Data Science for Transportation (2023) 5:19
https://doi.org/10.1007/s42421-023-00075-w

RESEARCH

Exploring Deep Learning Approaches for Short-Term Passenger Demand Prediction

Zahra Ghandeharioun[1] · Parham Zendehdel Nobari[1] · Wenhui Wu[1]

Received: 20 April 2023 / Revised: 19 June 2023 / Accepted: 13 July 2023 / Published online: 23 August 2023

Abstract
An accurate short-term passenger demand forecast makes a contribution to the coordination of traffic supply and demand. Forecasting the short-term passenger demand for the on-demand transportation service platform is of utmost significance since it might incentivize empty cars to relocate from over-supply regions to over-demand regions. Yet, because spatial, temporal, and exogenous dependencies need to be evaluated concurrently, short-term passenger demand forecasting may be rather difficult. This article aims to investigate several methods that can be utilized to forecast short-term traffic demand, with a primary emphasis on deep learning approaches. We examine varying degrees of temporal aggregation and how these levels affect various architectural configurations. In addition, by analyzing 22 models representing 5 distinct architectural configurations, we illustrate the influence of varying layer configurations within each architecture. The findings indicate that the long-term short memory (LSTM) structures perform the best for short-term time series forecasting, but more complex architectures do not significantly enhance the outcomes. Moreover, considering the spatiotemporal aspects results in an improvement in the prediction of more than fifty percent. In addition, we investigate the vectorization of time, also known as Time2Vec, as a way of embedding to make it possible for a selected algorithm to recognize periodic characteristics in time series, and we show that the outcome is improved by fifteen percent.

Keywords Demand prediction · Spatio-temporal features · On demand services · Deep learning

Introduction

On-demand transportation services can now be provided in an unique and popular way due to the rise of online apps and platforms for car-hailing services. Nearly, every transportation service advertises that they are committed to attracting more customers. Hailing a taxi on street is getting less popular as people would rather use taxi service platforms such as Uber, DiDi and Lyft, which are gaining popularity at a rapid rate, to request a pickup on their smartphones. This is because these platforms offer more customization options.

✉ Zahra Ghandeharioun
zahrag@ethz.ch

Parham Zendehdel Nobari
parhamz@student.ethz.ch

Wenhui Wu
wenhwu@student.ethz.ch

[1] Department of Civil, Environmental and Geomatic Engineering, Institute for Transport Planning and Systems, ETH Zurich, 8093 Zurich, Switzerland

Several of the companies that provide on-demand transportation are losing money due to an imbalance between supply and demand. The cause of the imbalance between supply and demand is, on one hand, long wait times for passengers who are located in an area where there are no available taxis, and, on the other hand, idle taxis and the drivers who are lingering around to find customers. Both of these factors contribute to the problem. This leads in a loss of income not only for taxi firms but also for individual drivers, as well as a waste of time for passengers who are waiting.

The forecasting of short-term taxi demand is an essential component of the solution to the problem of an imbalance between supply and demand for taxi companies. If taxi businesses were to have accurate demand predictions in advance, they would be able to pre-allocate taxi fleets from areas of oversupply to regions of excess demand in order to fulfill the demand of passengers and enhance the performance of their services. Both transportation operators and passengers would benefit from proactive decision assistance provided by predictive data analytics.

Therefore, the subject of demand forecasting has constantly been under study by many researchers over the last several decades, and a large number of solutions have been proposed. These solutions range from model-based time series analysis through machine learning approaches and model-free deep learning models.

The conventional method of analyzing time series makes use of statistical models, which may provide accurate projections of future values based on the recurrence of historical patterns. The most well-known ones are, the Bayesian Forecasting, the autoregressive integrated moving average (ARIMA) model and the Kalman filter Guo et al. (2014). These techniques have been used in a wide variety of more complex statistical models, including the applications in the prediction of traffic flow Vlahogianni et al. (2004); Williams and Hoel (2003); for example. These methods, are dependent on the particular mathematical assumption about the input data, which limits the application of this methods. Moreover, the mathematical assumptions constrains coping with the complex properties of the data collected from a wide variety of sources.

Data-driven methods try to tackle the same problem of imbalance between demand and supply by considering the different data sources. These data can be categorized by three aspects, called dependencies according to Zhang et al. (2017).

Temporal dependencies: passenger demand has a significant periodicity (for example, it is predicted to be high during morning and evening peaks and low during sleeping hours), and short-term demand is reliant on the trend of the closest previous demand.

Spatial dependencies: passenger demand in one zone was endogenously reliant on all zonal variables in the network in Yang et al. (2010). An enhanced model that can capture local spatial dependencies is needed since adjacent factors affect more than distant variables.

Exogenous dependencies: weather conditions, points of interest (POI), and travel time rates may strongly affect short-term passenger demand. Exogenous variables have temporal and spatial interdependence.

In this work, we tackle the prediction of the demand for on-demand services primarily by deep learning approaches. In particular, we conduct a cross-comparison between models and datasets. We investigate the effect of temporal aggregation and consideration of spatial and exogenous dependencies. The contribution of our work is as follows:

- This paper studies short-term demand prediction for on-demand services.
- The short-term demand prediction model studies different data aggregation levels of the input data and shows how this affects the prediction results.
- The current study includes independent and dependent temporal and spatiotemporal variables, considering the characteristics of demand prediction.
- In the current work, we provide a representation of time, in the form of vector embedding, to automate the feature engineering process and model time better.
- The method presented in this paper is compared with classical machine learning methods to show performance of the algorithm.

The rest of the paper is organized as follows. In Sect. 2, we review the most relevant literature on the short-term demand prediction for on-demand services in terms of motivation and methods. In the subsequent section, we state the problem at hand, and in Sect. 4, we explain all the models studied in the current work. The datasets and their features are presented in Sect. 5. Finally, the results and conclusions are provided in Sect. 6 and Sect. 7 respectively.

Literature Review

We investigate two aspects in the literature that underpins our paper: (1) Motivations for accurate short-term demand prediction for on-demand services (2) machine learning methods that have been used in the short-term prediction of on-demand or similar passenger demand prediction conditions.

Motivations for Accurate Short-Term Demand Prediction for On-Demand Services

The placement of idle cars to predict future demand and operating states is crucial for the operation of on-demand services like taxis, dynamic ridesharing, or vehicle sharing Sayarshad and Chow (2017). Demand prediction for optimizing the operation of on-demand mobility systems is studied in different approaches Zardini et al. (2021). Fleet operators rely on estimates of upcoming user requests to efficiently place empty vehicles and manage their fleets of vehicles Dandl et al. (2019). Yang et al. (2002, 2010); Yang and Yang (2011) developed a meeting function to define the search frictions between drivers of unoccupied taxis and waiting passengers in light of the fact that they cannot be matched concurrently in a certain zone. The meeting function made it clear that the density of waiting passengers and available taxis in a given zone at a given time determined the meeting rate, indicating that the waiting time for passengers, the searching time for drivers, and the arrival rate of passengers (demand) were all endogenously correlated.

When the arrival rate of empty cabs perfectly matched the arrival rate of waiting passengers, the equilibrium condition was attained. The exogenous factors, such as the number of taxis in the fleet and the fare per trip, had an impact on this equilibrium state as well as the endogenous variables. The taxi operator may use the on-demand service platform to coordinate supply and demand, therefore, influencing the equilibrium state by controlling the entrance of taxis and setting the taxi price structure, such as non-linear pricing Yang et al. (2010). However, researchers discovered that when there was an excess of empty taxis or waiting passengers in that location, a regional disequilibrium would arise Moreira-Matias et al. (2013). Due to this imbalance, resources may not meet supply and demand, resulting in poor taxi usage in certain areas and low taxi availability in others. The taxi operator must thus prioritize developing a short-term passenger demand forecasting model that can be used to perform effective taxi dispatching and expedite route finding to reach equilibrium across metropolitan regions Zhang et al. (2017). Additionally, when demand is unknown, drivers frequently behave extremely differently. For instance, if parking is available, drivers might simply wait in one spot. Alternatively, one could wander the streets. Due to increasing traffic congestion and the diverting of passengers from public transportation, they are likely to have a negative impact on the environment Zhang and Zhang (2018). However, some drivers go in advance to possible passenger pickup sites based on past knowledge of demand patterns. Improved operations using algorithms targeted at enhancing empty vehicle routing and repositioning for both taxi and ridesourcing systems is a vast area of research Yu et al. (2019); Chen et al. (2021). There are many other applications for more precise forecasting. A more precise dynamic surge pricing setting is made possible by knowing which areas will likely have higher demand in the upcoming timestep Iglesias et al. (2018); Chen (2016). When demand is strong in a particular region, and at a particular time, ride-hailing businesses may use dynamic pricing for a variety of reasons, such as weather-related events, special occasions, and so forth. Previous studies have shown that the ride-hailing compensation model is problematic for drivers because there is a risk that they will not have a job due to demand uncertainty, which results in lost wages for drivers as they wait for new passengers and are unsure of where demand may be for high-yielding rides that could potentially provide them with a source of income Chen et al. (2021); Li et al. (2019); Zoepf et al. (2018). Surge pricing has a detrimental impact on passenger demand as well Chen et al. (2015). The issues that can arise in on-demand services include cancellation by the passenger due to a long wait before being assigned to a vehicle, cancellation by the passenger due to a longer-than-expected pick-up time after being assigned a vehicle, and passenger reorder and rebooking after cancellation, despite the fact that passenger behavior in on-demand services has not been as thoroughly studied as driver behavior Wang and Yang (2019); Chen et al. (2021).

Machine-Learning Methods for short-term ride-hailing demand prediction

Short-term prediction of transport demand is a topic, which attracts many researchers. Vlahogianni et al. (2004), in his study of the literature on short-term traffic forecasting, noted that due to the rapid advances in data accessibility and computing capacity, researchers were switching from traditional statistical models to neural network-based methodologies. Deep learning in particular has been widely used in transportation state prediction Ke et al. (2021). And new opportunities arises with advances in deep learning techniques to address short-term transport behavior. Deep learning often involves the training of convolutional neural networks (CNNs) which are capable of capturing high-order spatial-temporal correlations in transportation prediction problems. There is a broad range of problems in the domain of transportation, which are similar to short-term passenger demand forecasting. Researchers have used CNNs for a variety of prediction tasks, such as speed evaluation Ma et al. (2015), bike usage prediction Zhang et al. (2016), and demand-supply prediction for ride-hailing services Ke et al. (2017). To analyze time series data, a number of deep learning models have been proposed, and they have demonstrated cutting-edge performance in practical applications. For instance, Huang et al. (2014) introduced a multi-task learning structure to perform road traffic flow prediction and provided a deep belief network to detect the spatio-temporal properties. Cheng et al. (2016) proposed a DL based approach to forecast day-to-day travel demand variations in a large-scale traffic network. Similarly, Lv et al. (2015) used a stacked autoencoder model based on traffic prediction method. Ma et al. (2015) extended the deep learning theory for the large-scale traffic network analysis, and predicted the evolution of traffic congestion with the help of taxi GPS data. Recurrent neural networks (RNNs) and their extensions such as long short-term memory (LSTM) are well fit for processing time series data streams. Xu et al. (2018) applied LSTM to predict taxi demand in New York City. Some researchers integrated RNNs with CNNs to make full use of spatial-temporal information to forecast short-term ride-hailing demand Ke et al. (2017),[29] studied different pre-training approaches of DNN for traffic prediction.

Extensions to the integrated deep learning algorithms have been made, drawing on but not limited to the CNN and RNN mechanisms. To forecast the demand for taxis,

Li et al. (2019) created a contextualized spatial-temporal network that includes local spatial context, temporal evolution context, and global correlation context. For the purpose of forecasting demand for ride-hailing services, Geng et al. (2019) put out a spatial-temporal MCG (STMCG) model that makes use of non-Euclidean correlations. Zhou et al. (2018) created an attention-based deep neural network to estimate multi-step passenger demand for bikes and cabs based on an encoding–decoding structure between CNNs and ConvLSTMs. An LSTM model is used to simulate the temporal features as well as other traffic-related data in Yang et al. (2019)'s hybrid deep learning architecture.

Short-term passenger demand depends on additional explanatory factors in addition to its own spatiotemporal characteristics (some with spatiotemporal properties and some only with temporal properties). The proposed architectures, a deep learning CNN structure, are developed in the next section. In contrast to earlier temporal convolutional networks, it uses raw counts of ride-hailing pickups along with a variety of temporal and spatial features (such as socioeconomic variables, spatial heterogeneity, weather, etc.) that are used in a multivariate architecture for the effective short-time demand prediction of ride-hailing services. The established deep neural network's learning capacity has been improved thanks to the CNN architecture and the utilization of temporal and spatial information.

Research Problem

The goal of the short-term demand forecast is to predict the number of on-demand taxi ride requests that will be needed at some point in the future in a certain unit area by taking into account the requests that have been placed in the past. In order to forecast the demand for on-demand services, in addition to using data from past demand, we also consider temporal and spatiotemporal features. Generally, the problem can be seen as a time series forecasting problem. There are several time-series prediction algorithms. In the following section, we explain the studied models. Moreover, we propose a representation of time in the form of vector embedding so that the feature engineering process may be automated and time can be modeled more accurately. The vector embedding of time is then combined with machine learning methods.

Modelling

Most short-term demand forecast systems in transportation have traditionally relied on running models using univariate trip count data obtained by loop detection, GPS, and other sources Vlahogianni et al. (2004). The trip count is a continuous and time-dependent variable that constitutes time-series data. Predicting time-series data falls under the regression class of machine learning algorithms.

We devised five models that are currently widely used in the literature for the prediction task: Random Forest (RF), Long Short-term Memory (LSTM), Convolutional Neural Network (CNN), LSTM-CNN autoencoder, and Deep CNN. Furthermore, we combined the vectorization of time (Time2Vec Kazemi et al. (2019)) as a layer to stack it with other models and try its power in our case study. In the following, we briefly explain each model.

Random Forest Regressor

Random Forest is an ensemble learning approach, which can undertake classification and regression tasks. A random forest is an ensemble of unrelated decision trees. These multiple decision trees are averaged to build a more robust model with a better generalization performance and less susceptibility to overfitting Breiman (2001). It is implemented with sklearn.ensemble.RandomForestRegressor(). Within the random forrest regressor, four hyperparameters are tuned: Max depth, which is the maximal length of a path from the decision tree root to the leaf; Max features, which is the maximum number of features that are examined for the splitting of each node within the decision tree; Min samples split, which is the minimum samples limit that is imposed to stop the further splitting of nodes; N estimators, which is the number of trees in the forest. The tuning process is conducted by means of an exhaustive grid search. Finally, model RF-Model 1 is the model with 200 trees; RF-Model 2 is the model with 50 trees; and RF-Model 3 is the tuned forest. The grid search space of the tuned model for the different

Table 1 Grid search space for RF-Model 3 hyperparameter tuning across different datasets

Dataset	Dataset A	Dataset B	Dataset C
Number of fits	750	2500	1080
max-depth	[3, 4, 5, 6, 7]	[7, 10, 12, 15, 18]	[7, 10, 12, 15, 18]
max-features	[3, 4, 5, 6, 7]	[3, 4, 5, 6, 7]	[6, 15, 21, 28]
min-samples-split	[3, 6, 12, 18, 24]	[4, 12, 24, 96]	[20, 387, 1574]
n-estimators	[50, 70, 100, 200]	[30, 50, 70, 100, 200]	[20, 30, 50, 70, 100, 200]

datasets is presented in Table 1. The hyperparameter space was adjusted according to the number of data points within each dataset.

One of the major pros of the random forest model is that it is an interpretable transparent model. Due to this model's attribute, a relative feature importance analysis can be carried out to determine which features have a more pronounced effect on the prediction outcome. Furthermore, it is also possible to extract a random tree from the ensemble and to examine the splitting and the residual errors after each split according to a feature. In any decision tree, the upper levels of the tree usually comprise splits based on more important features. The deeper we run down the tree levels, the more the data's variance is covered by more and more splits of less important features. The feature importance analysis of our dataset will be presented in Sect. 5.

Long Short-Term Memory Network (LSTM)

Long Short-term Memory network is a special type of Recurrent Neural Network. An LSTM unit consists of the cell, input, output, and forget gates. The model is implemented with Tensorflow.keras.layers.LSTM(). The relevant hyperparameter is the number of LSTM units. The model takes transformed timestep window tensors as input. The different timestep window sizes for all datasets are shown in Table 2. Six different architectures are designed for the task. The schematic diagrams of the architectures are shown in Fig. 16.

Models 1 and 2 share the same architecture. In the first layer, successive LSTM units learn the demand patterns which are then passed to a neural layer with eight nodes which interprets the features further and then feeds it to the output layer with one node for the single output variable. Models 3,4 and 5 share the same two-layer stacked LSTM architecture. The difference between M3, M4, and M5 is the hyperparameter settings. Finally, there is M6 which has three stacked LSTM layers and is the deepest of all architectures.

Time2Vec For a variety of challenges involving sequence modeling, recurrent neural networks (RNNs) have shown outstanding results. The majority of RNN models assume that inputs are synchronous and do not include time as a characteristic. Since time is recognized as a crucial component, it is often included as yet another input dimension. In actual application, RNNs often fall short of adequately using time as a feature. Many researchers create hand-crafted time features tailored to their particular problems and input those characteristics into the RNN to aid in better use of time. But, hand-crafting features can be costly and demands subject-matter knowledge.

In the current work, we implement Time2Vec as explained in Kazemi et al. (2019) and adopt their solution in developing Neural Network with LSTM model explained above. Their goal is to provide a vector embedding for the representation of time. Mathematically, the implementation of Time2Vec is as follows:

$$t2v(\tau)[i] = \begin{cases} \omega_i\tau + \phi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i\tau + \phi_i), & \text{if } 1 \le i \le k. \end{cases} \quad (1)$$

Where k is the Time2Vec dimension, τ is a raw time series, $\mathcal{F}$ is a periodic activation function, ω, and ϕ are a set of learnable parameters. In order to enable a chosen algorithm to detect periodic behaviors in data, we set $\mathcal{F}$ to be a sin function. In addition, the linear term captures non-periodic patterns in the input that rely on time while also representing the passage of time. Several architectures may simply apply this vector representation of time due to its simplicity. In this instance, by changing a simple Keras dense layer, we attempt to translate this idea into a Neural Network structure. This custom layer's output consists of the user-specified hidden dimension ($1 \le i \le k$), which comprises the network's learned sinusoids and a linear representation of the input ($i = 0$). With this instrument in our possession, all we have to do is to stack it with more layers to test its effectiveness in our case study. We stack the Time2Vec layer on the best-performing LSTM from the architectures shown in Fig. 1 and show the results in Sect. 6.

Convolutional Neural Network (CNN)

CNN is an artificial neural network that is modelled after the visual cortex. In a convolutional layer, the algorithm carries out a mathematical operation called convolution on a rolling kernel across the input space. Convolutions include different filters which extract feature maps from the input space. CNN is implemented with Tensorflow.keras.layers.Conv1D(). Three relevant hyperparameters are considered: input shape (units take input values in time step format), filters (number of filters in the convolutional layer), kernel size (length of the one-dimensional convolutional window). The timestep window setting is the same as in LSTM models (see Table1). Two models with different numbers of filters are designed (See Fig. 2).

Table 2 Timestep window setting for data transformation

Dataset	Timestep window size	Interpretation	Input shape
A (Temporal 1 hr)	3	3 h	(3, 8)
B (Temporal 15 min)	2	30 mins	(2, 9)
C (Spatio-Temporal 15 min)	8	2 h	(8, 34)
C (Deep Architecture)	96	6 h	(96, 34)

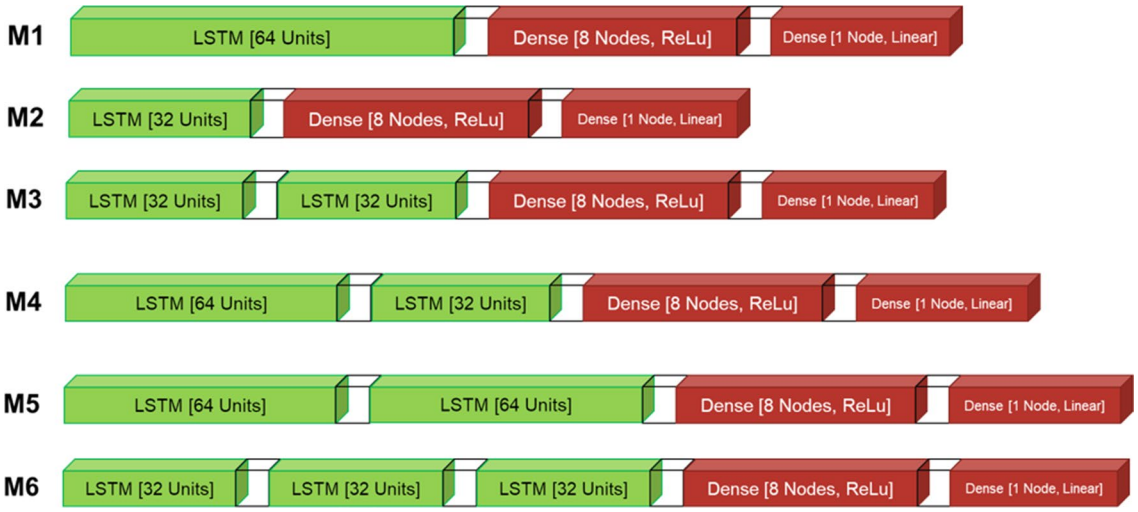


Fig. 1 Different LSTM architectures with various hyperparameter settings

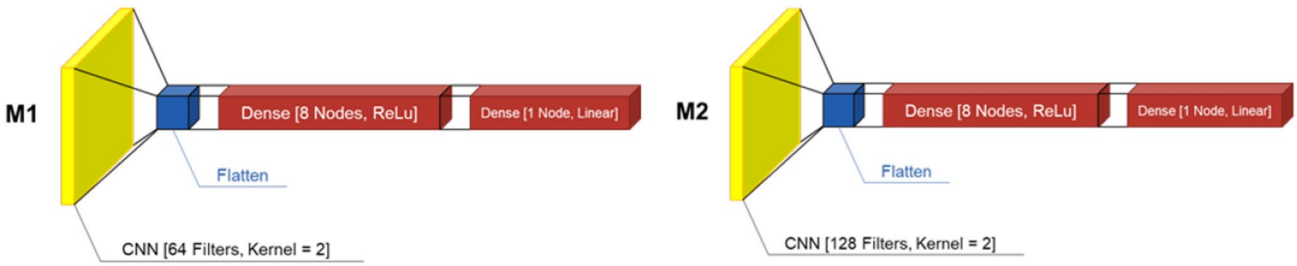


Fig. 2 CNN architecture with various hyperparameter settings

In this architecture, the CNN layer extracts feature maps from the transformed input tensor, and then a flattening layer feeds the outputs of the CNN layer to a fully connected layer to articulate the features. Finally, same as before, one node outputs the target variable.

LSTM-CNN Autoencoder

This model is a hybrid of CNN and LSTM models, using a CNN layer as an encoder and an LSTM layer as a decoder. Implementing methods include four-layer functions from Tensorflow.keras.layers, including LSTM(), Conv1D(), and Maxpooling1D(). Max pooling is used to reduce the dimensionality of the feature map so higher-order patterns can be extracted. There are five relevant hyperparameters: input shape (units take input values in tuples of time step format), filters (number of filters in the convolutional layer), kernel size (length of the 1D convolutional window), pool size (size of the 1D max pooling window), and units (number of LSTM units). The same time step windows in Table 2 were used. The Four different architectures are shown in Fig. 3.

The models share the same architecture, and their differences lie in their varying hyperparameter settings. The repeat vector creates LSTM unit readable input tensors from flattened data.

Deep CNN

In deep CNN architectures, convolutional layers are stacked using max pooling layers. This is realized by four functions from Tensorflow.keras.layers: Conv1D(), MaxPooling1D(), BatchNormalization(), and Dropout(). Compared with CNN, there are two additional hyperparameters. Pool size is the size of the 1D max pooling window, while dropout rate is the dropping rate of units during training. Table 2 shows the time step window configuration, which is 96 instances. The window size is set large enough to enable a multi-layer stacked architecture. Successive convolutional and max pooling layers with kernel sizes and pool sizes larger than

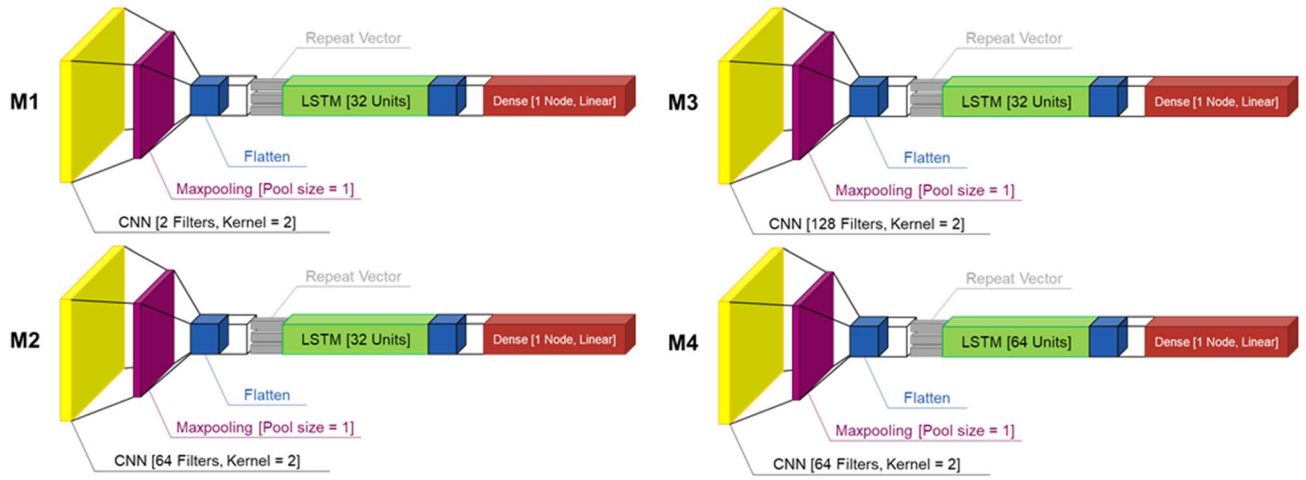


Fig. 3 LSTM-CNN autoencoder architecture with various hyperparameter settings

Fig. 4 Different deep CNN architectures with various hyperparameter settings

1 rapidly reduce the input shape dimensionality, which is why a larger window size is warranted. As a result of these stacked layers, the model should theoretically be able to extract more far-reaching patterns. Figure 4 shows the six different models. Models M1 and M2 share the same architecture and have different hyperparameter settings. They include three convolutional layers followed by three max pooling layers. The resulting feature maps are flattened and are connected to a one-node output layer. Model 2 has more convolutional layers compared to model 1. Model 3 has another fully connected layer with 32 nodes which is close to the independent feature count of 28. Model 4 uses an additional layer of batch normalization following each convolutional layer which standardizes the layer inputs. Models 5 and 6 make use of a 20% dropout in the second and second to third max-pooling layers, respectively. Dropout acts like a mask that randomly nullifies the contribution of some neurons randomly during training, creating a large

number of neural networks with different architectures in parallel. This is an effective regularization method to reduce overfitting and improve generalization error in deep neural networks of all types.

Data

In this project, we build a multivariate time-series dataset using a variety of variables provided by transport operations and travel behavior study fields to be useful to demand prediction. These criteria include temporal, meteorological, socioeconomic, and demographic factors. As a result, the data are grouped into three categories and comes from four separate sources.

First, the trip data includes green taxi trip data. The data were collected from the New York City Taxi & Limousine Commission (NYC TLC) NYC (2019) and are available to the public. Furthermore, in order to capture interannual demand patterns for green taxis we use 2 years of green taxi pickup data from 2017 to 2019. This decision stems from two reasons: firstly, we decided to set the study period cutoff date before 2020 to avoid the impact of COVID-19 and its lockdowns on travel behavior secondly, from 2017 onwards the trip records pickup and drop-off locations are formatted according to the NYC TLC's own taxi zones (see Fig. 5) as opposed to coordinates in previous records. The coordinates formatted trip records are more tedious to work with and are not as precise due to GPS logging errors as shown in Fig. 6, where a considerable number of pickup records fall outside the study area. Taxi zones formatted data is more suitable for integrating spatial features and is less prone to errors.

The green taxi data have detailed information about each trip. The most important and relevant variables in these datasets are the location and time of pickup and drop-off since they are the keys to spatial data augmentation and temporal features such as day, hour, weekday, and the like can be extracted from them. The location variable is the code of the taxi zones where pick-up or drop-off happens and the time variable is when, respectively, the trip starts and ends in minutes.

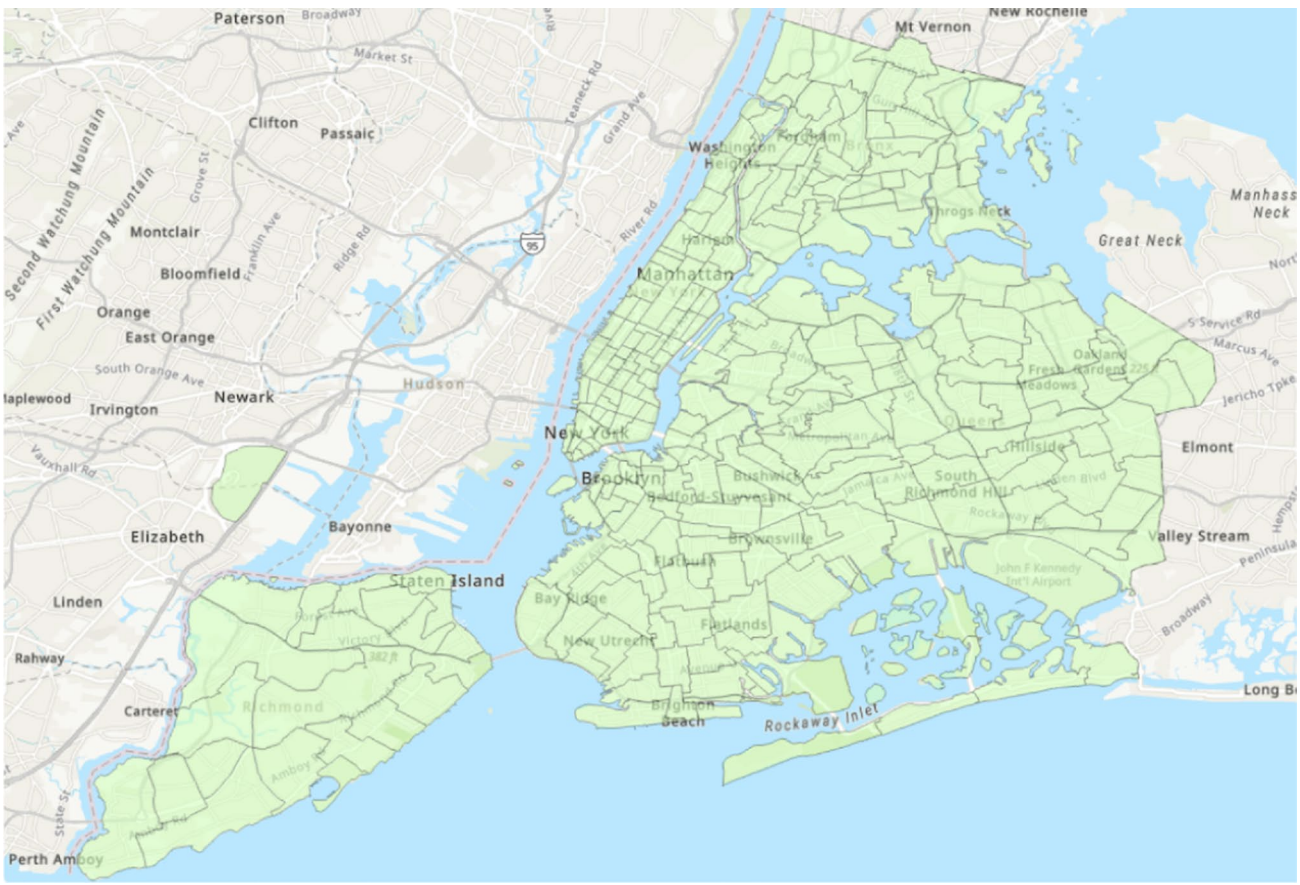


Fig. 5 Taxi zones in New York City according to NYC TLC NYC (2019)

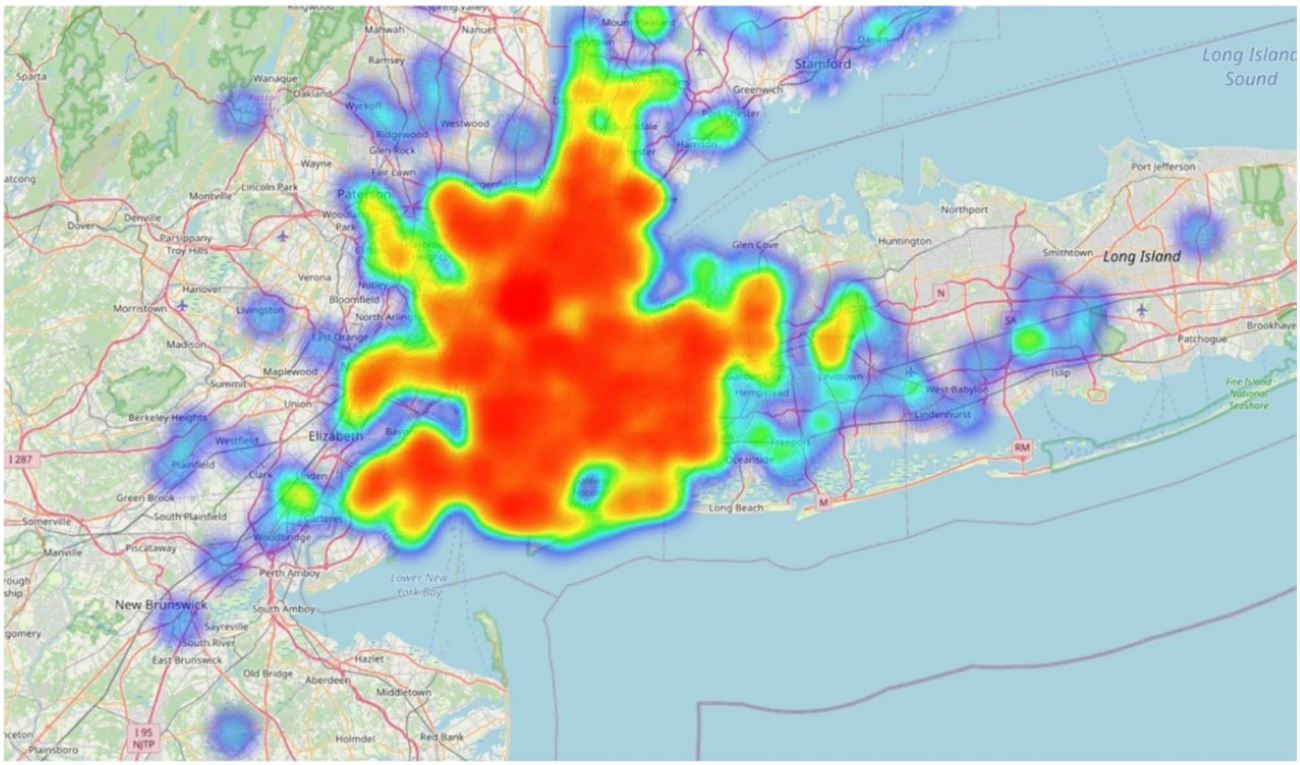


Fig. 6 The spatial distribution of pickup coordinates from green taxi trip records

Second, the socio-demographic features are gathered from the US Census Bureau. These features are arranged in five different types of features: population, housing unit, social characteristics, economic characteristics, and household characteristics. All features and their respective descriptions are shown in Table 3.

Third, climate features are also included. They are provided by the US National Oceanic and Atmospheric Administration. The New York Central Park daily weather data are used to describe meteorological and inclement weather impacts. The dataset has not only information on temperature and precipitation, but also data on heating degree days, cooling degree days, snowfall, and snow depth.

Finally, point-of-interest data are obtained from NYC Open Data (2023) and categorized in six categories explained in Table 3

Data Aggregation and Exploratory Data Analysis

To carry out the predictions, the trip data are aggregated in both temporal and spatial bins. The rationale behind this aggregation is two-fold: firstly, ride-hailing service providers usually look for demand forecasts in a given region for the next 15 min or 1 h to dispatch vehicles, and secondly, aggregation results in fewer data instances and less model training time as a result. In order to account for these needs and limitations, both temporal aggregation and spatial aggregation are employed. To investigate the effects of temporal aggregation on prediction accuracy, we apply two temporal aggregation schemes: a 15-min aggregation and a 1-h aggregation scheme.

The trip counts resulting from different temporal aggregation schemes are shown in Fig. 7. In the first week of 2017, clear patterns of daily traffic peaks are discernible in both 15-min and 1-h aggregation. A morning peak and an evening peak can be observed nearly every day. In the first month of 2017, a regular pattern across weeks is shown in both 1-hour and 15-minute aggregated data in the middle row of Fig. 7. Finally, the trend for 2017 shows that the demand for green taxis is higher in the colder months and decreases with warmer weather, which further indicates the seasonality of green taxi demand and underlines the importance of including climate features for prediction purposes.

Spatial aggregation aims to further augment data with spatial features (socio-demographic and climate features). The trip data and the spatial features have different spatial resolutions: trip data are at the taxi zone level, while the spatial features are at the borough level. Eventually, all the trip data are aggregated to the borough level so they can be joined with spatial features. There are two major reasons why taxi zone level aggregation is not pursued in this

Table 3 List of all selected features in dataset C

Independent variable name	Variable description	Variable type
LocationID	TLC Taxi Zone in which the taximeter was engaged	int
LocationID	TLC Taxi Zone in which the taximeter was disengaged	int
trip_distance	The elapsed trip distance in miles.	float
total_amount	The total amount charged to passengers. Does not include cash tips.	float
RINTERNATIONALMIG	Net international migration rate	float
household_estimated	Annual estimates of number of housing units	float
male_rate	Male rate among 15-year-olds and over	float
employment_rate	In-labor-force rate among 16-year-olds and over	float
mean_commute_min	Mean travel time to work in minutes	float
other	Mode share of means of transport other than driving, PT and walking	float
3+_veh	Households with more than 3 vehicles	float
temp_min	Lowest temperature of the day (F)	float
temp_depart	Daily temperature departure from normal (F)	float
precipitation	Rain intensity (inch)	float
new_snow	Snowfall (inch)	float
snow_depth	Depth of snow (inch)	float
year	The year of trip	int
month	The month of trip	int
day	The day of trip	int
weekday	The day of the week in which trip took place	int
hour	The hour of trip	int
weekend_true	Whether the trip took place on a weekend	int
POI_Density_Cat1	Residential	float
POI_Density_Cat2	Transportation facility, Government Facility	float
POI_Density_Cat3	Cultural, Recreational, Religious Facility	float
POI_Density_Cat4	Social and Health facility	float
POI_Density_Cat5	Commercial POIs	float
POI_Density_Cat6	Education facility	float
Borough_Bronx	Whether the trip took place in the Bronx	int
Borough_Brooklyn	Whether the trip took place in Brooklyn	int
Borough_Manhattan	Whether the trip took place in Manhattan	int
Borough_Queens	Whether the trip took place in Queens	int
Bourough_Staten_Island	Whether the trip took place in Staten Island	int
Dependant variable name	Variable description	Variable type
trip_counts	Number of trip counts within 15 min (1 hr in case of dataset A)	float

project. On one hand, the available socio-demographic data is not at the taxi zone level. If the taxi zone is set as the target spatial aggregation level, the spatial features will have to be broken down and distributed among taxi zones, and a distribution method must be estimated, which may, in turn, introduce bias to the dataset. On the other hand, a finer aggregation level results in a much larger dataset, raising computational limits. Compared with the 5 boroughs, there are 263 taxi zones in New York City, and a spatial aggregation at the taxi zone level, and considering all the features will lead to an increase in training compute for an LSTM model with 8 input neurons and 100 epochs by approximately 10 orders of magnitude.

As mentioned before, apart from univariate trip records, socio-demographic, and climate features are also included. Figure 8 shows the spatial distribution of two example socio-demographic features: employment rate and mean commute time. One can observe the lower employment rates in less affluent boroughs such as Staten Island and the

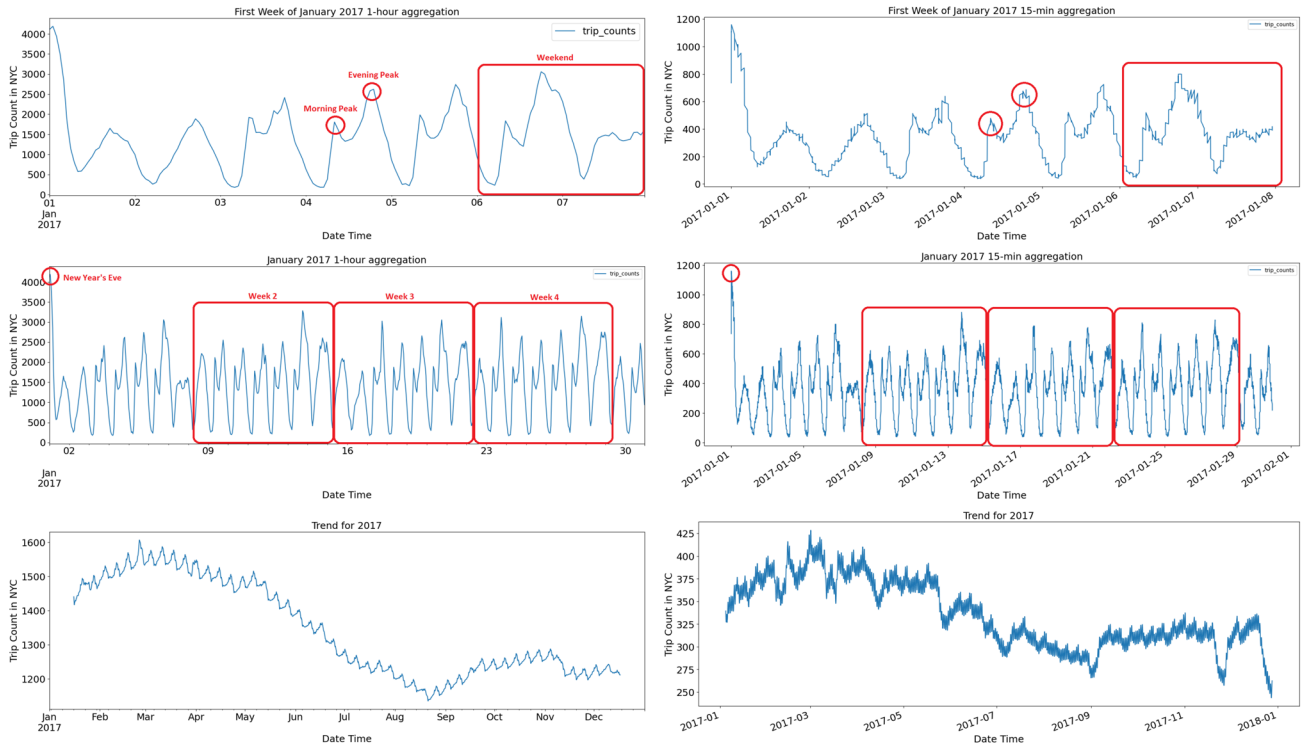


Fig. 7 Effect of temporal aggregation on trip count data (left: 1 h aggregation, right: 15 min aggregation, top row: trip count for the first week of January 2017, middle row: trip count for January 2017, bottom row: trip count for 2017)

Bronx. Lower employment rates can result in fewer commute trips and fewer trips in general. Conversely, mean commute times in less affluent boroughs are much higher than in more affluent boroughs like Manhattan. This suggests that more residents from the Bronx and Staten Island go to work in other, more affluent boroughs where there are more jobs and opportunities.

Furthermore, Fig. 9 shows two example climate features: daily minimum temperature and precipitation. The seasonality of both features is what we come to expect from climate data.

Dataset Design

For the purpose of investigating the effect of temporal aggregation and the inclusion of spatial features on green taxi demand prediction, three datasets were designed for cross-comparison. As seen in Fig. 7, dataset A is comprised of trip records and temporal features, and the data are temporally aggregated in one-hour intervals. Dataset B is designed to have the same trip records and similar temporal features, however, its data is temporally aggregated in 15-min intervals allowing for more fine-grained predictions in time. Finally, spatial features and climate features are integrated

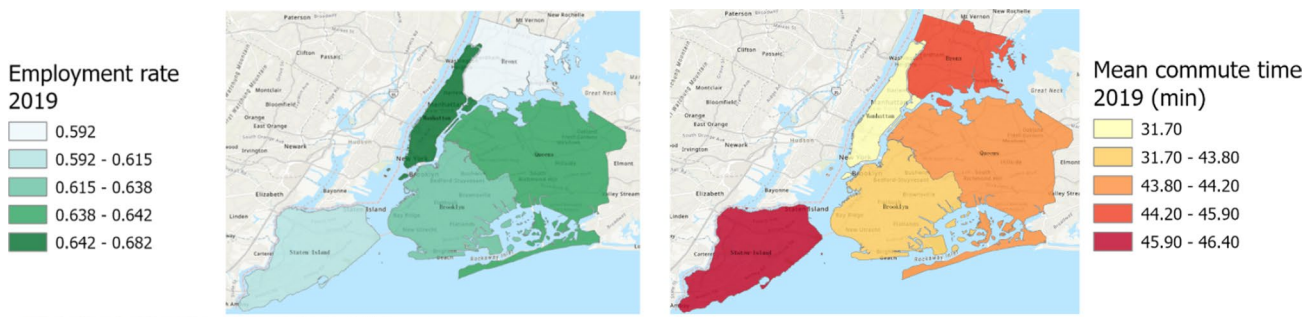


Fig. 8 Spatial distribution of employment rate (left) and mean commute time (right) in New York City in 2019 NYC (2019)

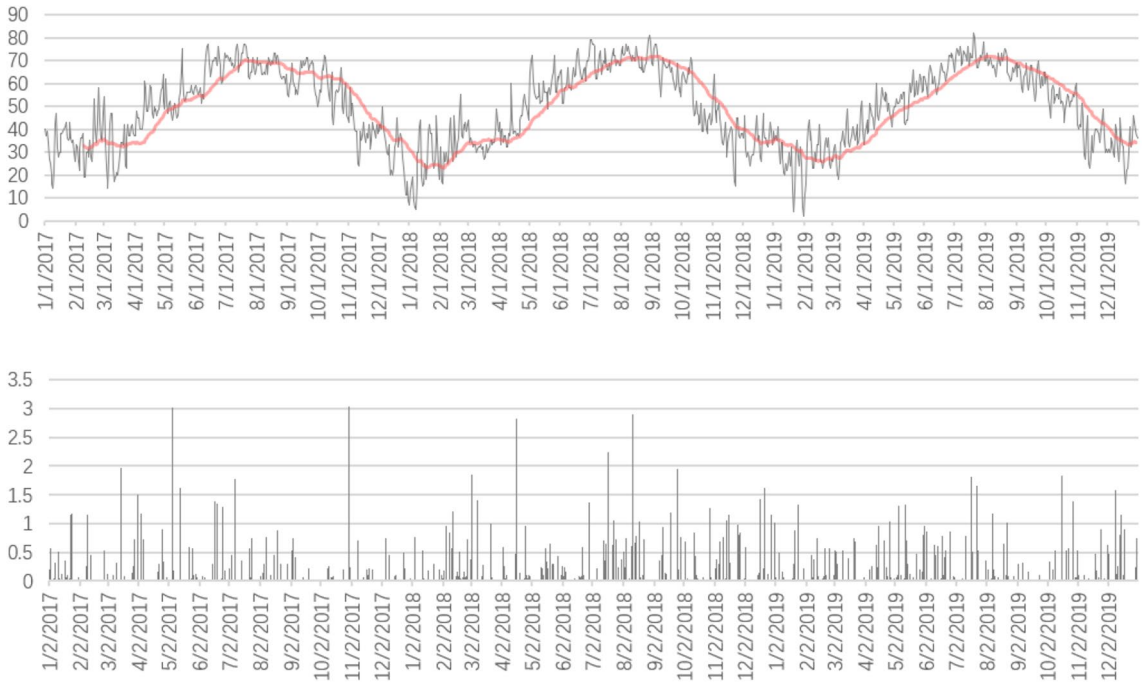


Fig. 9 Daily minimum temperature in Fahrenheit (top) and precipitation in inches (bottom)

into dataset B to create dataset C, which is a spatiotemporal dataset. It is spatially aggregated at the borough level and temporally aggregated in 15 min.

Data Preparation

The datasets are cleaned to remove any NA instances. Moreover, the data is scaled to avoid model bias towards features with larger values. To this end, the min–max scaler is employed to scale and standardize continuous data. The intention is to preserve the feature distribution while standardization, which is why data normalization was not used. After scaling, all the features are compressed to fit from 0 to 1, apart from the categorical data coded with one-hot coding. These categorical features include weekend and borough, which inform about whether the trip occurred on the weekend and in which borough the trip started.

Feature Selection

Feature selection is an important step that aims to reduce the computational cost of modeling and improve performance, by reducing the number of input variable dimensionality Vlahogianni et al. (2004). The objective here is to filter out redundant features that are linearly correlated to other features and are, therefore, not independent variables that add more information to the model. To achieve this, a Pearson correlation matrix is produced for dataset B and dataset C (dataset A has the same features as dataset B), illustrating the degree of linear collinearity between the two features. Collinearity of more than 0.5 indicates that the two features are correlated and a collinearity of 1 indicates a perfect correlation. Collinearity of -1 indicates a perfect inverse correlation and is also equally unwanted. Figure 11 shows the correlation matrices for datasets B and C.

Upon closer examination, it becomes clear that most socio-economic and climate features are correlated and should be filtered. The opted feature selection method is

Fig. 10 The three different datasets, dataset A and B, are temporal, and dataset C is a spatiotemporal dataset

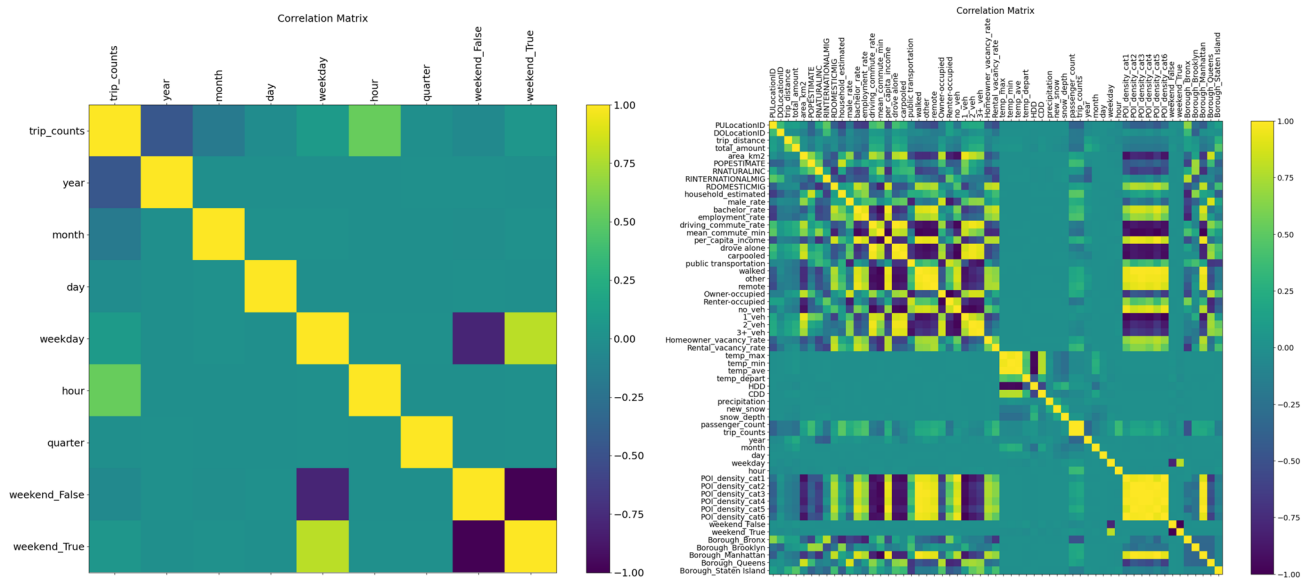


Fig. 11 Correlation matrices for dataset B with 9 features (left) and dataset C with 60 features (right)

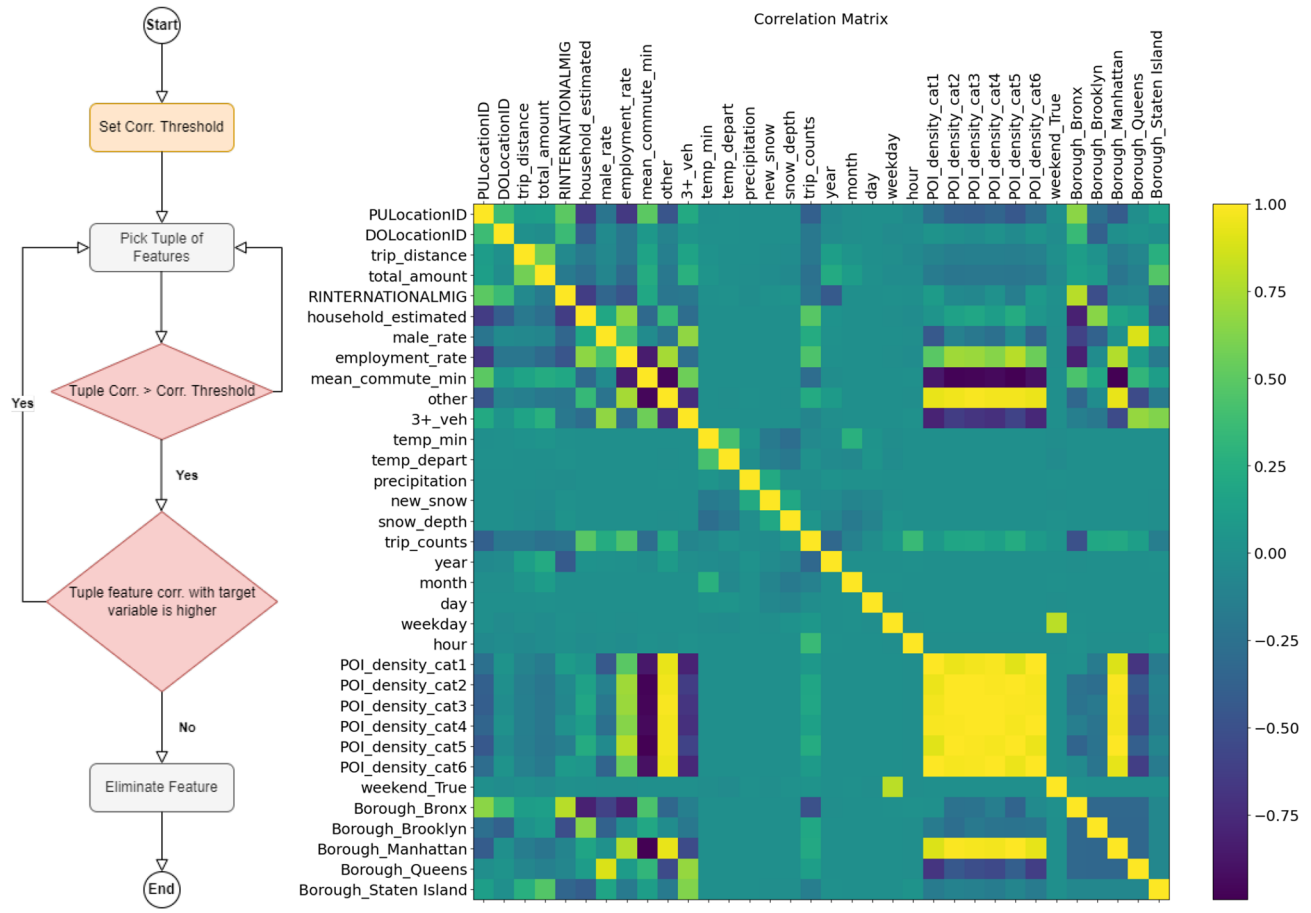


Fig. 12 Feature selection procedure flowchart (left), filtered dataset C with 34 features (right)

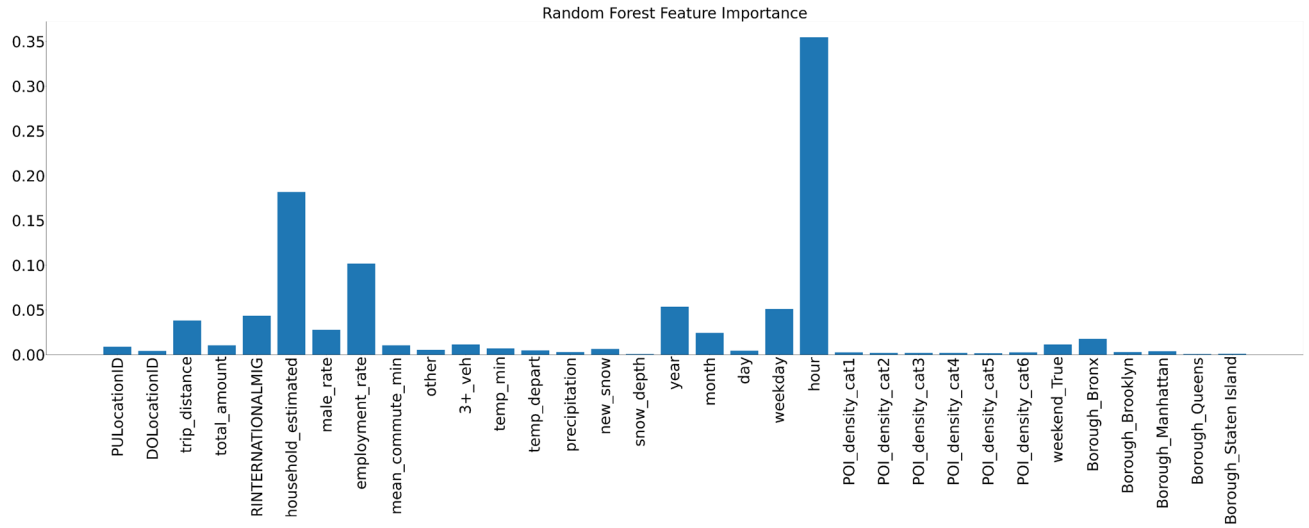


Fig. 13 Random Forest feature importance analysis of dataset C

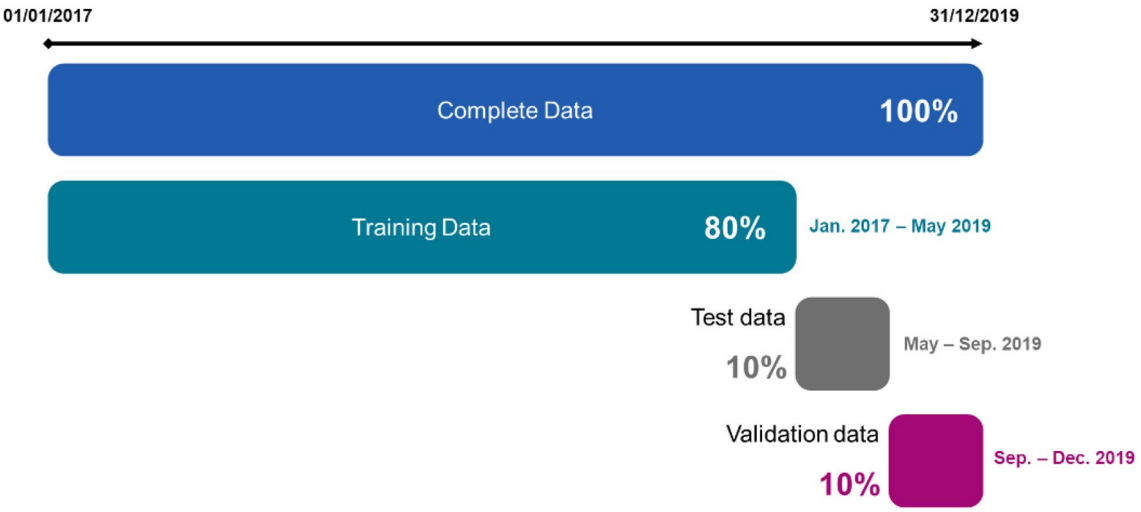


Fig. 14 Dataset Splitting

a procedural pairwise collinearity check that progressively filters out the redundant features with the least correlation to the target variable, in this case, trip counts. This procedure is depicted in Fig. 10 as a flowchart. Initially, we set a correlation threshold of 0.7. In other words, pairs of features with a correlation of more than 0.7 or less than −0.7 are selected for further inspection. After this step, the feature with the least correlation to the target variable is filtered and eliminated. As a result of feature selection, the number of features drops from 60 down to 34 (see Fig. 12).

It should be noted that adding one more feature was left out not directly due to this filtering procedure but to other reasons. This feature is passenger count with a linear correlation of 0.99 in relation to the target variable trip counts. This would suggest that most green taxi trips have only one passenger on board in New York City, which renders the problem of counting the number of trips within a given time interval almost indistinguishable from that of counting the number of passengers per trip in the same time window. This would mean that including the passenger count feature is tantamount to a tautological endeavor since one

would essentially try to predict trip counts with trip counts. Furthermore, the one-hot-coded spatial borough features are also excluded from the filtration process to allow for the observation of the regional differences in ride-hailing demand contribution.

Feature importance: as mentioned in Sect. 4 in **Random Forest Regressor**, one of the advantages of Random Forest model is to be interpretable and transparent, this attribute can provide a feature importance analysis to identify which features can significantly influence the prediction.

The feature importance analysis based on Random Forest 13 on dataset C shows that the most important variables are hours of the day, number of households in the departing borough, and employment rate in the departing borough. Moreover, the result suggests that few trips are from Staten Island, whereas the Bronx has the largest share of trip counts and therefore contributes more to the prediction outcome.

Data Transformation and Splitting

In preparing for the final data manipulation step, the data is sorted according to each incident's timestamp. Consequently, the data is split into training, testing, and validation sets in an 80–10–10% split as a general rule of thumb for machine learning, according to Fig. 14.

Importantly, since we are dealing with time-dependent data, it is essential that the splitting follows a chronological order so that earlier instances in time are used to predict instances in the future. The training data spans 29 months from January 2017 to May 2019 and the testing and validation data each span 5 and 4 months from May 2019 to September 2019 and from September 2019 to December 2019, respectively. The models are fit on the training sets and are optimized with the help of the validation data. Ultimately, the predictive performance of the fitted model on held-out testing set is assessed. As a further step, some of the machine learning models presented later in the following section, require the data to be transformed into a tensor with a certain shape. These models train on rolling timestep windows of a given size and extract certain patterns and features in the timestep window to predict the value immediately after the timestep window. Figure 15 shows such a rolling timestep window mechanism with a time-step window size of 8.

In this example, the models are trained on a time window of using 8 instances of independent feature vectors X_1 to X_8 to extract intrinsic patterns and features within the dataset in order to predict the target variable instance number 9 (y_9). In the next iteration, the time window is rolled over one-time unit, and independent feature vectors X_2 to X_9 are used to predict target variable instance y_{10} and so forth.

To train the models, we use 100 epochs and a learning rate of 0.0001. An epoch is a forward and backward pass of the dataset through the neural network, which is used to change the weights of the hidden layers, otherwise known as optimization. The learning rate is a tuning hyperparameter that determines the step size of the optimization iteration. The weights of the models are tuned through the minimization of the loss function, which in our case, is the mean squared error since this is a regression problem. Moreover, we use Adam as the optimizing algorithm because it is a stochastic algorithm that is more computationally efficient than gradient descent (Kingma and Ba 2017). Each model's testing and training root mean square error (RMSE) is used as a comparison metric since it can be interpreted as the accuracy of prediction in terms of the number of trip counts per unit of time.

Results

The results for all the models and architectures on the testing dataset are reported in Table 4. Moreover, the training times for dataset C for all the models are also reported. Root mean square error (RMSE) is the reported metric to compare the models defined as:

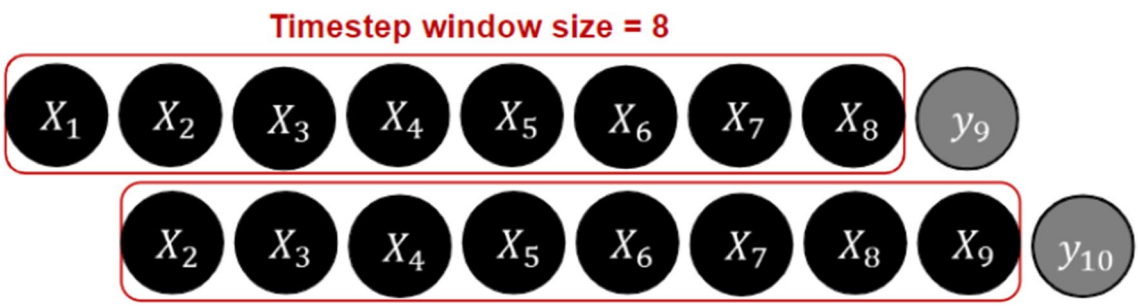


Fig. 15 Data shape transformation and rolling timestep window

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(\hat{x}_i - x_i)^2},$$

where x_i is the i^{th} Actual value and $\hat{x}_i$ predicted value of the demand, respectively, and n is the size of the test set.

Our experiment platform is a server with 8 CPU cores (Intel(R) Xeon(R) CPU @ 3.20 GHz), 256 GB RAM, with Python 3.9.16.

One of the main findings is that including spatial features can improve model performance. Compared with the best-performing model of Dataset B (LSTM Model 4), the best one for Dataset C (LSTM Model 4) has a 54.5% lower RMSE. Furthermore, all other models are improved by introducing spatial and climate features (see Table 4). Among all the models in our explorations, the best-performing models in each architecture are depicted in Fig. 16 for all three datasets. Moreover, the testing performance of the models with the best RMSE values for dataset C is presented in Fig. 17.

Results in Table 4 show that two-layer stacked LSTM architectures (M3–M5) performed better on average across the different datasets, and their performance got better with more data points. Model 4 of LSTM architecture with two layers of stacked LSTM learns the demand pattern and then passes it to a neural layer with eight nodes, which makes the interpretation. The last layer of one node produces the single output variable. The difference between model 4, model 3, and model 5 results stems from different hyperparameter settings. Model 1 and Model 2 of LSTM architecture have lower prediction accuracy on average. Interestingly, Model 6 does not outperform two-layer stacked architectures even though it is the deepest LSTM architecture. Furthermore, by stacking Time2Vec embedding with LSTM Model 4, we achieve lower RMSEs. This shows that the vector embedding of time series helps the algorithm to detect periodic behaviours in data in a more precise way.

The results for Random Forest show that RF-Model 1 and RF-Model 2 show signs of overfitting since training performance are orders of magnitude better than testing performance. In other words, the model is biased towards the training set, and it mimics its patterns to perfection, but as soon as it is faced with out-of-set data, it cannot generalize the learned patterns enough to be able to make decent predictions. Moreover, the performance curve of testing indicates that the models underestimate the demand at evening peaks. To address the overfitting issue, the tuned RF-Model 3 is designed, which is the best performing Random forest model out of 750 fits for dataset A. Although overfitting is mitigated, the model is not performing very well.

CNN models are much lighter than LSTM models and therefore run faster through training epochs. CNN Models have slightly lower but comparable accuracy levels compared to LSTM architectures. Interestingly, increasing the number of filters from 64 to 128 resulted in a worse accuracy

Table 4 Summary of RMSE of prediction models on green taxi test dataset

Model	Architecture	Dataset A Testing	Dataset B Testing	Dataset C Testing	Training time (min)
	Model 1	69.18	18.77	7.62	215
	Model 2	96.07	19.59	9.38	190
	Model 3	95.02	17.49	7.74	226
	Model 4	90.91	16.31	7.42	217
	Model 5	86.25	18.28	8.13	224
LSTM	Model 6	84.37	16.49	8.57	463
LSTM+Time2Vec	Model 4 + Time2Vec	77.27	13.86	6.80	220
	Model 1	76.12	18.81	8.52	42
CNN	Model 2	124.73	18.75	8.55	53
	Model 1	198.23	51.87	16.43	15
	Model 2	198.00	52.00	16.43	11
Random forest	Model 3	236.01	57.37	16.06	327
	Model 1	99.74	20.23	8.94	60
	Model 2	146.16	20.83	7.64	94
	Model 3	128.64	17.09	7.60	310
LSTM-CNN Autoencoder	Model 4	89.58	19.66	8.28	135
	Model 1	–	–	18.41	1500
	Model 2	–	–	19.97	2040
	Model 3	–	–	18.40	1992
	Model 4	–	–	18.37	1450
	Model 5	–	–	17.58	1413
Deep CNN	Model 6	–	–	17.48	1202

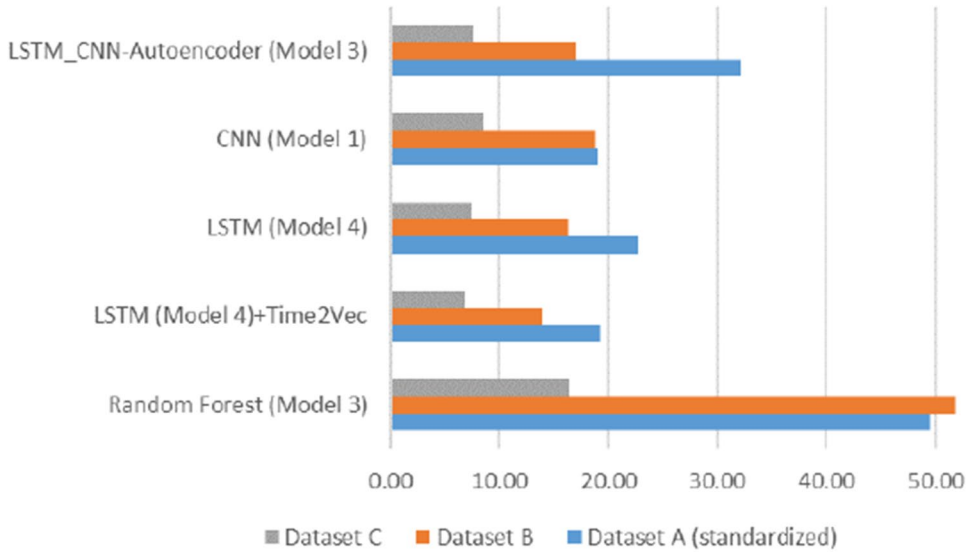
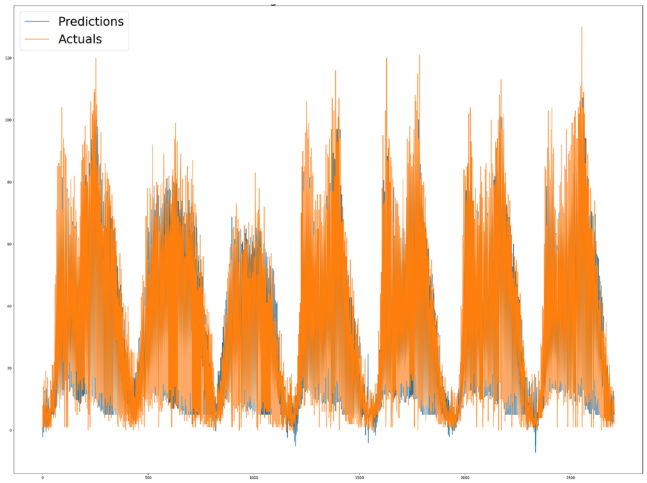
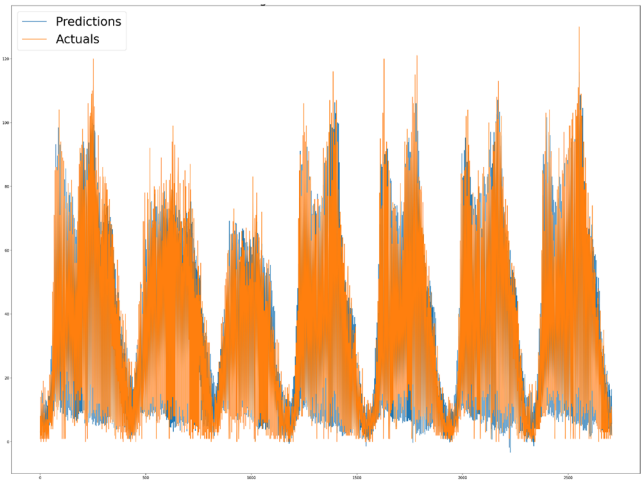


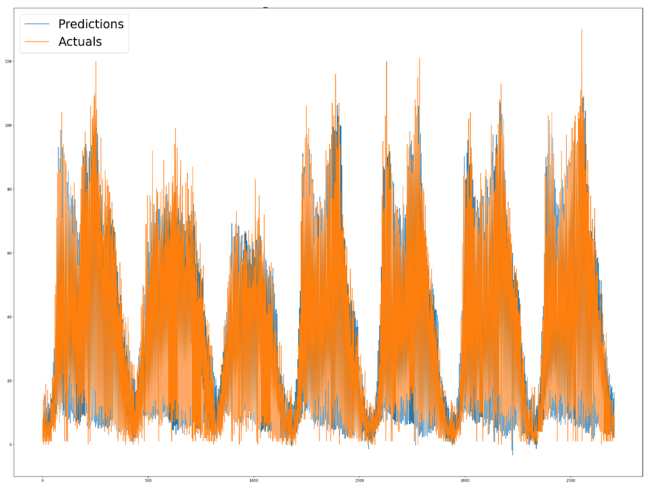
Fig. 16 RMSE comparison between best-performing architectures for green NYC taxi data



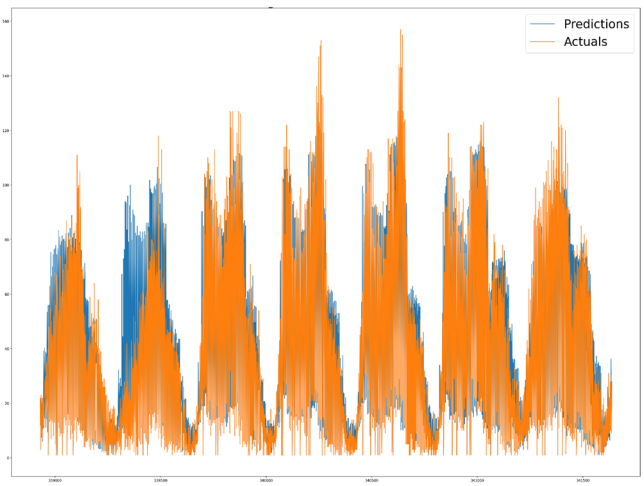
(a) CNN Model 1, RMSE=8.52



(b) LSTM-CNN Model 3, RMSE=7.60



(c) LSTM Model 4, RMSE=7.42



(d) RF Model 3, RMSE=16.43

Fig. 17 Testing performance of models with best RMSE values for dataset C for 1 week

Table 5 Standardized comparison between 1-h aggregation (dataset A) and 15-minute aggregation (dataset B)

Model	Architecture	Dataset A (standardized) Testing	Dataset B Testing	Performance improvement
LSTM	Model 4	22.72	16.31	28.21%
CNN	Model 1	19.03	18.81	1.15%
Random Forest	Model 1	49.55	51.87	−4.47%
LSTM-CNN Autoencoder	Model 3	32.16	17.09	46.85%

across all datasets, indicating that more complicated feature maps do not necessarily output more accurate results. The LSTM-CNN Autoencoder architecture performs better than simple CNN, but it runs slower. However, it is more computationally efficient than LSTM architectures.

In deep CNN architecture, Model 2, with more convolutional layers, results in better model performance compared to model 1. It is interesting to observe that increasing the convolutional filters and thereby increasing the number of feature maps only has a constructive effect in deeper models, which might suggest that increasing filters in lighter models leads to overfitting and lower performance. The performance of Model 3 with another fully connected layer with 32 nodes which is close to the independent feature count of 28, dropped compared to Models 1 and 2. Model 4, with an additional layer of batch normalization following each convolutional layer, results in faster models (fewer epochs are required for training). All things constant, model 4 indeed trained faster compared to model 3. Models 5 and 6 make use of a 20% dropout in the second and second to third max-pooling layers, respectively, which contributes to better results compared to other deep CNN models.

The different temporal aggregation has different impacts on model performance. By comparing the results of dataset A and dataset B shown in Table 4, RMSEs for dataset B are approximately 75% lower than A. This is because the RMSE for dataset A indicates the error within 1 h, while in dataset B, this error is for 15 min. Therefore, a comparison should be based on standardized values as presented in Table 5. The results indicate that a finer temporal aggregation improve models involving LSTM layers, as well as LSTM-CNN Autoencoder architectures, while CNN and random forest are not sensitive to the change of aggregation level. Consequently, the aggregation approach should be carefully selected for LSTM modeling.

Conclusions

In this work, we explored the task of predicting the demand for on-demand services with different deep-learning approaches. By conducting a cross-comparison between different architectures, we showed how different layers can affect the prediction performance. By vector embedding the time series we improved the results. Moreover, by examining the temporal aggregation levels, we showed how different architectures are affected by the level of aggregation.

Improved datasets and modeling approaches should be included to develop the work further. The explored models can still be improved. Currently, all models except the random forest models are tuned manually. To further improve the performance, the models need to be fine-tuned through robust grid searches. Moreover, spatial features at higher resolution levels can realize prediction at the taxi zone level, which is more helpful to the taxi service providers. Then, Temporal features like real-time traffic load, congestion sites, and accidents can also be included. Finally, the models can be applied to other transportation modes like private cars and buses. To acquire these data, extensive data collection and fusion are needed. On the other hand, advanced models, and architectures such as customized CNN layers and multi-step architectures can be employed to improve model performance further.

It is worth mentioning that many studies on this subject focus solely on the location and time of demand as the only input to their algorithms. In our methodology, we aimed to incorporate all publicly available features to explore their impact on the prediction. A comprehensive list of all the features we have included is provided in Table 3. To the best of our knowledge, no other studies have examined such a wide range of features and their influence on the prediction. The inclusion of all these features necessitates working at a certain level of data aggregation provided by the publicly available feature data source. Consequently, obtaining fine-grained feature data for smaller zones was not feasible for us. Considering these reasons, we selected methods that could accommodate all the data and features we intended to work with. To compare our results with other works, we refer to the work by Chen et al. (2021). The results provided in Table 2 and Table 5 of their manuscript (Chen et al. 2021) show that the RMSE values for their predictions are worse than our results, and the number of features we have considered is greater.

Author Contributions ZG: Conceptualization, methodology, data curation, formal analysis, software, writing—original draft preparation,

visualization, investigation, supervision, validation, writing—reviewing and editing. PZN: conceptualization, methodology, software, formal analysis, visualization, investigation, validation, writing—reviewing and Editing. WW: conceptualization, methodology, software, formal analysis, visualization, investigation, validation, writing—reviewing and editing. All authors reviewed the manuscript.

Funding Open access funding provided by Swiss Federal Institute of Technology Zurich. No funding was obtained for this study.

Availability of Data and Materials Raw data used in this study are available at NYC (2019) https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page and [36] https://data.cityofnewyork.us/City-Government/Points-Of-Interest/rxuy-2muj. The rest of the materials are not publicly available due to further development of the research project.

Declarations

Conflict of interest There is no financial or non-financial conflict of interest for the current work.

References

Breiman L (2001) Random forests. Mach Learn 45:5–32

Chen MK (2016) Dynamic pricing in a labor market: surge pricing and flexible work on the Uber platform. In: Proceedings of the 2016 ACM conference on economics and computation. EC '16. Association for Computing Machinery, New York, NY, USA, p 455. https://doi.org/10.1145/2940716.2940798

Chen L, Thakuriah PV, Ampountolas K (2021) Short-term prediction of demand for ride-hailing services: a deep learning approach. J Big Data Anal Transp 3(2):175–195. https://doi.org/10.1007/s42421-021-00041-4

Cheng Q, Liu Y, Wei W, Liu Z (2016) Analysis and forecasting of the day-to-day travel demand variations for large-scale transportation networks: a deep learning approach. In: TRB 2017 transportation analytics contest. https://doi.org/10.13140/RG.2.2.12753.53604

Chen L, Mislove A, Wilson C (2015) Peeking beneath the hood of Uber. In: Proceedings of the 2015 internet measurement conference. Association for Computing Machinery, New York, pp 495–508. https://doi.org/10.1145/2815675.2815681

City of New York Department of Information Technology & Telecommunications (2023) City of New York department of information technology & telecommunications: point of interest (CommonPlace). https://data.cityofnewyork.us/City-Government/Points-Of-Interest/rxuy-2muj. Accessed 1 Feb 2023

Dandl F, Hyland M, Bogenberger K, Mahmassani HS (2019) Evaluating the impact of spatio-temporal demand forecast aggregation on the operational performance of shared autonomous mobility fleets. Transportation 46(6):1975–1996. https://doi.org/10.1007/s11116-019-10007-

Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv. https://doi.org/10.48550/arXiv.1412.6980

Geng X, Li Y, Wang L, Zhang L, Yang Q, Ye J, Liu Y (2019) Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. Proc AAAI Conf Artif Intell 33(01):3656–3663. https://doi.org/10.1609/aaai.v33i01.33013656

Guo J, Huang W, Williams BM (2014) Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. Transp Res C Emerg Technol 43:50–64. https://doi.org/10.1016/j.trc.2014.02.006. (**Special Issue on Short-term Traffic Flow Forecasting**)

Huang W, Song G, Hong H, Xie K (2014) Deep architecture for traffic flow prediction: Deep belief networks with multitask learning. IEEE Trans Intell Transp Syst 15:2191–2201

Iglesias R, Rossi F, Wang K, Hallac D, Leskovec J, Pavone M (2018) Data-driven model predictive control of autonomous mobility-on-demand systems. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE Press, Brisbane. p 1–7. https://doi.org/10.1109/ICRA.2018.8460966

Kazemi SM, Goel R, Eghbali S, Ramanan J, Sahota J, Thakur S, Wu S, Smyth C, Poupart P, Brubaker M (2019) Time2Vec: learning a vector representation of time. arXiv . https://doi.org/10.48550/ARXIV.1907.05321

Ke J, Zheng H, Yang H, Chen XM (2017) Short-term forecasting of passenger demand under on-demand ride services: a spatio-temporal deep learning approach. Transp Res C Emerg Technol 85:591–608. https://doi.org/10.1016/j.trc.2017.10.016

Ke J, Feng S, Zhu Z, Yang H, Ye J (2021) Joint predictions of multi-modal ride-hailing demands: A deep multi-task multi-graph learning-based approach. Transp Res C Emerg Technol 127:103063. https://doi.org/10.1016/j.trc.2021.103063

Li S, Tavafoghi H, Poolla K, Varaiya PP (2019) Regulating TNCs: should Uber and Lyft set their own rules? Transp Res B Methodol. https://doi.org/10.1016/j.trb.2019.09.008

Lv Y, Duan Y, Kang W, Li Z, Wang F-Y (2015) Traffic flow prediction with big data: a deep learning approach. IEEE Trans Intell Transp Syst 16(2):865–873. https://doi.org/10.1109/TITS.2014.2345663

Ma X, Yu H, Wang Y, Wang Y (2015) Large-scale transportation network congestion evolution prediction using deep learning theory. PLoS One 10(3):1–17. https://doi.org/10.1371/journal.pone.0119044

Moreira-Matias L, Gama J, Ferreira M, Mendes-Moreira J, Damas L (2013) Predicting taxi-passenger demand using streaming data. IEEE Trans Intell Transp Syst 14(3):1393–1402. https://doi.org/10.1109/TITS.2013.2262376

NYC (2019) NYC taxi and limousine commission: NYC TLC trip record data. https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page. Accessed 1 July 2019

NYC Open Data (2023) Point of Interest (CommonPlace). City of New York Department of Information Technology & Telecommunications. https://data.cityofnewyork.us/City-Government/Points-Of-Interest/rxuy-2muj. Accessed 1 Feb 2023

Sayarshad HR, Chow JYJ (2017) Non-myopic relocation of idle mobility-on-demand vehicles as a dynamic location-allocation-queueing problem. Transp Res E Logist Transp Rev 106(C):60–77. https://doi.org/10.1016/j.tre.2017.08.003

Tan H, Xuan X, Wu Y, Zhong Z, Ran B (2016) A comparison of traffic flow prediction methods based on DBN. In: CICTP 2016. p 273–283. https://doi.org/10.1061/9780784479896.026

Vlahogianni EI, Golias JC, Karlaftis MG (2004) Short-term traffic forecasting: overview of objectives and methods. Transp Rev 24(5):533–557. https://doi.org/10.1080/0144164042000195072

Wang H, Yang H (2019) Ridesourcing systems: a framework and review. Transp Res B Methodol 129:122–155. https://doi.org/10.1016/j.trb.2019.07.009

Williams BM, Hoel LA (2003) Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: theoretical basis and empirical results. J Transp Eng 129(6):664–672. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664)

Xu J, Rahmatizadeh R, Bölöni L, Turgut D (2018) Real-time prediction of taxi demand using recurrent neural networks. IEEE Trans Intell Transp Syst 19(8):2572–2581. https://doi.org/10.1109/TITS.2017.2755684

Yang H, Yang T (2011) Equilibrium properties of taxi markets with search frictions. Transp Res B Methodol 45(4):696–713. https://doi.org/10.1016/j.trb.2011.01.002

Yang H, Wong SC, Wong KI (2002) Demand-supply equilibrium of taxi services in a network under competition and regulation. Transp Res B Methodol 36(9):799–819. https://doi.org/10.1016/S0191-2615(01)00031-5

Yang H, Leung CWY, Wong SC, Bell MGH (2010) Equilibria of bilateral taxi-customer searching and meeting on networks. Transp Res B Methodol 44(8):1067–1083. https://doi.org/10.1016/j.trb.2009.12.010

Yang S, Ma W, Pi X, Qian S (2019) A deep learning approach to real-time parking occupancy prediction in spatio-temporal networks incorporating multiple spatio-temporal data sources. arXiv . https://doi.org/10.48550/ARXIV.1901.06758

Yu X, Gao S, Hu X, Park H (2019) A Markov decision process approach to vacant taxi routing with e-hailing. Transp Res B Methodol 121(C):114–134. https://doi.org/10.1016/j.trb.2018.12.013

Zardini G, Lanzetti N, Pavone M, Frazzoli E (2021) Analysis and control of autonomous mobility-on-demand systems. Annu Rev Control Robot Auton Syst. https://doi.org/10.1146/annurev-control-042920-012811

Zhang Y, Zhang Y (2018) Exploring the relationship between ridesharing and public transit use in the united states. Int J Environ Res Public Health. https://doi.org/10.3390/ijerph15081763

Zhang D, He T, Lin S, Munir S, Stankovic JA (2017) Taxi-passenger-demand modeling based on big data from a roving sensor network. IEEE Trans Big Data 3(3):362–374

Zhang J, Zheng Y, Qi D, Li R, Yi X (2016) DNN-based prediction model for spatio-temporal data. In: Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems. SIGSPACIAL '16. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2996913.2997016

Zhou X, Shen Y, Zhu Y, Huang L (2018) Predicting multi-step citywide passenger demands using attention-based neural networks. In: Proceedings of the eleventh ACM international conference on web search and data mining. Association for Computing Machinery, New York, pp 736–744. https://doi.org/10.1145/3159652.3159682

Zoepf S, Chen S, Adu P, Pozo G (2018) The economics of ride hailing: driver revenue, expenses and taxes