School of Computer Science and Engineering

Faculty of Engineering

The University of New South Wales

Taxi Demand Analytics and Prediction

by

Yao Yuan

Thesis submitted as a requirement for the degree of

Bachelor of Engineering in Computer Engineering

Submitted:	28 November 2019	Student ID:	z5092195
Supervisor:	Dr.Wei Liu	Topic ID:	

Abstract

Taxi demand reflects the travel behavior preference of people in different locations at different times. Predicting taxi demand can help taxi companies to organize taxi fleet and also to help passengers and drivers to reduce waiting time. In the past, many researchers have used statistical methods to forecast taxi demands. In recent years, deep learning approaches have been demonstrated to learn various features and to explore the patterns behind large sets of data, which provides the new opportunity to learn the taxi demand pattern efficiently and effectively.

This thesis project will utilize the Long-Short Term Memory model, which is popular in deep learning, to find the pattern behind the taxi demand in New York City and then to predict the taxi demand using this pattern. The project will also predict the taxi demand using another regression model XGBoost model, which is powerful in machine learning. Various evaluation methods will be conducted on these two models, and the results will be compared from different perspectives. Finally, observations from them will be concluded.

Acknowledgements

The author would like to thank his supervisor, Dr. Wei Liu and accessor, Dr. Lina Yao for giving crucial and helpful guidance in the formation, development, and shaping of this thesis.

Abbreviations

LSTM Long-Short Term Memory

XGBoost Extreme Gradient Boosting

RMSE Root Mean Square Error

MAE Mean Absolute Error

MPE Mean Percentage Error

Contents

List of Figures

Chapter 1

Introduction

Taxis have become an ideal traveling vehicle for people since 1897; although, nowadays, there are many new traveling options raised, like Uber driving and GoGet car-sharing, taking a taxi is still one of the most popular traveling methods in daily life.

In recent years, it is not hard to find that a taxi driver hangs around in a location and cannot find passengers. However, at the same time, there might be several people waiting for taxis in a different location; and this is a problem that bothers both taxi drivers and passengers. Therefore, in order to solve this problem and have an effective taxi dispatching, taxi demand needs to be predicted. For example, if the taxi company can predict the taxi demand in different locations at different time intervals during a day, they can organize their taxi fleet and distribute them to those locations to pick up the passengers, and this will primarily reduce the waiting time for both taxi drivers and passengers and lead into a 'Win-Win' condition.

Taxi demand prediction is challenging because many factors are related to or even affect it. For example, weather and temperature are two factors that can affect the taxi demand because people are more likely to take a taxi when they meet the day with extremely high or low temperatures or heavy raining, which has been demonstrated by Kamga [KYS13]. Location can also be a factor that is related to taxi demand; because the center of a city is always the place that has a higher taxi pickup density compared

with the countrysides, as most workplaces and entertainment shopping centers are located at there [Sha18].

In 2016, Kai Zhao and his team proposed and implemented three predictors to forecast the taxi demand at high spatial resolution [ZKF+16]. They are the Markov predictor, the Lempel-Ziv-Welch predictor, and the Neural Network predictor. They found that the performance of these three predictors varies, and on average, the theoretical maximum predictability can be as high as 83%. Kai Zhang, Zhiyong Feng, and other researchers proposed a framework for passenger demand prediction and recommendation [ZFC+16]. Their study is based on spatiotemporal clustering, and they proposed a demand hotspot prediction framework to generate recommendations to tell taxi drivers where to pick up the passengers. Many other studies have explored how to predict the taxi demand, and they will be discussed in Chatper 2.

This thesis project aims to predict taxi demand in New York City. It will utilize a recurrent neural network model and a regression model to forecast the demand and compare the performance between them. The rest of the report will be organized as follows. Chapter 2 explains the background knowledge and related work for this project. Chapter 3 states the solutions and implementation of this project. In Chapter 4, results will be compared and analyzed. The last chapter, Chapter 5, will conclude the observations from this project and future works will be mentioned.

Chapter 2

Background

In this chapter, background knowledge will be explained, and some related works will be explored. From some literal reviews, knowledge gaps will be pointed out and how this thesis project will fill that.

2.1 Data

2.1.1 Data clustering

Clustering is the task of grouping a set of objects in a certain way that objects in the same cluster are more similar to each other in certain features than to those in other clusters [DG06]. Several studies on taxi demand prediction have shown that data clustering is one method that can improve the accuracy of the forecast. The reason for this improvement is that clustering can find the hidden features behind the data and then dividing data into different groups so that models can learn these data better, which will eventually improve the results. Davis and his team explored how to use a multi-level clustering approach to improve their accuracy on forecasting taxi demand [DRJ16]. They found that their results can achieve a 20% improvement on the linear time-series model fitting by utilizing the clustering method.

K-means clustering method will be considered in this project if improvements are needed in the later stage of the project.

K-Means Clustering

K-means Clustering is an approach that automatically partitions a data set into K disjoint subsets. Figure 2.1 shows the pseudo-code for this algorithm. The goal of K-means Clustering is to maximize the homogeneity within each cluster by minimizing the square-error between each point and the cluster center[HW79, EAM06]. The formula for the square error is [EAM06]:

$$E = \sum_{i=1}^{K} \sum_{j=1}^{n} |dist(x_j, c_i)|^2 \tag{2.1}$$

K-means Clustering algorithm has many advantages; for example, it is easy to implement, and the results it presents are easy to understand. However, it also has many drawbacks. The parameter K cannot be determined by the K-means Clustering algorithm itself, which means it needs to be defined before the algorithm starts[HW79, KMN+02]. It is an NP-hard problem to choose an optimal K value[Vat09]. Besides, Outliers in a data set can dramatically influence the results of the K-mean Clustering [CTC+08]; but good data preprocessing will eliminate this problem.

Input: *k* (the number of clusters),
D (a set of lift ratios)
Output: a set of k clusters
Method:
Arbitrarily choose *k* objects from *D* as the initial cluster centers;
Repeat:
1. (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
2. Update the cluster means, i.e., calculate the mean value of the objects for each cluster
Until no change;

Figure 2.1: Pseudo Code for K-means clustering [ZC14]

2.2 Model

In this section, the Long-Short Term Memory model and Extreme Gradient Boosting model will be introduced. Background knowledge of each model will be briefly explained.

2.2.1 Long-Short Term Memory model

Long-Short Term Memory model is an artificial Recurrent Neural Network model which also refers to the LSTM model. It is a model that is capable of learning long-term dependencies by utilizing some gating mechanisms to store information [HS97, GSC99]. It is unlike feed-forward neural networks that only predict the outputs based on the current inputs; since LSTM contains memories, it can use the information that stored in its memories to predict the outputs together with current inputs [KJFF15]. Figure 2.2 is a graph that depicts one LSTM cell.

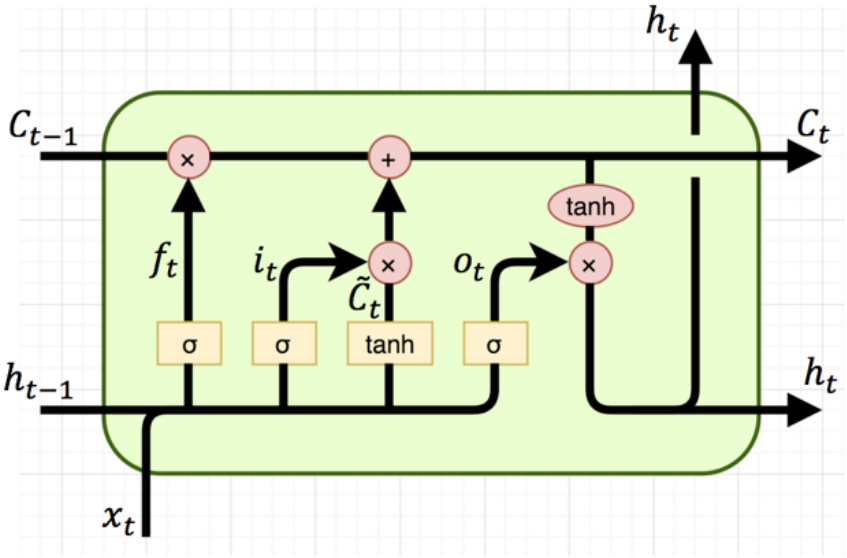


Figure 2.2: LSTM Cell [Hao17]

The mathematical equations to each value in the cell are shown as following:

$$i_t = \sigma(x_t U_i + h_{t-1} W_i) \tag{2.2}$$

$$f_t = \sigma(x_t U_f + h_{t-1} W_f) \tag{2.3}$$

$$o_t = \sigma(x_t U_o + h_{t-1} W_o) \tag{2.4}$$

$$\hat{C}_t = tanh(x_t U_g + h_{t-1} W_g) \tag{2.5}$$

$$C_t = \sigma(f_t * Ct - 1 + i_t * \hat{C}_t) \tag{2.6}$$

$$h_t = tanh(C_t) * o_t \tag{2.7}$$

W is the recurrent connection between the previous hidden layer and current hidden layer; U is the weight matrix connecting the inputs to the current hidden layer. The input gate (i) defines how much of the newly computed state for the current input will be passed through (2.2). The forget gate (f) defines how much of the previous state will be forgotten that will not be passed to the current state (2.3). The output gate (o) defines how much of the internal state will be exposed to the external network, which is the higher layers, and the next time step (2.4). C with a hat on it represents the hidden state that is calculated base on the current input and the previous hidden state (2.5). C without a hat stands for the new internal memory of the unit, which can also be interpreted as a combination of previous memory and the new input (2.6). * in equation (2.6) and (2.7) represents the element-wise multiplication.

In this project, since taxi demand prediction is a spatial-temporal series problem, an intelligent sequence analysis model is needed. Therefore, the LSTM model is a suitable model for it. Some studies have already used LSTM models to forecast the taxi demand, and they will be discussed in the section of 'Related Works'.

2.2.2 Extreme Gradient Boosting Model

Extreme Gradient Boosting, which is also called XGBoost, is an implementation of gradient boosted decision trees that are aiming to faster the speed and improve the performance. It is an implementation created by Tianqi Chen, now with many contributions from other developers [CG16]. XGBoost is an advanced gradient boosting

algorithm. It is powerful to deal with all kinds of irregular data because this algorithm supports many features. It is sparse aware, which means it can automatically handle the missing data values; its block structure supports the parallel tree construction, and it can further boost an already fitted model by continuous training on new data [Bro16, CG16]. Figure 2.3 shows the areas that the XGBoost model optimizes the normal Gradient Boosting Machines. Therefore, this is a suitable model for taxi demand prediction.

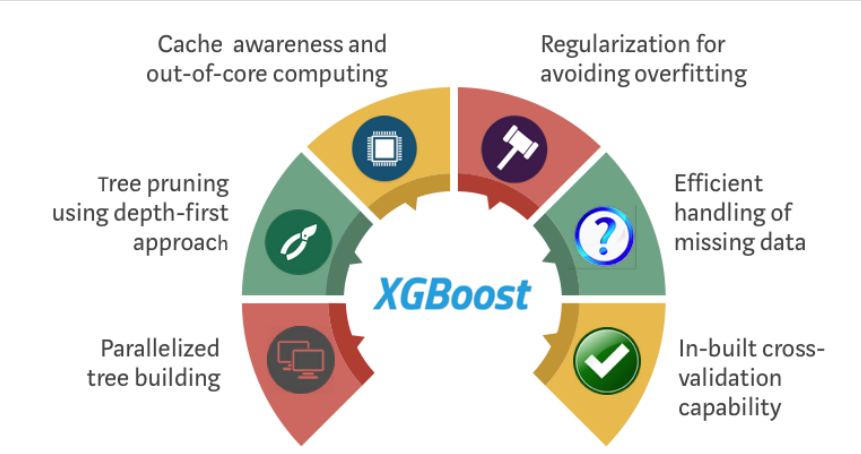


Figure 2.3: XGBoost Advantages graph [Mor19]

2.3 Evaluation

In this subsection, two evaluation methods will be introduced and will be used later in the project. They are Symmetric Mean Absolute Percentage Error and Root Mean Square Error.

2.3.1 Mean Absolute Error

Mean absolute error (MAE) is a method that evaluates the result of a prediction based on mean values. There are many versions of MAE; this project is going to use the

version that is often used in practice. The formula for this is:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|F_t - A_t| \tag{2.8}$$

Where A is the actual value, and F is the forecast value. This error generally shows the difference between the actual values and predicted values in the unit of the number of pickups.

2.3.2 Mean Percentage Error

Mean Percentage Error (MPE) is the error based on the percentage values. This error method can give an idea of how accurate the predicted values are in all predicted locations. It is the value of the sum of the absolute difference between prediction and real values over the sum of real values.

$$MPE = \frac{\sum_{i=1}^{n}|f_i - o_i|}{\sum_{i=1}^{n} o_i} \tag{2.9}$$

In equation (2.10), parameter f represents forecast values; o stands for observed values, which can also be considered as real values, and n is the sample size.

2.3.3 Root Mean Square Error

Root Mean Square Error (RMSE) is the standard deviation of prediction errors (residuals). Prediction errors are a measure of how far from the regression line where target data points are; In other words, RMSE tells how concentrated the predicted data is around the line of best fit. It is common to apply RMSE to forecasting problems to verify experimental results. Figure 2.4 shows a sample plot of RMSE. The mathematical formula for RMSE is [Mon18]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_i - o_i)^2} \tag{2.10}$$

In equation (2.10), parameter f represents forecast values; o stands for observed values which can also be considered as real values and n is the sample size.

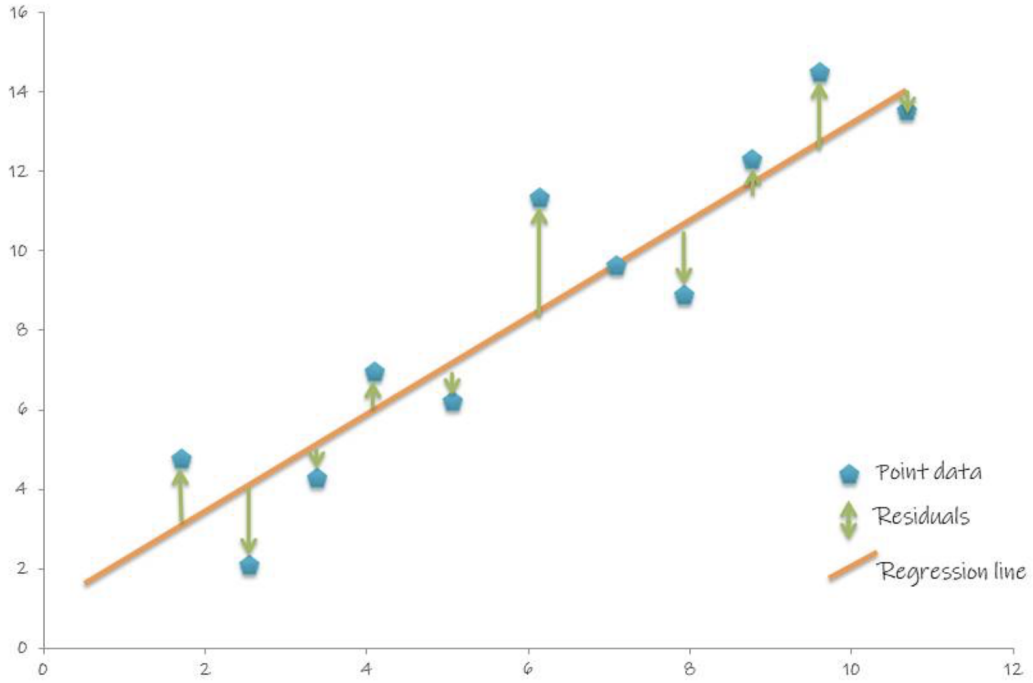


Figure 2.4: A sample plot of RMSE [Mon18]

2.4 Related Works

In this section, some related works and studies on taxi demand prediction will be discussed. The "knowledge gap" will be pointed out and how this project is different from them.

2.4.1 Related Work: using Recurrent Neural Network

As the recurrent neural network becoming more and more popular in data prediction, some studies have explored to forecast the taxi demand using this network. Jun Xu and his team did one typical study; they proposed a real-time method for predicting taxi demand and explained their work in the paper "Real-Time Prediction for Taxi Demand Using Recurrent Neural Network" [XRBT18]. In their study, they also used the LSTM model to train their data. Figure 2.5 shows the learning model they used. Since they

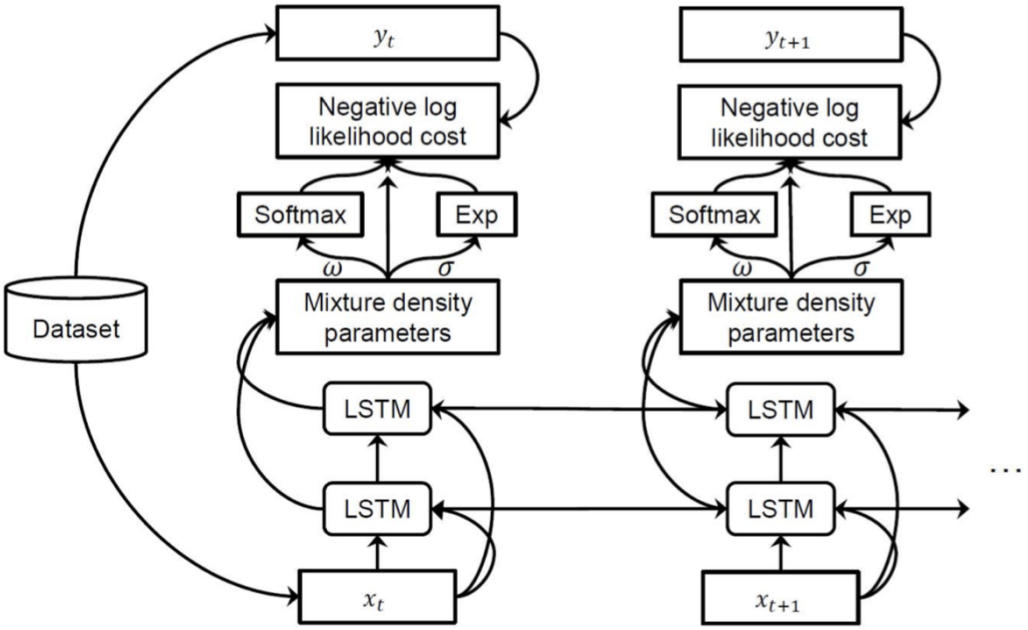


Figure 2.5: The LSTM-MDN learning model unrolled through time-steps [XRBT18]

did not consider other factors like weather, events; and they knew this would cause some uncertainties in their model, instead of predicting a deterministic value, they chose to predict the entire probability distribution of taxi demands in all areas. Therefore, they introduced Mixed Density Network(MDN) to predict probability, which is indicated as Mixture density parameters in Figure 2.5. This study gives a direction on how to predict taxi demand using a recurrent neural network, and many methods can be learned and applied to the project.

However, there are many differences between this study and this project. Firstly, in their study, they also use the taxi data in New York City; but instead of doing data clustering, they merely divided the whole city into small areas using the Geohash library, which can divide a geographical area into smaller subareas with arbitrary precision. Besides, as mentioned above, they predicted the probability distribution of taxi demand; but in the case of this project, deterministic results will be forecast. The last but not the least, in their model, they did not consider many underlying factors that can affect the taxi demand; however, in this project, some factors, like weather and events, will

be considered.

2.4.2 Related Work: predicting in Event Areas

As mentioned in the previous section, this project is going to predict the taxi demand with other factors into consideration. A study aims at exploring deep learning architectures for combining time-series and textual data for mobility demand prediction in eventful areas is done by Filipe Rodrigues and his partners [RMP19]. In their study, they used the neural network architectures, which combine word embeddings, convolutional layers, and an attention mechanism. The data set they used can be divided into two categories, the Taxi data set and the Textual data set. They also chose to use New York City taxi data set; however, they only used the data with pickups happened in a bounding box of 0.003 decimal degrees (roughly 500 m) in the Barclays Center and Terminal 5; because these two places are typical event areas in New York City. Besides, they also removed the regular recurring trend in taxi demand based on the historical data; because their goal was to find the taxi demand caused by the events. For the textual data set, they firstly did data mining to retrieve some textual data from websites like Facebook, Twitter, and also the official ticket website for the Barclays Center and Terminal 5; and then filter and process the raw textual data into a word array. To pass this word array into the neural network, they encoded it into a numeric vector, and each number in the input vector represents a word. In their study, they proposed two models to forecast the taxi demand and compared their performance. The first model is DL-LSTM, as shown in Figure 2.6. DL stands for the dense layer, and since it uses an LSTM to encode the time-series data, this model is called DL-LSTM. The second model is DL-FC, as shown in Figure 2.7. FC represents the fully connected network in the dense layer.

This study guides a way of how to predict taxi demand with other factors considered in the recurrent neural network model, and the factors it considered are weather and events. However, this project will be different from it. Firstly, all areas in New York City will be considered instead of only two event areas. Secondly, it will predict the

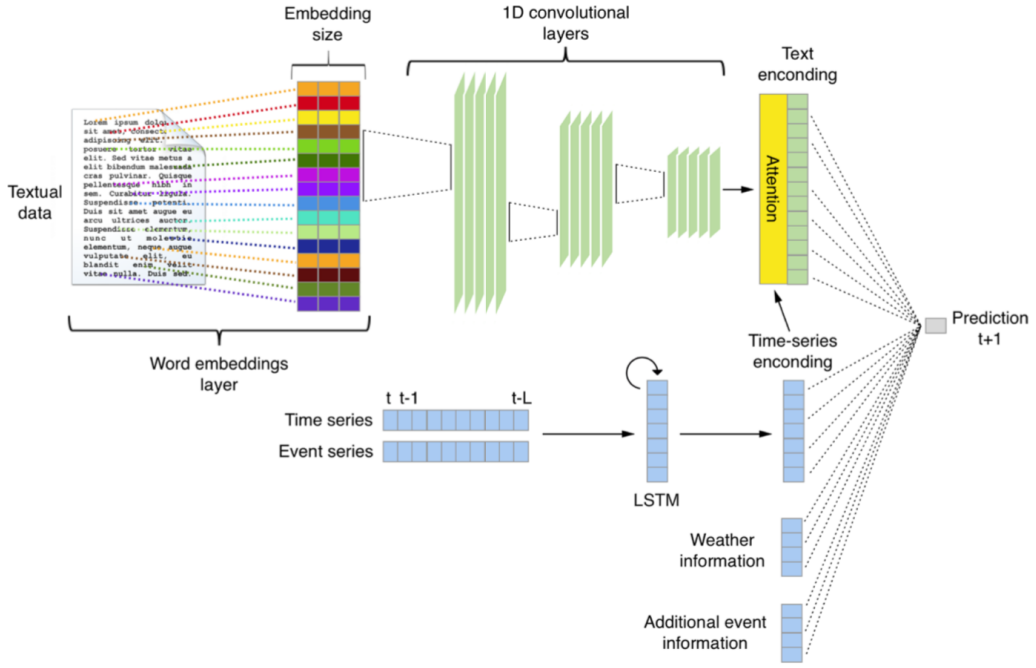


Figure 2.6: Proposed neural network architecture with an LSTM layer for modelling the time-series observations; DL-LSTM [RMP19]

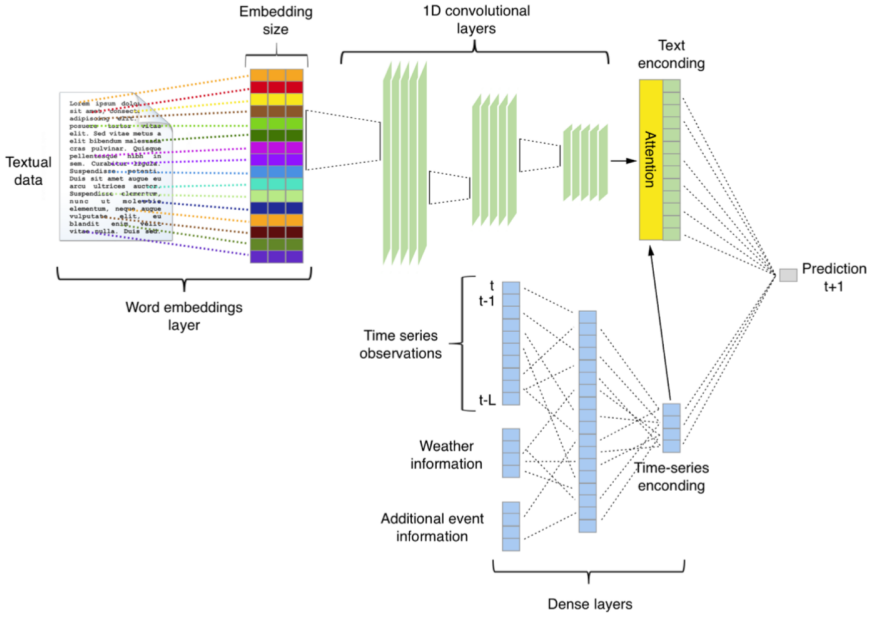


Figure 2.7: Proposed neural network architecture with FC layers for modelling the time-series observations; DL-FC [RMP19]

taxi demand with regular trend considered. Besides, the word embeddings on textual data are complicated and time-consuming. In the case of this project, only some data in festival days and event areas will be considered having event attributes; in other words, it will not explore how taxi demand related to a specific event.

2.5 Knowledge gap and Aims

After doing those literal surveys, some knowledge gaps are found on the topic of taxi demand prediction. Although some studies have used the recurrent neural networks to predict taxi demand, none of them considers related underlying factors. In other words, there is not a study that predicts the taxi demand for both regular trend and the trend caused by other factors. Therefore, this project aims to fill this gap by predicting the taxi demand for regular daily demand, together with the demand caused by other related factors. Besides, the comparison between different models will be made in this project as well. The expectation for the project is that after feeding an input data set into the model, it can predict the taxi demand at a location in a time interval accurately.

Chapter 3

Solution and Implementation

In this chapter, the solutions and implementations of the project will be explained. The process of implementation is split into two parts. The first one is data pre-processing, and the other one is model construction.

3.1 Data Pre-processing

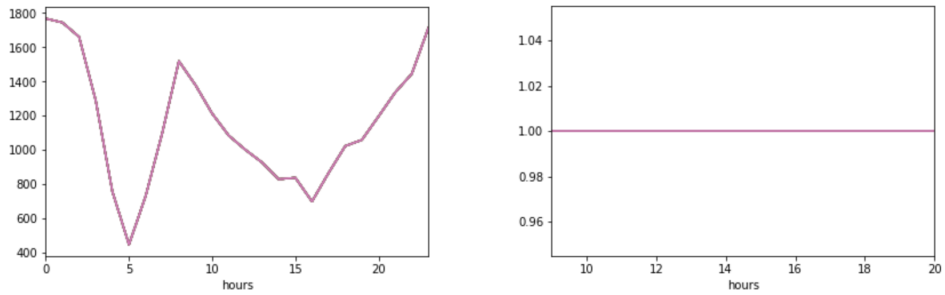
Data pre-processing is an essential step in this project because it can provide a reliable and useful data set to the model and, finally, produce good results. It consists of three parts, which are data visualization, data filtering, and data clustering. The details of these three parts will be explained in the following sections.

3.1.1 Data Retrieval

The project used the data set collected by the New York City Taxi Council, which is a government apartment. Taxi drivers themselves fill the data records. There are ten years of data, and this project used the data set from January and February in 2017. Figure 3.1 shows a part of the sample data set.

	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_flag	PULocationID	DOLocationID	payment_type	fare_amount	extra	mta_tax	tip_amount	tolls_amount	improvement_surcharge	total_amount
1	1	2017-01-09 11:13:28	2017-01-09 11:25:45	1	3.3	1	N	263	161	1	12.5	0	0.5	2	0	0.3	15.3
1	1	2017-01-09 11:32:27	2017-01-09 11:36:01	1	0.9	1	N	186	234	1	5	0	0.5	1.45	0	0.3	7.25
1	1	2017-01-09 11:38:20	2017-01-09 11:42:05	1	1.1	1	N	164	161	1	5.5	0	0.5	1	0	0.3	7.3
1	1	2017-01-09 11:52:13	2017-01-09 11:57:36	1	1.1	1	N	236	75	1	6	0	0.5	1.7	0	0.3	8.5
2	2	2017-01-01 00:00:00	2017-01-01 00:00:00	1	0.02	2	N	249	234	2	52	0	0.5	0	0	0.3	52.8
1	1	2017-01-01 00:00:02	2017-01-01 00:03:50	1	0.5	1	N	48	48	2	4	0.5	0.5	0	0	0.3	5.3
2	2	2017-01-01 00:00:02	2017-01-01 00:39:22	4	7.75	1	N	186	36	1	22	0.5	0.5	4.66	0	0.3	27.96
1	1	2017-01-01 00:00:03	2017-01-01 00:06:58	1	0.8	1	N	162	161	1	6	0.5	0.5	1.45	0	0.3	8.75
1	1	2017-01-01 00:00:05	2017-01-01 00:08:33	2	0.9	1	N	48	50	1	7	0.5	0.5	0	0	0.3	8.3
2	2	2017-01-01 00:00:05	2017-01-01 00:05:04	5	1.76	1	N	140	74	2	7	0.5	0.5	0	0	0.3	8.3
2	2	2017-01-01 00:00:05	2017-01-01 00:15:36	1	8.47	1	N	138	262	1	24	0.5	0.5	7.71	5.54	0.3	38.55
1	1	2017-01-01 00:00:06	2017-01-01 00:11:56	2	2.4	1	N	142	236	2	10.5	0.5	0.5	0	0	0.3	11.8
1	1	2017-01-01 00:00:06	2017-01-01 00:23:37	2	12.6	5	N	161	265	1	60	0	0	10	0	0.3	70.3
1	1	2017-01-01 00:00:06	2017-01-01 00:08:53	1	0.9	1	N	234	186	1	7	0.5	0.5	2.05	0	0.3	10.35
2	2	2017-01-01 00:00:06	2017-01-01 00:09:30	4	2.43	1	N	141	107	1	9.5	0.5	0.5	2.7	0	0.3	13.5
2	2	2017-01-01 00:00:06	2017-01-01 00:16:05	2	2.6	1	N	79	163	1	12.5	0.5	0.5	2.76	0	0.3	16.56
2	2	2017-01-01 00:00:06	2017-01-01 00:18:12	5	4.25	1	N	148	36	2	16.5	0.5	0.5	0	0	0.3	17.8
2	2	2017-01-01 00:00:07	2017-01-01 00:07:42	1	0.65	1	N	48	68	1	6.5	0.5	0.5	1.7	0	0.3	9.5
2	2	2017-01-01 00:00:09	2017-01-01 00:34:21	1	3.42	1	N	230	148	1	22.5	0.5	0.5	0	0	0.3	23.8
1	2	2017-01-01 00:00:10	2017-01-01 00:24:52	1	6.6	1	N	186	232	2	23	0.5	0.5	0	0	0.3	24.3
1	1	2017-01-01 00:00:10	2017-01-01 00:02:49	1	0.5	1	N	36	37	2	4	0.5	0.5	0	0	0.3	5.3
1	1	2017-01-01 00:00:11	2017-01-01 00:08:27	1	1.2	1	N	41	74	1	7.5	0.5	0.5	1.75	0	0.3	10.55
1	1	2017-01-01 00:00:12	2017-01-01 00:12:53	1	1.7	1	N	125	45	2	9.5	0.5	0.5	0	0	0.3	10.8
1	1	2017-01-01 00:00:12	2017-01-01 00:09:48	1	5.3	1	N	138	192	2	16	0.5	0.5	0	0	0.3	17.3
1	1	2017-01-01 00:00:13	2017-01-01 00:01:04	2	0.2	1	N	143	143	2	3	0.5	0.5	0	0	0.3	4.3

Figure 3.1: Jan 2017 sample taxi data in NYC



(a) Location: Manhattan. Alphabet City (b) Location: Queens. Jamaica Bay

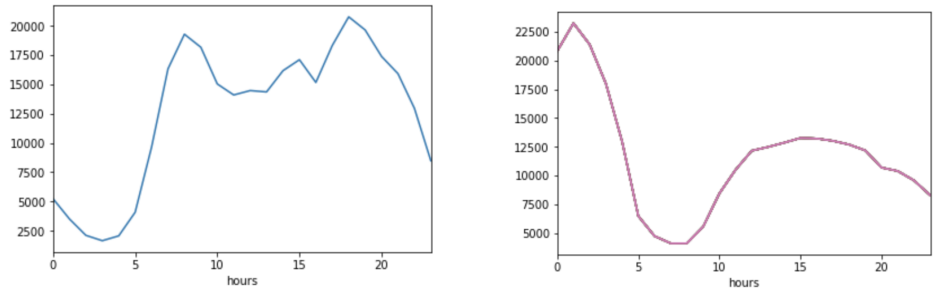
Figure 3.2: Daily Pickup Distribution on 07/01/2017

3.1.2 Data Visualization

The purpose of data visualization is to have a rough understanding of what data sets look like and what factors are related to taxi demand instead of intuitions and non-evidenced assumptions. To implement the step of data visualization, matplotlib in Python was used. In this project, the factors of location, time, and holiday are assumed to be related to taxi demand. Therefore, some data visualization plots are computed to confirm the assumptions.

In the assumptions, location is one factor that is related to taxi demand prediction. Therefore, Figure 3.2 is a line plot for the daily pickup distribution for two areas; one is in Manhattan Alphabet City, which is known as the 'heart' of New York City, and the other is in Queens Jamaica Bay which is a suburb of New York City. From the two plots, we can see that the taxi demand in Manhattan is much higher than that in Queens. Therefore, we can conclude that location is a factor that is related to taxi demand.

Another factor that is assumed is holidays. In this project, holidays stand for the festival holidays or weekends. In order to confirm this assumption, two graphs are plotted for comparison, as shown in Figure 3.3. The plot on the left-hand side is the daily pickup distribution on the 9th of January, which is a workday, and the other one is the plot for the New Year Holiday at the same location. From the comparison of the shapes of two plots, we can easily see that the daily pickups on regular days are different from that in holidays. Therefore, we can conclude that the holiday is one



(a) Regular Work day in 09/01/2017 (b) Festival holiday in 01/01/2017

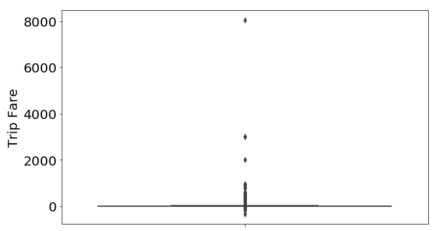
Figure 3.3: Daily Pickup Distribution on Regular day and Festival Holiday in the same location

factor that is related to taxi demand.

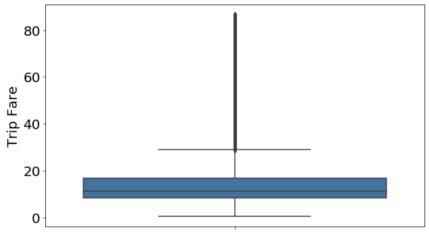
The factor time can also be observed from Figure 3.3. As shown in the graph.(a), there is an increase in the line from 5 to 10 in the morning. It is because people are going to work at that time interval, and some may choose to take taxis. There is another peak from around 4 to 8 in the afternoon. The reason behind it might be that some people choose to take taxis to go home after work. Besides, even in holidays, different pickups happened in different time intervals, which can be observed from the graph.(b) in Figure 3.3. Therefore, time is indeed a factor that is related to taxi demand.

3.1.3 Data Filtering

As mentioned before, since the records in the data set are filled by the drivers themselves, there might be some mistakes in the records. More importantly, the correctness of the results will be affected if these mistakes are kept in the data set. Therefore, the step of data filtering is necessary for the data set.

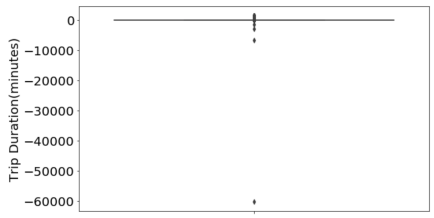


(a) Box Evaluation of Trip Fare before filtering



(b) Box Evaluation of Trip Fare after filtering

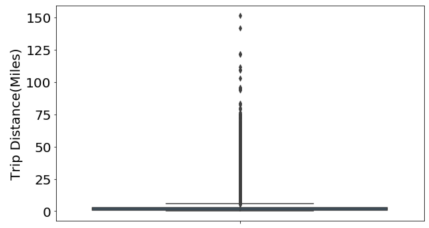
Figure 3.5: Trip Fare Filtering in dollars



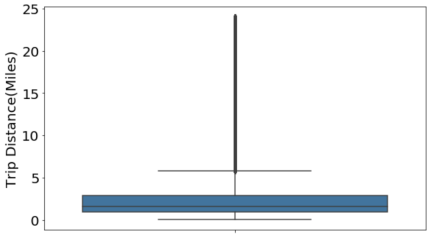
(a) Box Evaluation of Trip Duration before filtering

(b) Box Evaluation of Trip Duration after filtering

Figure 3.6: Trip Duration Filtering in minutes

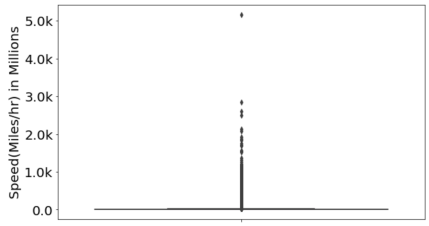


(a) Box Evaluation of Trip Distance before filtering

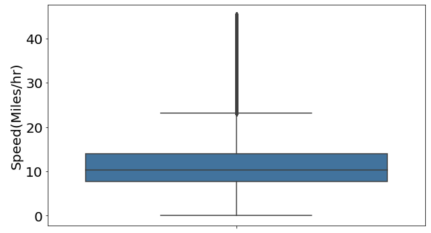


(b) Box Evaluation of Trip Distance after filtering

Figure 3.4: Trip Distance Filtering in miles



(a) Box Evaluation of Speed before filtering



(b) Box Evaluation of Speed after filtering

Figure 3.7: Speed Filtering in miles/hour

```
99.1 percentile value of trip distance is 18.9miles
99.2 percentile value of trip distance is 19.12miles
99.3 percentile value of trip distance is 19.41miles
99.4 percentile value of trip distance is 19.79miles
99.5 percentile value of trip distance is 20.2miles
99.6 percentile value of trip distance is 20.69miles
99.7 percentile value of trip distance is 21.21miles
99.8 percentile value of trip distance is 22.0miles
99.9 percentile value of trip distance is 24.03miles
100.0 percentile value of trip distance is 151.7miles
```

Figure 3.8: The trip distance percentage distribution in data set

```
quantile_tripDistance = new_frame_cleaned_speed.trip_distance.quantile(np.round(np.arange(0.991, 1.001, 0.001), 4))
qValues = np.round(np.arange(0.991, 1.001, 0.001), 3)
for i in qValues:
    print("{} percentile value of trip distance is {}miles".format((i*100), quantile_tripDistance[i]))
```

Figure 3.9: Python code for percentage calculating

Box evaluation plots in Python matplotlib was used in order to visualize how the project filters the data. Four attributes in the data set are filtered; they are trip distance, trip fare, trip duration, and speed. In each figure, it has two plots; the one on the left-hand side is before filtering, and the other one is after filtering. For Figure 3.4, we can see that one trip distance even has 150 miles, which definitely has left New York City; therefore, these outliers need to be removed. A percentage check for the whole data set was also implemented, which is presented in Figure 3.8. It can be found that 99.9% of the trips are locating within 24.03 miles. The related code is presented in Figure 3.9. Therefore, the data that have trip distance with 24.03 miles will be used. Similarly, I repeat the procedures for trip fare, trip duration and speed; Figure 3.5 and Figure 3.6 show the results. There is a difference when filtering in trip duration. The project will consider the data that have a trip duration within 12 hours instead of looking at the time that 99.9% of the data located. Since the trip duration can be affected by weather, traffic congestion, and many other factors, it is hard to distinguish which value is unreasonable.

3.1.4 Data Clustering

In Chapter 2, the K-Mean data clustering method has been introduced. As mentioned in that section, data clustering can help increase the accuracy of the taxi demand prediction. However, the studies which use those clustering methods cannot provide a reasonable explanation of why they choose the specific parameter values in their clustering methods. Most of them use the tryanderror method to see which parameter gives the best result. Therefore, in this project, there will be no external clustering methods being introduced; and instead, it will use the taxi zone partition, which is done by the New York City government. Figure 3.10 shows the map of the zone partition. If, in the late stage, results need improvements, those two clustering methods will be considered.

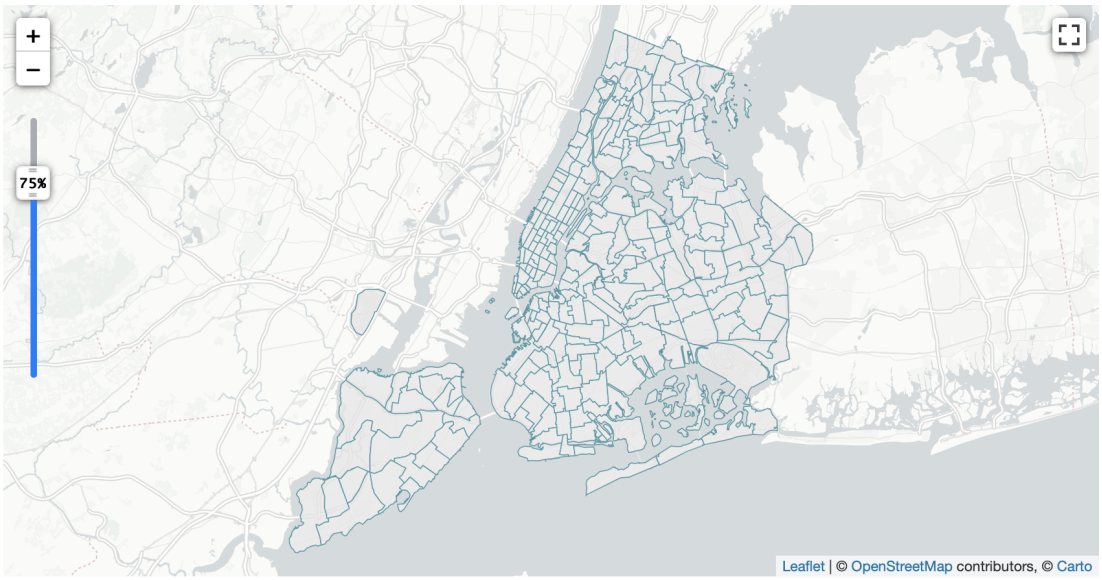


Figure 3.10: Taxi zone map in NYC

3.2 Model Construction

In this project, two models will be used to forecast taxi demand in New York City; they are the LSTM model and XGBoost model. In the following sections, the formation of input data and the implementation of the two models will be explained.

3.2.1 Data Input Format

dayofweek	PULocationID	Hours	time1	time2	time3	count
6	1	3	0	0	0	1
6	1	4	0	0	1	1
6	1	5	0	1	1	2
6	1	6	1	1	2	2
6	1	7	1	2	2	2
6	1	10	2	0	0	3
6	1	11	0	0	3	2
6	1	12	0	3	2	3
6	1	13	3	2	3	3
6	1	14	2	3	3	3
6	1	15	3	3	3	3
6	1	16	3	3	3	6
6	1	17	3	3	6	4
6	1	18	3	6	4	3
6	1	23	0	0	0	5
0	1	3	0	0	0	2
0	1	4	0	0	2	1
0	1	5	0	2	1	1
0	1	6	2	1	1	2
0	1	7	1	1	2	4
0	1	10	4	0	0	1
0	1	11	0	0	1	1

Figure 3.11: Input data set

Although the data set has been filtered and clustered, it is still not ready to feed into models. Features need to be determined in order to produce an efficient input data set. In previous sections, data visualization has been done, and it is found that

time, location, holidays are the factors that are related to the taxi demand. Therefore, the input data set is going to contain these factors and consider them as features. The project format the input data set using Python pandas, which is a powerful data manipulation tool; it reads the CSV files and converts them into data frames.

As shown in Figure 3.11, the input data set will have six features. 'dayofweek' represents the weekday of the pickups, which has been encoded into numbers; Monday is 0 and Sunday is 6. PULocationID stands for pick up location id, and it links to the zone divisions in Figure 3.10. 'Hours' is the time slot for the pickups since one day has been divided into 24 one-hour slots and encoded into numbers from 0 to 23. 'time1', 'time2', and 'time3' are the number of pickups in the previous three hour slots of the current hour; these are time series features. 'count' is the number of pickups in the current hour.

This input data set has many advantages. It contains the time relationships using the pickups from the previous three hours. It also has data from all locations which can train the model efficiently. Besides, the input data set will be normalized before feeding into models since normalization can reduce the range of input data values and improve the results in the end.

3.2.2 Model Implementation

In the following content, the implementation of the two models will be explained. Since these are two completely different models, they will be illustrated in two separate sections.

Long-Short Term Memory Model

To construct the LSTM model, Keras library is used. Figure 3.12 shows the Keras code to construct the LSTM model. The Sequential model means a linear stack of layers has been built for the LSTM model. Fifty LSTM cells have been added to the

```
model = Sequential()
model.add(LSTM((50), activation='relu', batch_input_shape=(None, 6, 1), return_sequences=True))
model.add(LSTM((1),return_sequences=False))
model.add(Dense(1))
model.compile(loss='mse', optimizer='adam',metrics=['accuracy'])
```

Figure 3.12: LSTM code

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 6, 50)             10400
_________________________________________________________________
lstm_2 (LSTM)                (None, 1)                 208
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2
=================================================================
Total params: 10,610
Trainable params: 10,610
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.13: LSTM parameters

first LSTM layer, and the activation function for this layer is Relu function. One cell layer is followed since the output is a single value. At the end of the network, a Dense Network is appended. The last step is to compile the model with an optimizer set to Adam, and the loss function is MSE, which stands for mean square error. Figure 3.13 shows the number of parameters in each layer of the network.

In Figure 3.14, it shows how does the LSTM model fit the training data. 'Epochs' is a parameter to show how many iterations the model will be trained. The evaluation section in the code shows how RMSE, MAE, and MPE are calculated. The mathematical equations for these evaluation error methods have been explained in Chapter 2.

```
# Train the LSTM model
model.fit(X_train, y_train, epochs=150, validation_data=(X_test, y_test))
results = model.predict(X_test)

# Evaluation
RMSE = math.sqrt(mean_squared_error(real_y[:], real_predict[:,0]))
MAE= statistics.mean(abs(real_y[:] - real_predict[:,0]))
MPE = np.sum(abs(real_y[:] - real_predict[:,0]))/np.sum(real_y[:])
```

Figure 3.14: LSTM training and evaluation

```
xg_reg = xgb.XGBRegressor(objective='reg:logistic', colsample_bytree = 0.9,
                          subsample= 0.96, min_child_weight = 6.19, learning_rate = 0.3,
                          max_depth=14, gamma=1, n_estimators = 463)
xg_reg.fit(X_train,y_train)
preds = xg_reg.predict(X_test)
```

Figure 3.15: XGBoost model code

```
# Fit data into the XGBoost model
xg_reg.fit(X_train,y_train)

# Make predictions
preds = xg_reg.predict(X_test)

# Evaluation
RMSE = np.sqrt(mean_squared_error(y_test[:], preds[:]))
MAE = statistics.mean(abs(y_test[:] - preds[:]))
MPE = np.sum(abs(y_test[:] - preds[:]))/np.sum(y_test[:])
```

Figure 3.16: XGBoost training and evaluation

XGBoost Model

In python, there is an XGBoost library, which is an open-source for developing. There are many parameters in the model, and different combinations of them can lead to different performances. Figure 3.15 is the code for the model construction.

The 'objective' parameter determines the loss function to be used; 'reg:logistic' means that the model is using logistic regression as its loss function. There are many other objectives, such as 'count:poisson' and 'reg:squarederror'. The 'colsample bytree' indicates the percentage of features used per tree, which is like the random forest algorithm. The 'subsample' is the percentage of samples used per tree. 'Learning rate' is the step size shrinkage used to prevent overfitting. 'n estimators' determines the number of trees to be built [xgb]. These are the key parameters that affect model performance the most.

As shown in Figure 3.16, 'xg_reg' created in Figure 3.15 is used to fit the training data. Predicted values are calculated by calling "predict" function. The evaluation error methods are like that in the LSTM model.

Chapter 4

Evaluation

This chapter is mainly provided for the purpose of showing the evaluation results of the predictions by the LSTM model and the XGBoost model. There are four sections in total; they are overall evaluation, feature importance, parameter tuning, and different locations. In each section, the results will be formatted into a table for easy comparison, and the discussions will also be presented underneath.

Three evaluation methods will be used. RMSE indicated the error of the model; the smaller this value is, the better the model is. MAE represents the mean absolute error; it shows the difference between predicted values and real values in the unit of pickups. MPE is the mean percentage error of all locations; the smaller the value it has, the better the model is. The details and mathematical equations on these evaluation methods are explained in Chapter 2.

4.1 Overall Evaluation & More data

In this section, the overall evaluation results for both models will be discussed, and results for more data will also be presented.

For the LSTM model, in Figure 4.1, when having 89541 input data records, it has an

RMSE value of 38.93, MAE values of 18.85, and 0.18 for MPE. The value 18.85 of MAE shows that the difference between predicted values and real values is around 19 pickups in all locations on average. MPE of value 0.18 shows that, in all locations, there are 18% error difference in average. However, when increasing the number of input data from 89541 to 170661, all three error values for the LSTM model decrease in a non-neglectable amount. It has 32.36 for RMSE, 16.47 for MAE, and 0.15 for MPE.

For the XGBoost model, in Figure 4.1, when having 89541 input data records, it has an RMSE value of 35.75, MAE values of 16.48 and 0.15 for MPE. When increasing the number of input data from 89541 to 170661, all three error values decrease, but the amount is not as high as that for the LSTM model. It has 34.80 for RMSE, 16 for MAE, and 0.14 for MPE.

model	number of data	RMSE	MAE	MPE
LSTM	89541	38.93	18.85	0.18
	170661	32.36	16.47	0.15
XGBoost	89541	35.75	16.48	0.15
	170661	34.80	16.00	0.14

Figure 4.1: Jan 2017 sample taxi data in NYC

If comparing the performance between the LSTM model and the XGBoost model, it is not hard to observe that when the number of input data is small, the XGBoost model is better than the LSTM model in all evaluation methods. However, when increasing the number of input data, the performance improvement for the LSTM model is more substantial than that for XGBoost, especially in RMSE. Although in MAE and MPE, the XGBoost model is still slightly better than the LSTM model when having more data, the RMSE for the LSTM model is much better than XGBoost. Therefore, from these comparisons, it can be concluded that the LSTM model is better than the XGBoost model when having more data.

Figure 4.2 and Figure 4.3 are the plots of the first 100 results for LSTM model and XGBoost model respectively. The red line represents the predicted values, and the

green line represents the real values. It can be found that the red line matches the green line in most of the areas for both models, but at the peak values, both models can not predict well. Besides, when comparing two models, the XGBoost model performs slightly better than the LSTM model on those peak values.

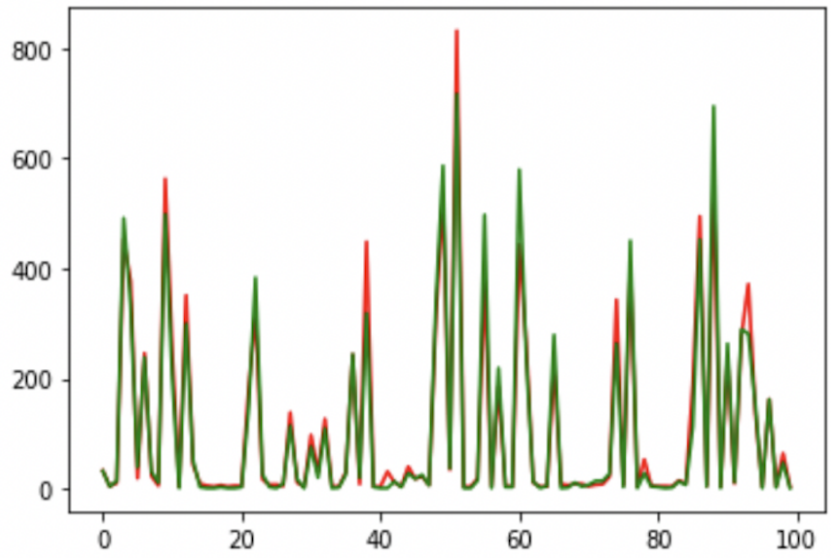


Figure 4.2: LSTM model evaluation graph

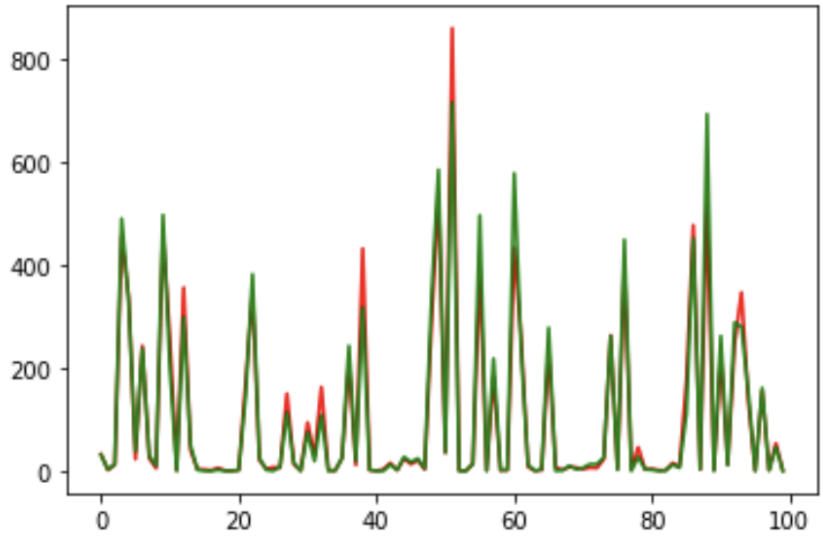


Figure 4.3: XGBoost model evaluation graph

4.2 Feature Importance

This section will explore the feature importance for both models. Exciting discoveries from the results will be discussed for each model in separate subsections.

4.2.1 Long-Short Term Memory Model

LSTM model	RMSE	MAE	MPE
All features	37.02	18.28	0.17
Without Hours	43.00	20.32	0.19
Without Locations	38.69	18.90	0.18
Without weekdays	38.89	21.61	0.20
Without Time1	37.44	19.57	0.18
Without Time1 & 2	37.31	17.33	0.16
Without All Times	146.54	96.70	0.90

Figure 4.4: LSTM model Feature Importance

Figure 4.4 is a table which shows error evaluation results for different input features. The first row, "All features", is the results for the LSTM model with all input features. The following rows are the results for the model with missing features. Therefore, the feature importance can be compared.

It is not hard to observe that when the input data does not have all times, which means Time1, Time2, Time3 are all missing, the performance of LSTM model is affected most; RMSE raises from 37.02 to 146.54, MAE raises to 96.70 and MPE reaches the values of 0.9. Therefore, it can be concluded that deleting all times has the most influence on the prediction of the LSTM model. This conclusion matches the expectations because the LSTM model needs time connections for training, which is explained in chapter 2. However, when comparing the last three rows, it is found that within features of Time1, Time2, Time3, Time3, which is the previous hour of the current hour, has the most significant effects on the results. Besides, when the input data does not have Time1 and Time2, the MAE and MPE are better than that when it has Time1 and Time2, which can be seen from the comparison between the first row and second last row.

The reason for this might be that time1 and time2 are the pickups that happened two and three hours before the current hour, which may not be helpful in prediction and even produce side effects. For the features of Hours, Locations, weekdays, if input data miss any one of them, the results will be worse. The effects caused by Hours are more substantial than that by weekdays, and locations have the least effects within these three features.

4.2.2 Extreme Gradient Boosting Model

XGBoost model	RMSE	MAE	MPE
All features	48.24	24.75	0.23
Without Hours	51.18	26.43	0.25
Without Locations	47.86	24.02	0.22
Without weekdays	48.09	24.77	0.23
Without Time1	48.12	23.89	0.22
Without Time1 & 2	46.28	23.94	0.22
Without All Times	87.57	53.46	0.50

Figure 4.5: XGBoost model Feature Importance

Figure 4.5 shows a table of error evaluation results for different input features. The effects of these features in the XGBoost model are similar to those in the LSTM model. Although input data without all Times has the most effects on the results, it is not as large as that in the LSTM model, which is in expectations because the XGBoost model does not strict requirements about time connections. XGBoost model is an advanced gradient boosting model; the number of features is more important to it. Therefore, when missing all times, the input data only has three features left, and it causes worse performance. Similar discoveries are also found in the XGBoost model. The missing of Time1 and Time2 produce better results, which might be the same reason as that in the LSTM model. For the features of Hours and weekdays, without any one of them would produce worse results. However, for locations, without it produces better results. The reason for this exciting finding might be that the number of pickups in different locations are quite different, which can be justified from data visualization in chapter 3, and XGBoost model can not resolve it like LSTM model since it is a gradient boosting

model based on decision trees.

4.3 Parameter Tuning

This section will explore how different parameter decisions influence both models. Exciting discoveries will be discussed for each model in separate subsections. In this section, evaluations are not only on three error methods, but the trade-off in training time will also be looked at.

4.3.1 Long-Short Term Memory Model

In the LSTM model, three kinds of parameters can be tuned; they are the number of LSTM cells, the number of training epochs, and the training-testing ratio of the input data. There are other kinds of parameters that could be tuned, but in this project, only these three will be tested.

LSTM model Parameter - Cell

Number of LSTM cells	Number of parameters	RMSE	MAE	MPE	Time
25	2810	39.99	19.05	0.18	Around 20mins
50	10610	38.96	18.85	0.18	Around 45mins
100	41210	36.68	18.84	0.18	Around 60mins

Figure 4.6: LSTM model Parameter - Cell

The number of LSTM cells is directly related to the number of parameters that will be used in the model. In Figure 4.6, it can be seen that the more LSTM cells the model has, the more parameters the model will use. Besides, as the number of cells increases, the errors are becoming smaller; but the time for the training process increases. Since, in practice, different applications will have different requirements, it is a trade-off between

accuracy and training time. In the case of this project, the goal is to achieve high accuracy; the number of cells chosen is 50.

LSTM model Parameter - Epoch

Epochs	RMSE	MSE	MPE	Time
100	38.93	17.99	0.17	Around 30mins
150	38.96	18.85	0.18	Around 45mins
200	37.02	18.28	0.17	Around 60mins

Figure 4.7: LSTM model Parameter - Epoch

One epoch in the training process of the LSTM model means that the whole data set has been trained for one time. In order to make the model stable and produce the best results, more epochs should be taken. In Figure 4.7, it is not hard to observe that the more epochs the model has, the smaller the errors will be. However, a larger number of epochs consume longer training time, which can be seen from the 'Time' column. Since the performance does not improve much, epochs of 100 are chosen to produce the best results in this model.

LSTM model Parameter - Ratio

Training:Test	RMSE	MAE	MPE
2:8	42.62	20.96	0.19
3:7	41.73	19.34	0.18
4:6	39.49	18.66	0.17
5:5	40.54	18.90	0.17
7:3	38.14	17.85	0.16
8:2	38.96	18.85	0.18

Figure 4.8: LSTM model Parameter - Ratio

The training-test split ratio determines how many input data will be used for training and testing. In order to produce the best results, a suitable amount should be chosen. From Figure 4.8, it can be seen that the smaller this training-testing ratio is, the larger

prediction errors will the model has. However, when the ratio is 7:3, the LSTM model has the best performance.

4.3.2 Extreme Gradient Boosting Model

XGBoost model Parameter - Learning rate

Learning rate [0,1]	RMSE	MAE	MPE
0.01	49.37	29.89	0.28
0.1	48.23	24.75	0.23
0.3	47.60	23.14	0.22
0.5	48.71	24.45	0.23

Figure 4.9: XGBoost model Parameter - Learning rate

The learning rate in the XGBoost model is a parameter that determines the amount that the weights are updated during training. If the learning rate is too large, the model might fail to converge. If the learning rate too small, the training process may take a long time. Therefore, it is important to choose a suitable value. As shown in Figure 4.9, when the learning rate equals 0.3, the XGBoost model produces the best results. Any value larger or smaller than 0.3 makes the performance worse.

XGBoost model Parameter - Objective

Objective	RMSE	MAE	MPE
reg:squarederror	47.60	23.14	0.22
reg:logistic	35.75	16.48	0.15
count:poisson	37.06	17.06	0.16

Figure 4.10: XGBoost model Parameter - Objective

As explained in Chapter 3, 'objective' in the XGBoost model determines what loss function will be used. 'Reg:squarederror' means regression with squared loss. 'Reg:logistic' represents logistic regression. 'count:poisson' is poisson regression for count data, and

it outputs mean of poisson distribution. In Figure 4.10, it can be seen that 'reg:logistic' produces the best results.

XGBoost model Parameter - Ratio

Training:Test	RMSE	MAE	MPE
2:8	57.60	31.34	0.29
3:7	55.06	29.15	0.27
4:6	53.78	27.65	0.25
5:5	50.86	26.59	0.25
7:3	49.23	25.00	0.23
8:2	48.23	24.75	0.23

Figure 4.11: XGBoost model Parameter - Ratio

In expectation, the XGBoost model will have better results if more input training data is fed since it is based on an advanced gradient boosting algorithm. As shown in Figure 4.11, the larger training-testing ratio is, the better results XGBoost model produces, and this justifies the expectations.

4.4 Different Locations

In this section, the performance on different locations conducted by the LSTM model and XGBoost model will be compared and discussed. Four featured locations will be looked at. They are Central Park, Midtown Center, JFK Airport, and Jamaica Bay in Queens.

4.4.1 Central Park

Central Park is an urban park in Manhattan in New York City. Central Park is the most visited urban park in the United States. There are 37.5–38 million visitors annually in estimation. Central Park is also one of the most filmed locations in the world [cen19].

The predictions done by both models are shown in Figure 4.12. In both plots, the red line represents the predicted values, and the green line is the real values. From the three error methods, it can be concluded that the XGBoost model has better results than the LSTM model. It can also be seen from the plots that the XGBoost model matches better in the graph.

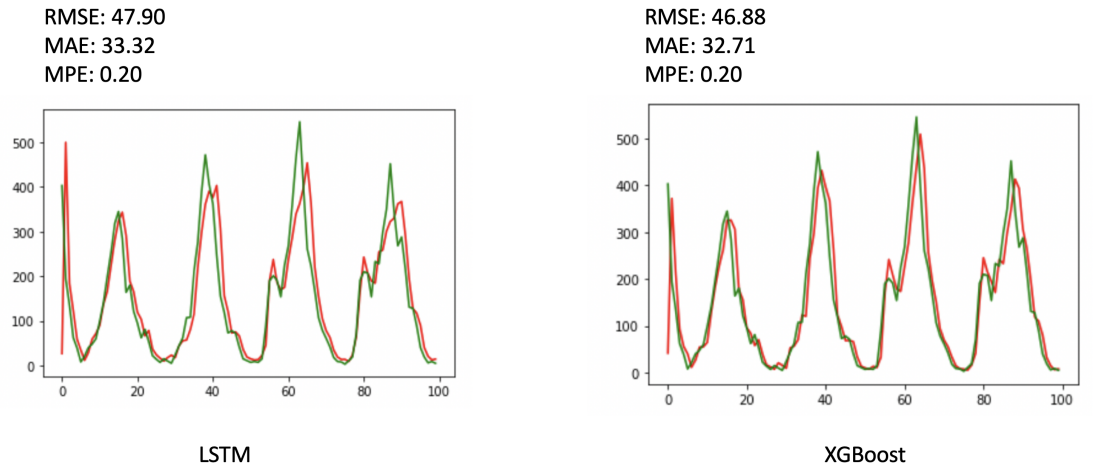


Figure 4.12: LSTM & XGBoost model prediction at Central Park

4.4.2 Midtown center

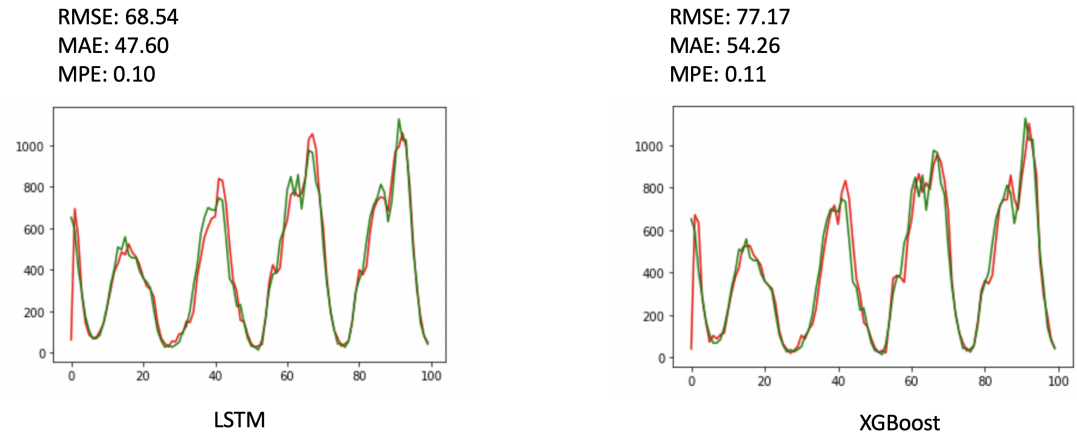


Figure 4.13: LSTM & XGBoost model prediction at Midtown Manhattan

Midtown Manhattan is the largest central business district in the world and ranks among the most expensive pieces of real estate. Midtown Manhattan is the central

portion of Manhattan in New York City. Midtown is home to some of the city's most iconic buildings, which includes the Empire State Building, Grand Central Terminal and Times Square [mid19].

In the location of Midtown Manhattan, as shown in Figure 4.13, the LSTM model has a better performance than the XGBoost model.

4.4.3 John F. Kennedy International Airport

John F. Kennedy International Airport is an international airport in New York which is also referred to as JFK airport. It is the primary international airport serving New York City. The airport is the busiest international air passenger gateway into North America. It has handled just over 61 million passengers in 2018 [JFK19].

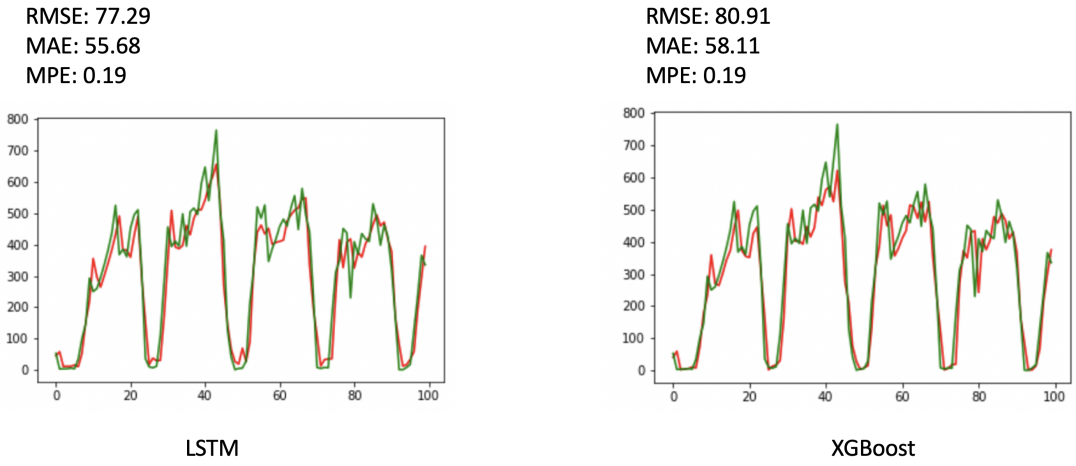


Figure 4.14: LSTM & XGBoost model prediction at John F. Kennedy International Airport

An airport is a typical place that has a great demand for taxis. From Figure 4.14, it can be seen that the LSTM model has a better performance than XGBoost in RMSE and MAE.

4.4.4 Jamaica Bay in Queens

Jamaica Bay is a partially natural, partially human-built estuary on the southern portion of the western tip Long Island, in New York [Jam19]. It is a place that rarely has people visited, which can be justified from data visualization in Chapter 3.

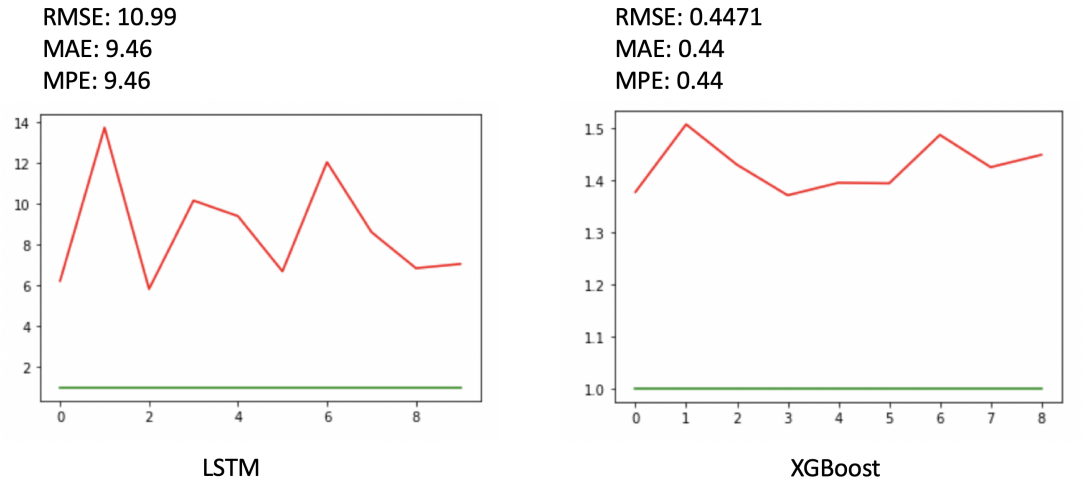


Figure 4.15: LSTM & XGBoost model prediction at Jamaica Bay in Queens

As stated before, Jamaica Bay in Queens is not a popular place. Therefore, most of the time, the real value is 0, which is shown in green in plots in Figure 4.15. It is not hard to observe that the LSTM model, in this case, does not perform as good as the XGBoost model. LSTM model predicts the number of pickups around ten, but XGBoost model forecasts between 1 and 2.

After analyzing the performance of both models in these featured places, it can be concluded that the LSTM model does a slightly better job in predicting in large population areas like Midtown center and John F. Kennedy International Airport. For some moderate places like Central Park, both models perform relatively equal. However, for low population areas like Jamaica Bay in Queens, the XGBoost model is much better than the LSTM model.

Chapter 5

Conclusion and Future work

5.1 Conclusion

The goal of this project is to analyze and predict taxi demand in New York City. The main challenges are data pre-processing, model constructions, and evaluations. As explained in previous chapters, data pre-processing include data visualization, data filtering, and clustering; model constructions include formatting input data and building models. In evaluations, three error methods have been chosen, and various evaluation perspectives have been explored in order to see which model is better.

The results of the predictions are satisfying and in expectation. For both the LSTM model and the XGBoost model, the best prediction difference is controlled within 16 pickups on average in all locations. For different featured locations, different models should be chosen in order to produce the best forecasting results. For example, in the business center and airports, the LSTM model is more suitable since these locations have specific patterns connected with time. In tourist attractions or some rarely visited places, the XGBoost model is more suitable since this model is based on an advanced gradient boosting model, which is strong enough to handle irregular data patterns. Besides, different parameters for different models can also produce different performance; therefore, in order to produce the best results, the right parameter combinations should

be chosen.

5.2 Future Work

There are many limitations to this project. Since both the LSTM and the XGBoost models are complicated, many different parameters can affect their performance. Besides, in this project, the data set is clustered by the NYC government according to different featured locations. It is possible that if different cluster methods are used, better results would be produced. Therefore, there are many opportunities for future work based on this project.

5.2.1 Parameter Tuning

In the future, it would be best to choose the parameters systematically. Although it would consume much time for training and testing, it is worth to do in order to produce the best-fitted models. Besides, not only the parameters evaluated in Chapter 4 can affect the models. Many other parameters can influence the model. For example, hyper-loss functions can be introduced to produce better results because, as seen in Chapter 4, both models do not perform well at peak values. When using different loss functions at peak values, the results might become better.

5.2.2 General Application

In this project, taxi demand in New York City has been chosen to analyze. If data in other cities, like Shanghai, Sydney, and Singapore, can be accessed, predictions can be made in those cities as well. It is also interesting to explore the common patterns between them and find a general application that can predict taxi demand in many cities.

Bibliography

[Bro16] Jason Brownlee. A gentle introduction to xgboost for applied machine learning, Sep 2016.

[cen19] Central park, Nov 2019.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[CTC+08] Han-wen Chang, Yu-chin Tai, Hsiao-wei Chen, Jane Yung-jen Hsu, and CP Kuo. itaxi: Context-aware taxi demand hotspots prediction using ontology and data mining approaches. *Proc. of TAAI*, 2008.

[DG06] M. A. Deshmukh and R. A. Gulhane. Importance of clustering in data mining. *International Journal of Scientific Engineering Research*, 7(2):247–251, February 2006.

[DRJ16] N. Davis, G. Raina, and K. Jagannathan. A multi-level clustering approach for forecasting taxi travel demand. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 223–228, Nov 2016.

[EAM06] Jeffrey Erman, Martin Arlitt, and Anirban Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data*, MineNet '06, pages 281–286, New York, NY, USA, 2006. ACM.

[GSC99] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

[Hao17] Zhang Hao. Lstm and gru – formula summary. *Isaac Changhau*, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[HW79] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[Jam19] Jamaica bay, Nov 2019.

[JFK19] John f. kennedy international airport, Nov 2019.

[KJFF15] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks, 2015.

[KMN+02] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):881–892, 2002.

[KYS13] Camille Kamga, M. Anil Yazici, and Abhishek Singhal. Hailing in the rain: Temporal and weather-related variations in taxi ridership and taxi demand-supply equilibrium. 01 2013.

[mid19] Midtown manhattan, Nov 2019.

[Mon18] Saul Montoya. How to calculate the root mean square error (rmse) of an interpolated ph raster?, Jul 2018.

[Mor19] Vishal Morde. Xgboost algorithm: Long may she reign!, Apr 2019.

[RMP19] F. Rodrigues, I. Markou, and F.C. Pereira. Combining time-series and textual data for taxi demand prediction in event areas: A deep learning approach. 49:120–129, 2019.

[Sha18] Gaurav Sharma. Taxi demand prediction- new york city, Sep 2018.

[Vat09] Andrea Vattani. The hardness of k-means clustering in the plane. *Manuscript, accessible at http://cseweb. ucsd. edu/avattani/papers/kmeans_hardness. pdf*, 617, 2009.

[xgb] Xgboost parameters.

[XRBT18] J. Xu, R. Rahmatizadeh, L. Bölöni, and D. Turgut. Real-time prediction of taxi demand using recurrent neural networks. *IEEE Transactions on Intelligent Transportation Systems*, 19(8):2572–2581, Aug 2018.

[ZC14] Pei-Yuan Zhou and Keith Chan. A model-based multivariate time series clustering algorithm. 05 2014.

[ZFC+16] K. Zhang, Z. Feng, S. Chen, K. Huang, and G. Wang. A framework for passengers demand prediction and recommendation. In *2016 IEEE International Conference on Services Computing (SCC)*, pages 340–347, June 2016.

[ZKF+16] K. Zhao, D. Khryashchev, J. Freire, C. Silva, and H. Vo. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 833–842, Dec 2016.