach usually treats spacial 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, as well as 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. (2024)[3].

1.2 Machine learning models - XGBoost

(DAR PAPILDYTI)

1.3 Advanced deep learning techniques - CNN, LSTM, ConvLSTM

(DAR PAPILDYTI)

1.4 Clustering algorithms

The overview of the researches and different approaches to spatiotemporal clustering algorithms used in academic papers.
(DAR PAPILDYTI APIE KIEKVIENO MINUSUS)
K-means - most simple aproach
DBSCAN - Density-based spatial clustering of applications with noise
ST-DBSCAN - explicitly handles spatiotemporal data, considering both spatial proximity and time when forming clusters

HDBSCAN - Hierarchical DBSCAN

2 Methodology

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

The initial idea was to simply implement K-means spacial clustering to the dataset and then apply VAR, XGBoost and CNN models, however the final architecture turned out to be a more complex, with hierarchical clustering firtly on temporal features and then applying the hybrid approach on spacial 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.

The second part - forecasting approaches also has moved to SARIMA as a baseline model as it captures seasonality better, and LSTM network for deep learning as it is able to capture non-linear spatiotemporal patterns.

2.1 Clustering algorithms

The first naive idea for clustering consisted only of spacial 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 [6], with it's 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 assigned as:

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

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 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 out to find the best approach to the problem.

DBSCAN - (PAPILDYTI)

ST-DBSCAN - (PAPILDYTI)

HDBSCAN - extends DBSCAN to handle clusters of varying densities [4]. Applied to taxi pickup locations, it automatically determines the number of clusters without requiring predetermined parameters like the number of clusters. The algoriths works as described in the equations 5, 6, 7, 8 below.

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

$$d_{\text{core}}(\mathbf{p}_i) = \text{distance to the } m\text{-th nearest neighbor of } \mathbf{p}_i \tag{5}$$

The parameter m represents the minimum number of samples in a neighborhood 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:

$$d_{\text{mreach}}(\mathbf{p}_i, \mathbf{p}_j) = \max\left\{d_{\text{core}}(\mathbf{p}_i), d_{\text{core}}(\mathbf{p}_j), \|\mathbf{p}_i - \mathbf{p}_j\|\right\} \tag{6}$$

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:

$$\text{MST} = \{(i,j) : d_{\text{mreach}}(\mathbf{p}_i, \mathbf{p}_j) \text{ is among the smallest } N-1 \text{ edges}\} \tag{7}$$

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:

$$\text{Stability}(C) = \sum_{\mathbf{p}_i \in C} \left(\lambda_{\text{death}}(\mathbf{p}_i) - \lambda_{\text{birth}}(\mathbf{p}_i)\right) \tag{8}$$

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:

$$\text{Select } C \text{ if Stability}(C) > \sum_{C' \in \text{children}(C)} \text{Stability}(C') \tag{9}$$

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 neighborhood (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, and within each of those temporal cluster - applying Hierarchical DBSCAN clustering algorithm to find natural clusters' centroids of spacial data, and reusing those as an initial starting point in K-means later refinement of clusters, resulting in a hybrid approach that can be described as innovative and experimental.

2.2 Forecasting models

(PAPILDYTI)

2.2.1 Evaluation criteria

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

(PAPILDYTI DAR KIEKVIENO APRASYMA)
MAE is an interpretable metric in original units that is robust to outliers. The mathematical formulation expresses it as the average absolute deviation from actual values.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{10}$$

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

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{11}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{12}$$

$$\text{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{13}$$

$$\text{sMAPE} = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{(|y_i| + |\hat{y}_i|)/2} \tag{14}$$

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(1 + y_i) - \log(1 + \hat{y}_i)\right)^2} \tag{15}$$

$$\text{SS}_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{16}$$

and the total sum of squares is:

$$\text{SS}_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{17}$$

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

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{18}$$

$$\text{MASE} = \frac{\text{MAE}}{\text{MAE}_{\text{naive}}} \tag{19}$$

For seasonal data, the denominator is typically the mean absolute error of a seasonal naive forecast:

$$\text{MASE} = \frac{\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}{\frac{1}{n-s}\sum_{i=s+1}^{n}|y_i - y_{i-s}|} \tag{20}$$

where s denotes the seasonal period.

The comparative table of evaluation metrics is provided in Table 1

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

Table 1 *Comparative Summary of Evaluation Metrics*

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

3 Experimental setup

Here in this section the whole process is described from receiving the data to the final model outputs evaluation.

3.1 Dataset

This research employs the NYC Yellow Taxi Trip Data dataset compiled by Vishesh Mittal and made available through the Kaggle platform [5]. 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.

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 as described in Table 2. There are some limitations to the data, such as: the concentration of data in January months may not adequately represent annual mobility patterns, seasonal variations, or the full spectrum of urban events that influence taxi demand throughout a year.

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. Below in the Figure 1 is the boxplot distribution of average speed values, there can be seen how it contained unreasonable values before, and how the distribution changed after capping it to 120 km/h ceiling.

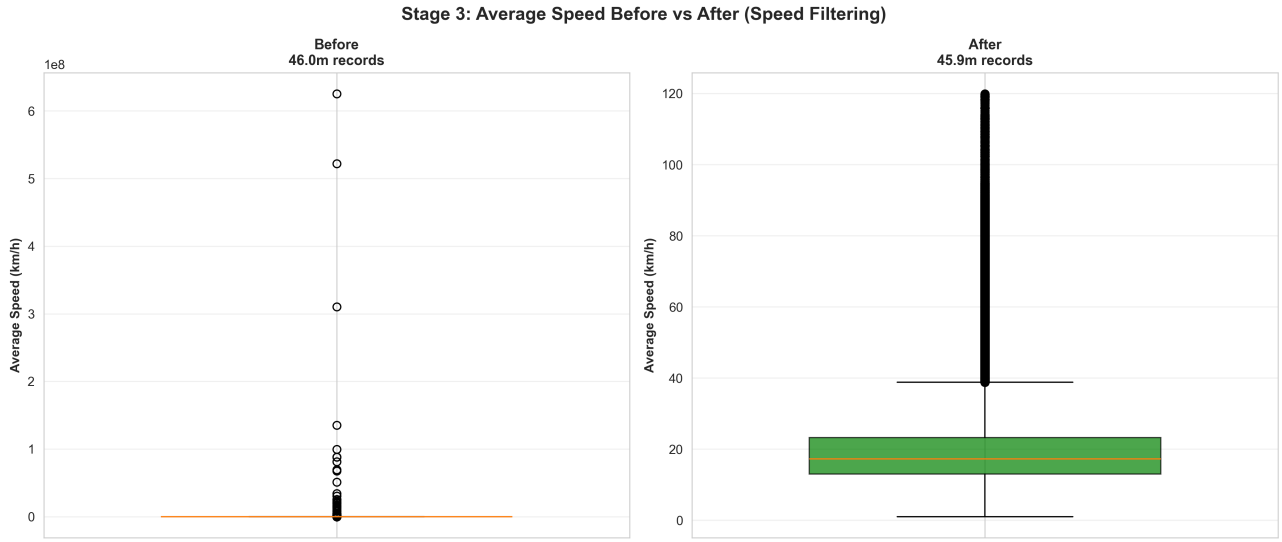

Figure 1 *Average speed boxplots*

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.

Some feature (including calculated) distribution before the cleaning can be seen in Figure 4.

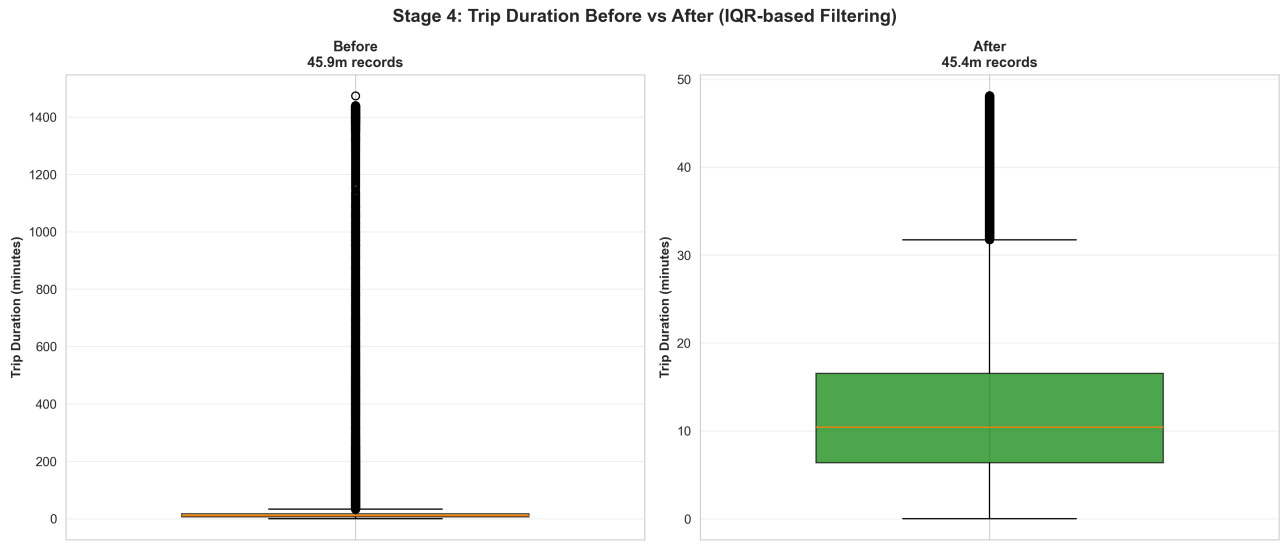

Figure 2 *Trip duration boxplots*

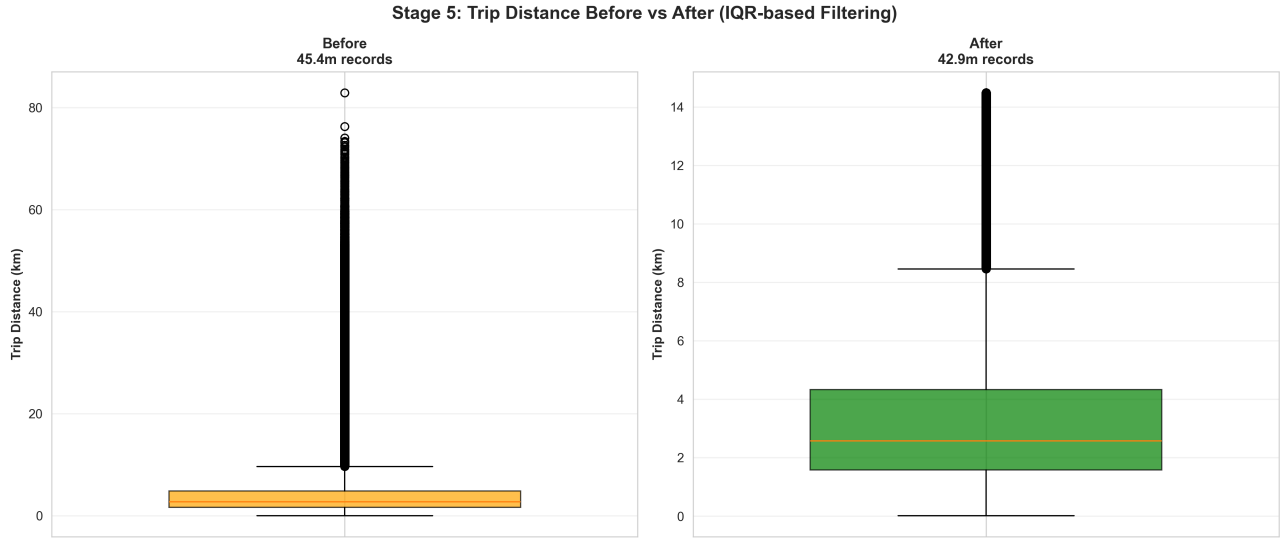

Figure 3 *Trip distance boxplots*

Additional data preparation included filtering out the trips which pickup coordinates were are not included in the rectange formed by New York City boundaries, i.e. pickup longitude being between -74.05 and -73.75, while 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.

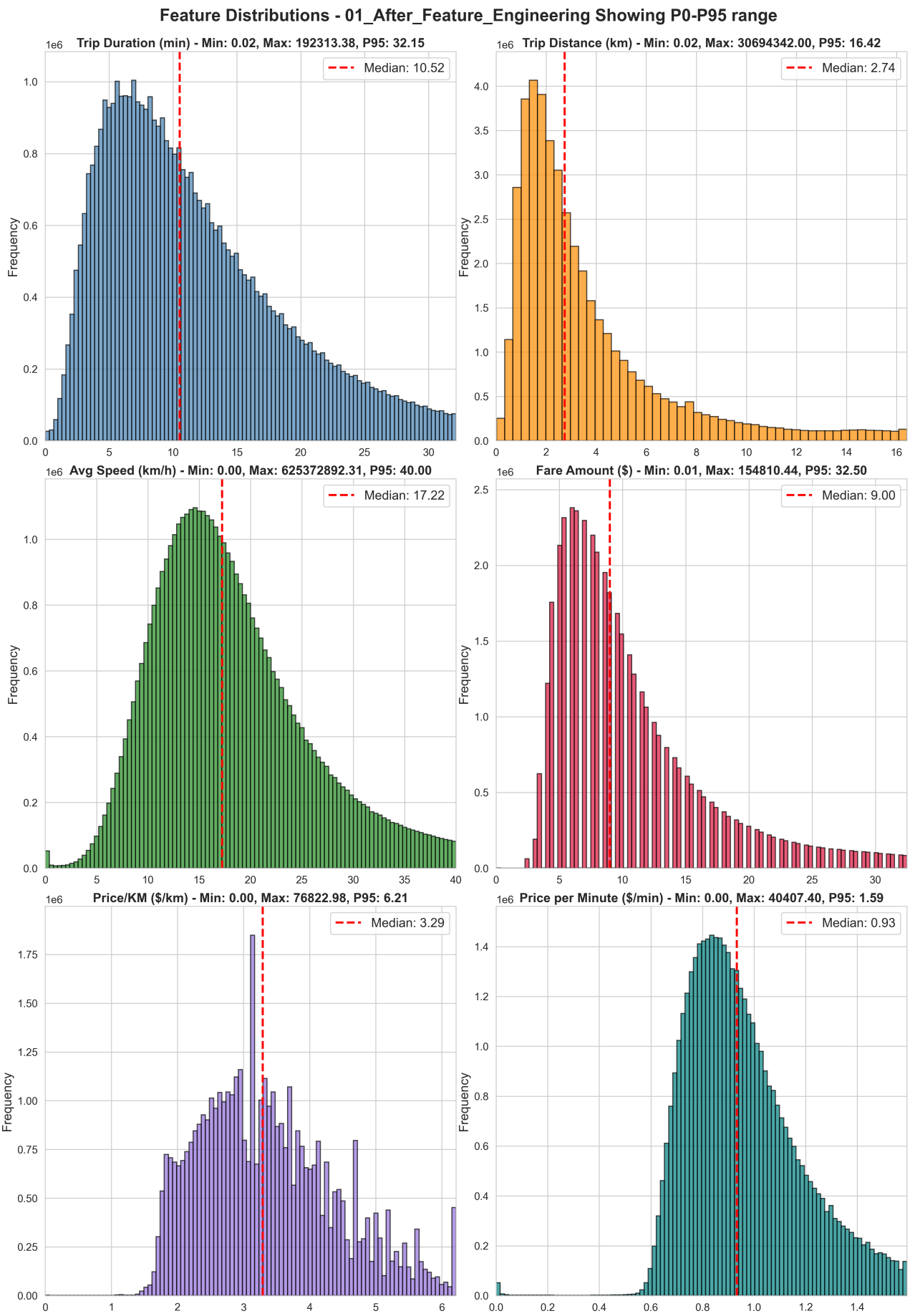

Figure 4 *Distribution of some features before cleaning and removing outliers*

3.2 Feature engineering with cleaning and EDA

Transforming Temporal Features

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 [2] . 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.

$$\text{Hour}_{\text{sin}} = \sin\left(\frac{2\pi \times \text{hour}}{24}\right) \tag{21}$$

$$\text{Hour}_{\text{cos}} = \cos\left(\frac{2\pi \times \text{hour}}{24}\right) \tag{22}$$

Variables:

- 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

$$\text{Day}_{\text{sin}} = \sin\left(\frac{2\pi \times \text{day}}{7}\right) \tag{23}$$

$$\text{Day}_{\text{cos}} = \cos\left(\frac{2\pi \times \text{day}}{7}\right) \tag{24}$$

Variables:

- 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

The features frequency distributions after implementing all cleaning and outliers removals can be seen in charts below in Figure 5

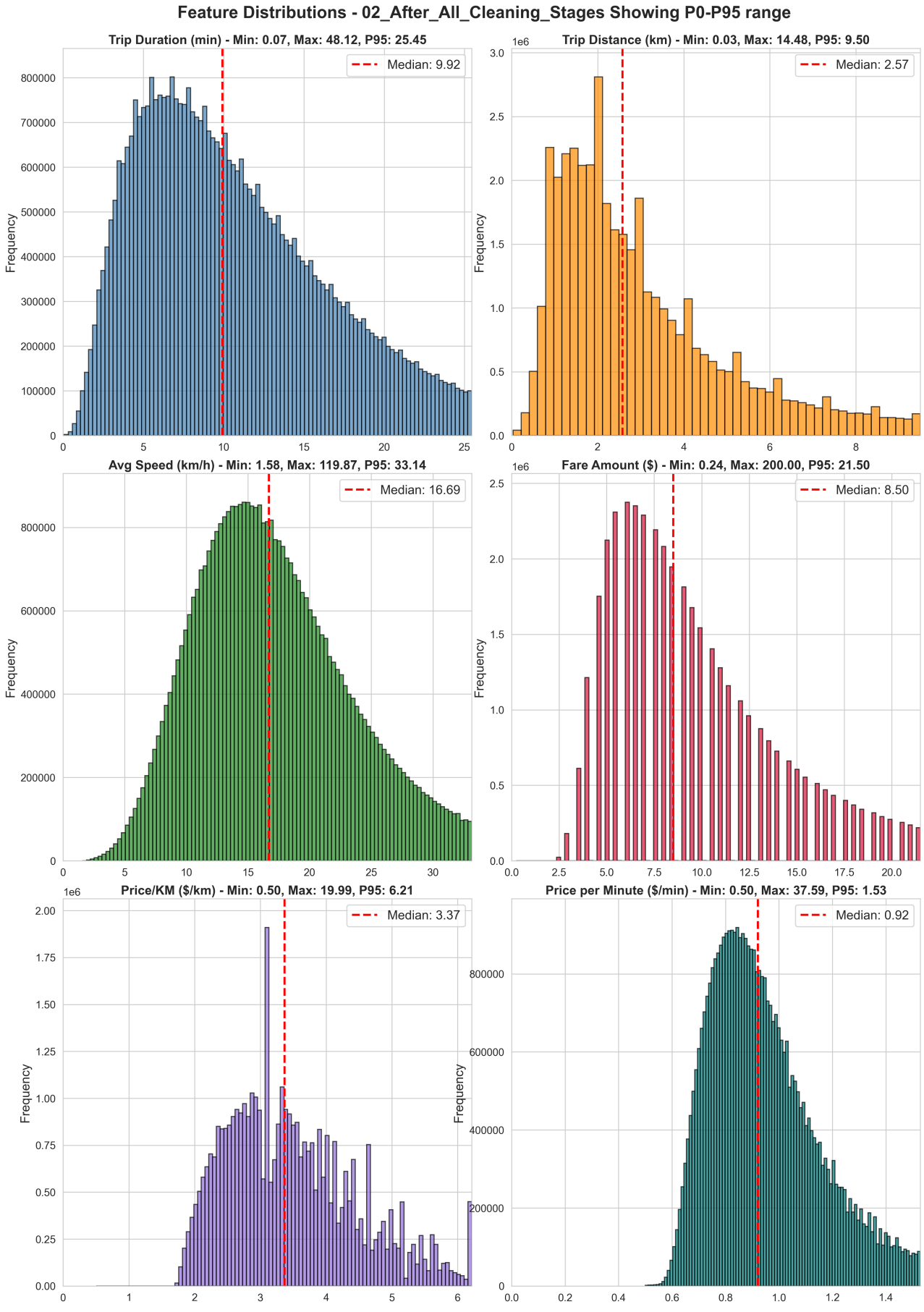

Figure 5 *Frequency distribution after cleaning*

3.3 Clustering algorithms

In this subsection it is described what was the initial approach on clustering the data and what resulted as a final technique to apply to define the both temporal and spacial distinctive patterns - K-means algorithm combination with HDBSCAN.

To better understand the data, exploratory data analysis was performed, and it showed that the temporal structure in dataset is quite different throughout the week, as can be seen in the heatmap graph in Figure 6, with a big concentration of demand on Thrursday, Friday and Saturday evenings and downtime during night and early mornings on a working days. Also frequency distribution plot in Figure 7 further proves different pickup hour demand patterns within weekdays and weekends.

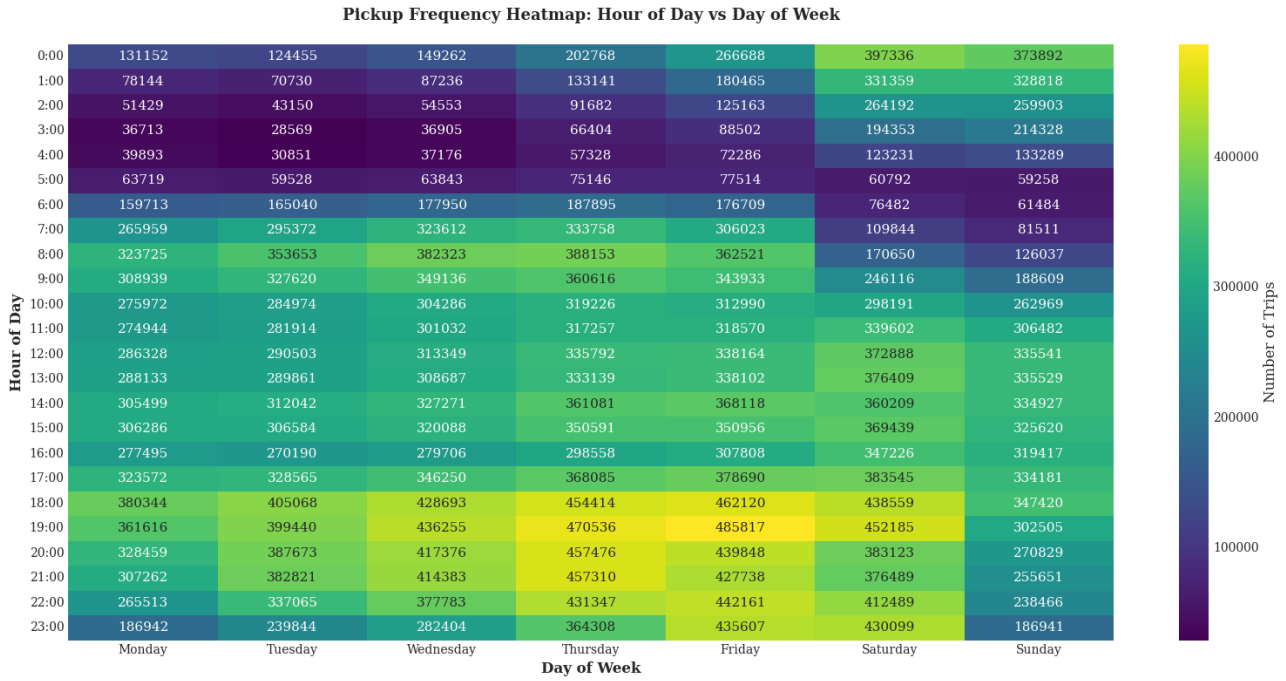

Figure 6 *Pickup time frequency heatmap*

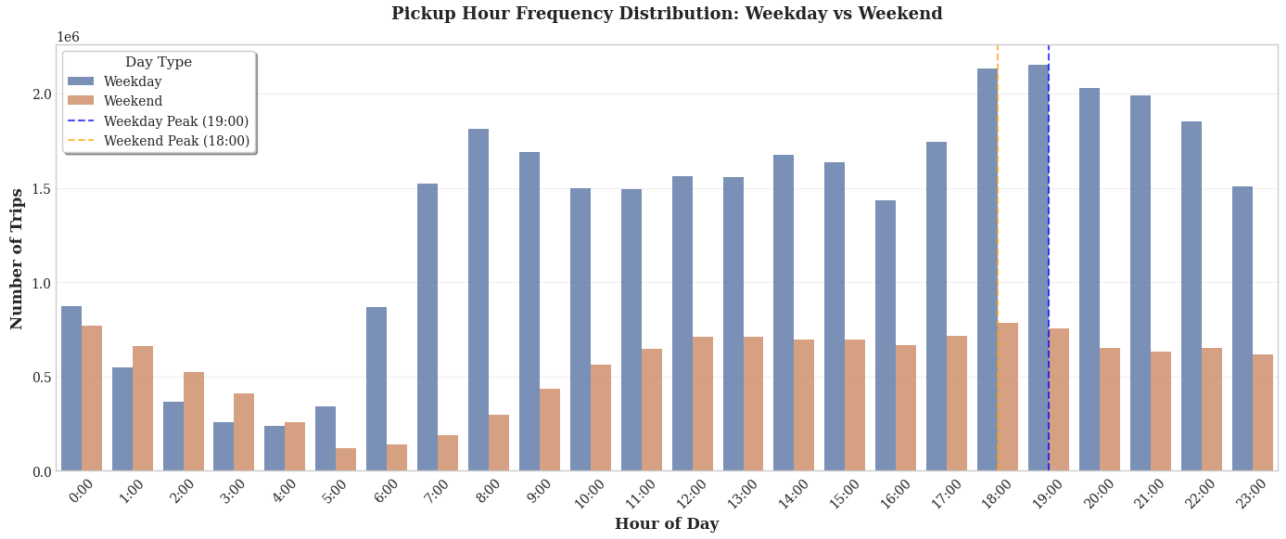

Figure 7 *Pickup time frequency distribution by weekdays*

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 spacial features into unified spatiotemporal distance metric for to use an input in DBCSAN algorithm as shown in the equation below (), however it was not efficient due to very high density of the data, with algorithm not able to distinguish any patterns correctly.

3.4 Data Splitting

In further step the data was aggregated to form a demand matrix with 1 hour window aggregated demand per each hierarchical cluster - to form the chronologically soreted initial dataset to feed into the models. The training part consists of 70% of data, while validation makes additional 10%, with the last 20% reserved for testing.

3.5 Model comparison

(PAPILDYTI)

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

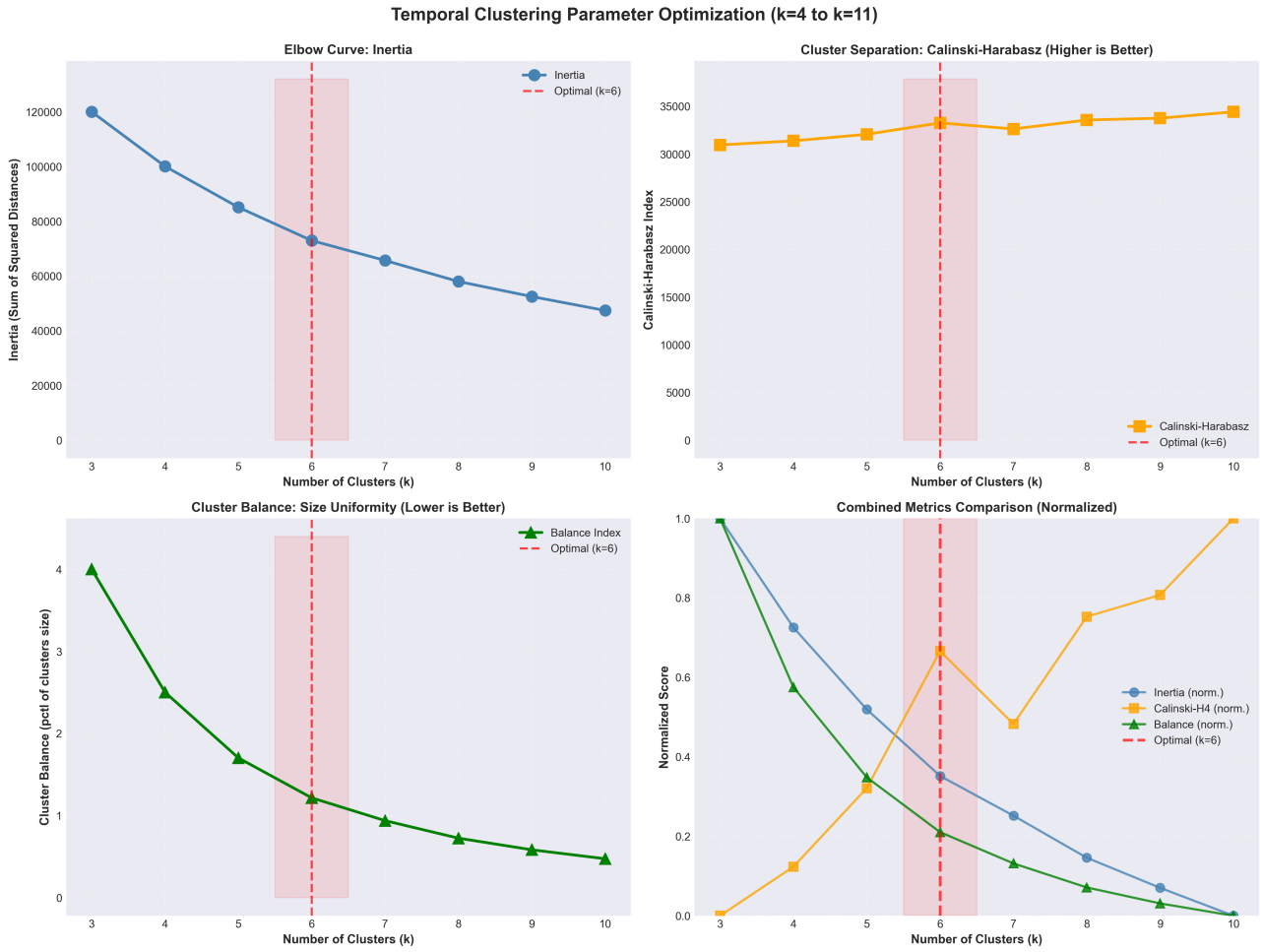

Figure 8 *Temporal clustering optimization*

For first part of combined clustering where K-means algorith was applied to temporal data it was needed to provide the finite number of clusters to capture all the patterns of time. For this decided to run parameter optimization - test the clustering from 4 to 11 number of clusters and smaller data sample of 500 000 trips to see how metrics change. Instead of looking solely at silhouette score as it might be not so informative on so dense concentrated data, it was decided to evaluate how will the change of numbers of cluster change the inertia and clusters balance, as well as calculated Calinski-Harabasz index. In Figure 8 it can be seen that, evaluating the elbow curve and finding the local maximum in Calinski-Harabasz index, the optimal point was selected with number of clusters being 6. The overview of the 1.5m trips sample of final six temporal clusters can be seen in graphs in Figure 9 - for example, T0 cluster being 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 T4 cluster concentrating mostly on weekends - Saturday and Sunday evening and night, continuing into Monday early night until morning.

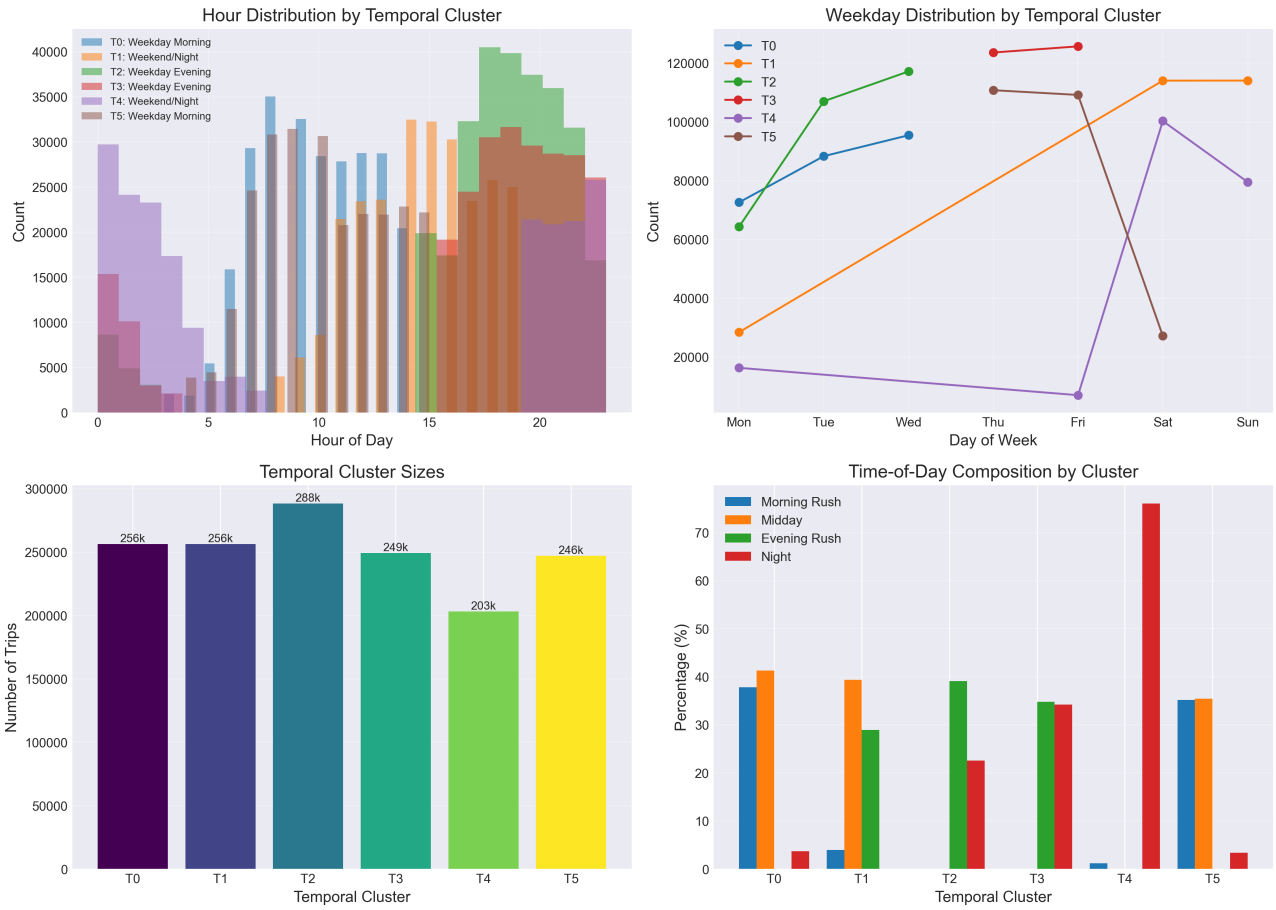

Figure 9 *Temporal clusters analysis*

3.6.2 Optimization for spacial clustering

For spacial HDBSCAN clustering for the first temporal cluster the test run of different parameters, such as minimum cluster size and minimum samples was also conducted, and the results can be seen in a heat map in Figure 10 - but that did not solve the question of what should be the best parameters for each model - 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 for clusters of different densities apply different parameters, this solved the problem and made algorithm working well on different temporal clusters.

With this approach i was possible to achieve similar size clusters distribution, as can be seen in Figure 11, where in the top right graph top 20 clusters by size are of adequate comparable sizes, and also trip characteristics for top 15 clusters such as average distance and average fare seems also look reasonably similar, as can be seen in the bottom right graph. Lower left graph shows that geographic spread in a combination with sizes is not that high among clusters.

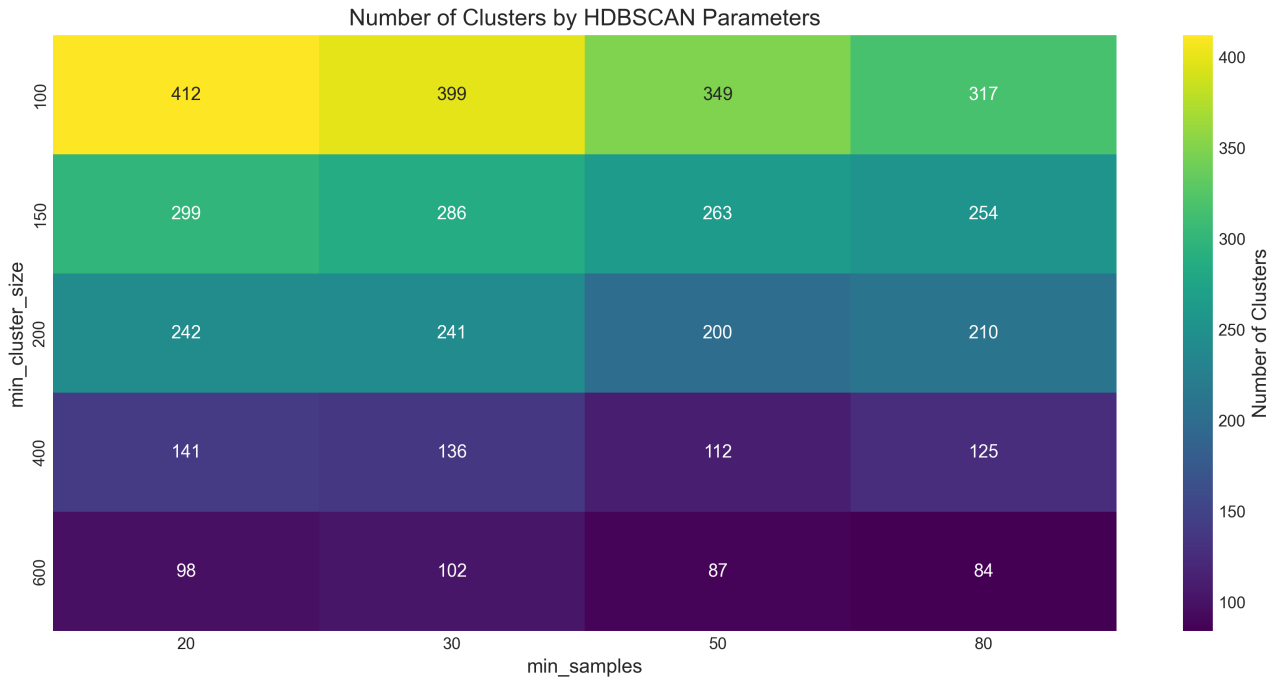

Figure 10 *Heatmap os spacial clustering parameters*

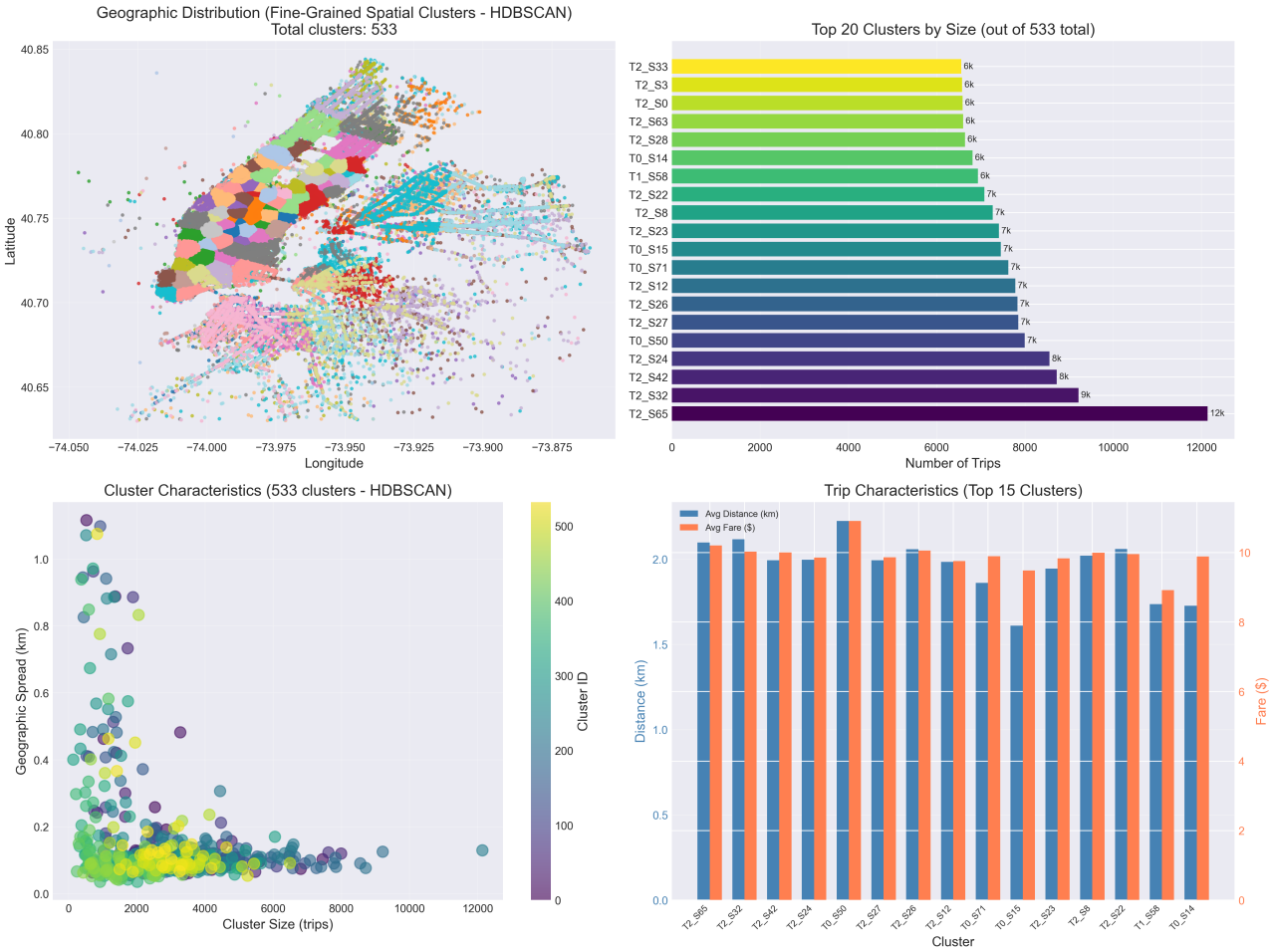

Figure 11 *Spacial clustering analysis*

3.6.3 Optimization for SARIMA

3.6.4 Optimization for XGBoost

Transfer Learning of Hyperparameters - 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 statiotemporal clusters. Tuning on the most data-rich cluster ensures the signal would be most robustly captured, avoiding overfitting to noise in smaller, sparser clusters.

With this approach, only selected top cluster by hourly demand was put into lightweight hyperparameter grid search tuning for XGBoost. The possible parameters combinations were taken as following:
'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 it was used the mode of the best parameters from top 5 clusters to create a robust set of hyperparameters for the remaining clusters, as the hyperparameters that work best for the top 5 heavy-traffic clusters will likely work very well for the smaller ones too.

3.6.5 Optimization for LSTM

When applying LSTM model, the number of optimizations were implemented to increase the performance of the model

3.7 Evaluating

In this section the evaluation of the models is done, using the described in methodology section evaluation metrics.

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

To sum up the aim of the paper the following results were concluded during the done research:

- SARIMA model, as expected, was not able to model and forecast properly, due to high amount of volatility in the data in a combination with sparsely or zero inflated data - it's multi-step forecasting inevitably converges to zero and was not able to predict peaks effectively.
- VAR model, even it was able to predict up to 10 clusters' demand, is not suitable for such high amount count of clusters due to the curse of dimensionality, as (PAPILDYTI)
- Both XGBoost and LSTM models, when tuned and optimized, proved to be robust and showed good performance in forecasting the taxi demand among different clusters.
- Different models performance differs on 1-day, 3-days and 7-days forecast horizons.

4.2 Discussion

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.

Another possibility would be to shorten forecasting horizon or make it in blocks, meaning first forecast first from 24 to 48 hours ahead, and then re-fit with updated data, so this should in theory reduce the recursive error accumulation.

4.3 Conclusions

SARIMA captures the cyclical peaks but displays them temporally displaced, reflecting a fundamental limitation of how ARIMA-class models process temporal information through differencing and filtering. For NYC hourly taxi demand data it shows as the model was able to learn the shape of the demand cycles, however, losing the precise temporal registration of actual events. The temporal increases (i.e. peaks) were caught because of the weekly seasonal component was detected correctly, but the non-seasonal differencing causes the forecast to lag. SARIMA cannot capture those non-linear events without external regressors. This explains why, in comparison, LSTM and XGBoost track peaks more accurately, as they implicitly learn non-linear relationships through hidden layers and tree ensembles, as was discussed in Gifty and Li work [1].

LSTM and XGBoost models are robust and could predict taxi demand to a reliable degree.

References and sources

[1] 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.

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

[3] 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.

[4] 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: http://dx.doi.org/10.1109/MFI49285.2020.9235263.

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

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

Appendix 1. Python Code

The code, written in Python, which was used for data processing and clustering algorithms as well as the pipeline for forecasting using the models can be found on Github repository, together with the instructions to reproduce results, by the following link:
The Github repository

Appendix 2. Using AI tools

AI tool, in particularly 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, also as a helper for enhancing the code to produce required visualizations, optimization of code computational performance and ensure code reusage, and adding logging and performance monitoring layers to maintain understandable running code.