REAL-TIME TAXI PREDICTION

CAPSTONE IN-HOUSE PROJECT

REVIEW-3

Guide Name: **Prof. Krishnamoorthy A**
Designation: Assistant Professor Sr.Grade 1
Mobile No.8056424062
Mail Id: krishnamoorthy.arasu@vit.ac.in

Deployed at https://taxi-demand.herokuapp.com/

Name : **Sourav Dey**
Reg. No. **17BCE0019**
Mobile No.8318788891
Mail Id.
sourav.dey2017@vitstudent.ac.in

Name : **Sweta Kumari**
Reg. No.**17BCE2388**
Mobile No.8521059895
Mail Id.
sweta.kumari2017@vitstudent.ac.in

Name : **Arushi Das**
Reg. No.**17BCE0087**
Mobile No.8556082463
Mail Id.
arushi.das2017@vitstudent.ac.in

INTRODUCTION

Taxi services have become increasingly common these days for a variety of reasons. Every day, businesses expand their service areas to offer such services to their customers. As a result, more taxis are traveling long distances to pick up or drop off passengers. Taxi drivers have no idea where their next customer will be waiting for them. The passengers normally would have to wait a long time for a taxi to come pick them up, which costs the driver both in terms of time and fuel. This paper investigates a method for forecasting taxi demand in various areas using historical data. This will train the model to be able to predict future demand using a portion of the dataset known as training data. The model will then be evaluated against a test set, and the error rate, or the difference between expected and actual values, will be measured, with the model attempting to reduce that rate as much as possible.

MOTIVATION

Taxis are an important element of urban life and are in large quantities. They are used to traveling to work, for pleasure, for solitude, and most importantly for the ease of having no pauses in between the pick-up and the end destination, as well as no other unidentified passengers to board or disembark. Many multibillion-dollar businesses have risen from the provision of such services to consumers via Internet apps, portals, or local services.As a result, the following circumstances emerge: Taxi drivers are frequently unable to locate fares to serve and thus miss out on fare possibilities; taxi drivers are frequently too far away from the fare, wasting time and fuel in the process; and clients are frequently forced to wait for long durations, either due to a lack of taxis or because the taxi is far enough away from their location. Customer satisfaction suffers as a result, and the cost of fuel and time rises.Taxi demand forecasting is difficult due to the numerous unconnected factors and the lack of a reliable source to collect them. Historical data may be utilized to obtain the necessary knowledge to aid in the forecasting of such requests.

AIM OF THE PROPOSED WORK

The population of urban regions is rapidly growing, and the need for transportation is growing as well. The number of available taxi drivers is beginning to dwindle, and we are seeing this in our daily lives. Taxis are scarce late at night, making it risky for anybody to be around. People are forced to wait for extended periods of time in risky conditions, lowering the taxi service's overall satisfaction rating, drivers are unsure of where to seek for the next fare after dropping off a passenger, and taxi drivers are hesitant to travel to a rather remote place for fear of not finding any customers and wasting time and fuel. If the demand for cabs can be foreseen, such problems can be avoided. I'd want to be able to forecast demand in a certain region so that a dispatch system can distribute taxis.

OBJECTIVES OF THE PROPOSED WORK

The models will be able to anticipate demand in real time. It must be deployable inside automobiles. In the following hour, drivers may choose to go to a place with higher demand. Drivers will be less inclined to switch to Uber/Lyft when their earnings increase. Taxi taxi firms could be able to keep their leasing money for a long time. If firms keep a share of the fares, they might collaborate with drivers to devise a strategy for dispatching their taxis during the day in order to gain profit.

LITERATURE SURVEY

Kai Zhao, Denis Khryashchev, Juliana Freire, Cla´udio Silva, and Huy Vol have divided the paper into 2 sections, first section is finding about the maximum predictability which basically is defined by the entropy of the taxi demand while considering both the randomness and temporal correlation [1]. The second section is to choose the right predictor amongst the three; Markov predictor, Lempel-Ziv-Welch Predictor and a neural network predictor. The paper suggest that involving more factors like whether conditions will help the Neural Network perform better, but the Markov predictor is better in terms of efficiency as it does its computation at 0.003% of the NN. The maximum predictability they have reached is an average of 83%.

The paper by Kiam Tian Seow, Nam Hai Dang and Der-Horng Lee approaches taxi demands by automating the assigning of taxis [2]. It focuses on group customer satisfaction instead of individual customer satisfaction by simultaneously assigning taxis to the number of requested services in a given time period. It concludes that being able to regroup the taxi demands based on the proximity of taxi and its passengers before every dispatch cycle and have the taxis roam in places that have a high predictability of requests, it decreases customer waiting time up to 33.1% and taxi cruising time by 26.3%.

Der-Horng Lee and Ruey Long Cheu conduct their research on the existing taxi dispatch system and its effectiveness [4]. It Highlights that the current dispatch system assigns the taxi the request and shows the shortest time possible without considering the traffic conditions. It proposes that when a request is made the closest taxi to it is assigned. This leads to more than 50% reductions in passenger pick up time and average travel distance.

Étienne Simon, Alex Auvolat , Pascal Vincent and Yoshua Bengioy used the multi-layer perceptrons, bidirectional recurrent neural networks and models [7]. Their goal is to predict a fixed length output from a variable length sequence. They used the dataset that composed of all the complete trajectories of 442 taxis running in the city of Porto(Portugal) for a complete year. The training dataset contains 1.7 million datapoints, each one representing a complete taxi ride. Their fully-automated neural network approach to predict the destination of a taxi.

Jun Xu , Rouhollah Rahmatizadeh, Ladislau Bölöni, in this paper, proposed the sequence learning technique that predicts the future taxi request in each area of the city based on the recent demand and other relevant information [8]. They used the LSTM, Long Short Term Method for the learning technique to store the relevant information for future use. They used the dataset of taxi in New York City by dividing the whole city into small areas and predicting the demand in each area. The overall result shows that their method outperformed the other prediction method, such as feed-forward neural networks.

SYSTEM ARCHITECTURE

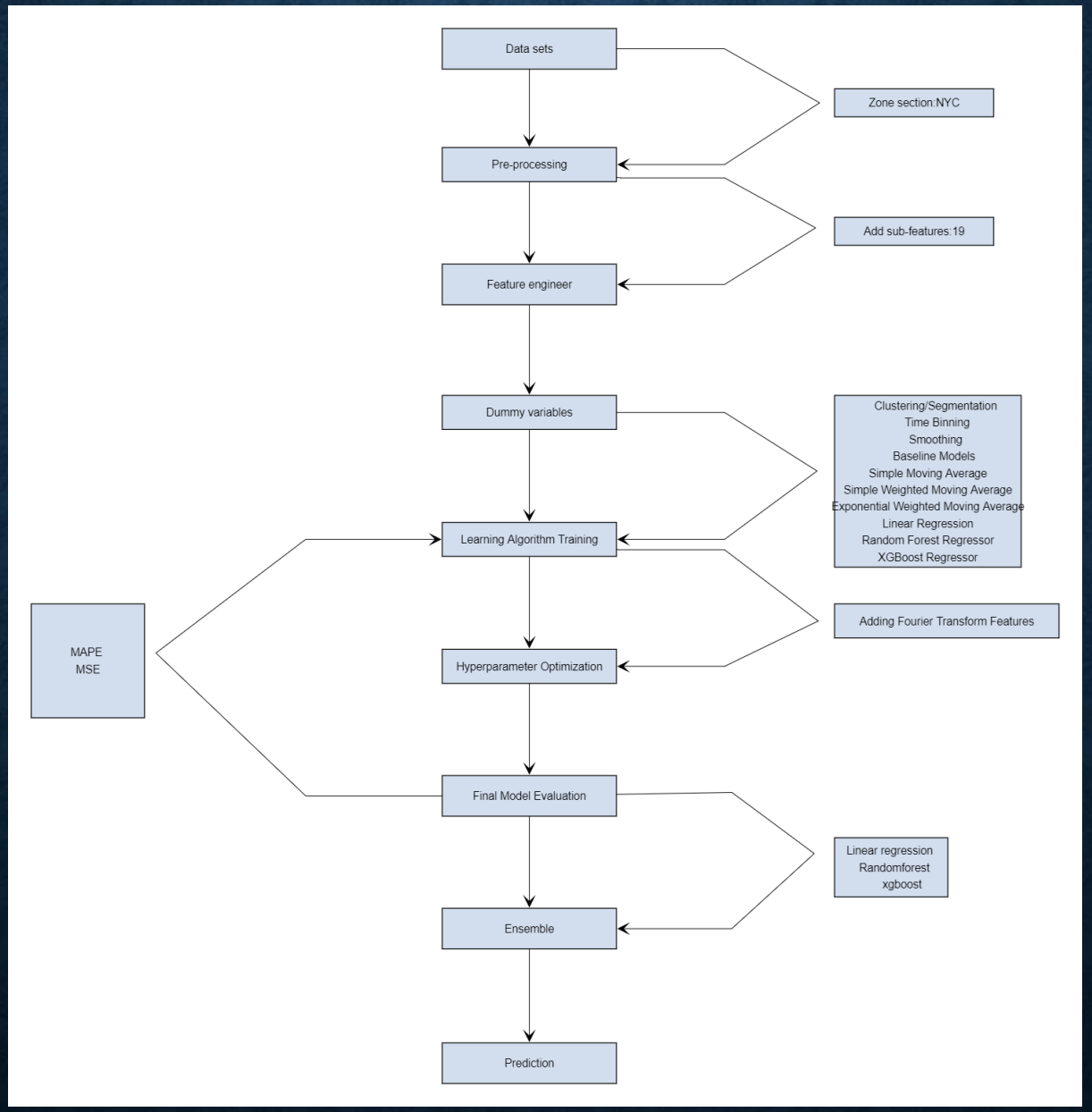


Fig1: System Architecture Design

Field Name	Description
VendorID	A code that identifies the TPEP provider who created the record. 1. CM Technologies 2. VeriFone Inc.
tpep_pickup_datetime	The date & time when the meter was turned on.
tpep_dropoff_datetime	The date & time when the meter was turned off.
Passenger_count	The number of passengers; value entered by the driver.
Trip_distance	Total trip distance in miles reported by the taximeter.
Pickup_longitude	Longitude where meter was started.

Fig 2: Dataset Sub-features

Pickup_latitude	Latitude where meter was started.
RateCodeID	The final rate code in effect: 1. Standard rate 2. JFK 3. Newark 4. Nassau or Westchester 5. Negotiated fare 6. Group ride
Store_and_fwd_flag	This flag indicates whether the trip record was held in vehicle memory before sending it to the vendor, aka "store and forward," because the vehicle did not have a connection to the server. Y= store and forward trip N= not a store and forward trip
Dropoff_longitude	Longitude where the meter was stopped.
Dropoff_ latitude	Latitude where the meter was stopped.
Payment_type	A numeric code signifying how the passenger paid for the trip. 1. Credit card 2. Cash 3. No charge 4. Dispute 5. Unknown 6. Voided trip

Fig 3: Dataset Sub-features (Continued)

Fare_amount	The time-and-distance fare calculated by the meter.
Extra	Miscellaneous extras and surcharges. Currently, this only includes. the $0.50 and $1 rush hour and overnight charges.
MTA_tax	0.50 MTA tax that is automatically triggered based on the metered rate in use.
Improvement_surcharge	0.30 improvement surcharge assessed trips at the flag drop. The improvement surcharge began being levied in 2015.
Tip_amount	This field is automatically populated for credit card tips.Cash tips are not included.
Tolls_amount	Total amount of all tolls paid on the trip.
Total_amount	The total amount charged to passengers. Does not include cash tips.

Fig 4: Dataset Sub-features (Continued)

Evaluation:

In order to evaluate the performance of our model, we split the data into a training set (80% of data set) and testing set (20% of data set) , where the training examples are all ordered chronologically before the testing examples. This configuration mimics the task of predicting future numbers of taxi pickups using only past data. We chose MSE to evaluate our prediction because it favors consistency and heavily penalizes predictions with a high deviation from the true number of pickups. From the point of view of a taxi dispatcher, any large mistake in gauging taxi demand for a particular tpep provider could be costly – imagine sending 600 taxis to a tpep_pickup location that only truly requires 400. This misallocation results in many unutilized taxis crowded in the same place, and should be penalized more heavily than dispatching 6 taxis to a zone that only requires 4, or even dispatching 6 taxis to 100 different zones that only require 4 taxis each. MSE most heavily penalizes such large misallocations and best represents the quality of our models' predictions.

Comparing the results between our different models, we also report the MAPE value (coefficient of determination) in order to evaluate how well the models perform relative to the variance of the data set.

	Model	MAPE(%)	MSE
0	Simple Moving Average Ratios	19.582447	1177.280010
1	Simple Moving Average Predictions	13.426683	311.275954
2	Weighted Moving Average Ratios	19.162557	1072.040084
3	Weighted Moving Average Predictions	13.163834	293.489068
4	Exponential Weighted Moving Average Ratios	21.776898	1817.878181
5	Exponential Weighted Moving Average Predictions	16.235122	436.496180

	Model	Mean_Absolute_Per_Error(%)
0	Simple Moving Average Ratios	19.582447
1	Simple Moving Average Predictions	13.426683
2	Weighted Moving Average Ratios	19.162557
3	Weighted Moving Average Predictions	13.163834
4	Exponential Weighted Moving Average Ratios	21.776898
5	Exponential Weighted Moving Average Predictions	16.235122
6	Linear Regression	16.252737
7	XGBoost Regressor	13.112175

Fig 5: MAPE & MSE of various models

METHODOLOGY

A. Dataset Features

We collected two datasets to conduct our experiments, as used by Zhao et al. (2017) [6]. The first data set comes from the NYC Trip Record Data. This public dataset contains: date, time, pickup location and dropoff of passengers, distance of travel, fares, types of payment and number of passengers. These data were provided by the New York Taxi and Limousine Commission by technology providers authorized by the Taxicab and Livery Passenger Enhancement program [19]. Each dataset record was generated by GPS devices installed in each taxi, a total of 13,237 yellow taxis in New York City made 13,813,031 passengers pickups in July 2014. Among the 18 attributes present in this data set, the main ones are; the driver id code, date and time of pickup, date and time of dropoff, travel distance in kilometers, latitude of pickup and dropoff. The second dataset (NYC PLUTO — Primary Land Use Tax Lot Output) is the set of geographic data, showing information about each lot, measurements of each building within a given lot and information about address, owners, use and geographic location expressed in the New York-Long Island coordinate system. This dataset is made available online by the New York City Department of Planning (NYC Department of City Planning).

B. Geographic Data pre-processing

The geographic data of the New York districts, provided by the New York City Planning Department, was treated with algorithms in Python, used for manipulating files .shp - shape format and translating geographic coordinates. Fig. 1 shows a picture that is a vector image of the geographic maps of the Manhattan district, generated during the datapoint mapping. In order to map each boarding point and the tuple (block, district) corresponding to that point, an algorithm in Python was developed and used. The set of boarding points and the geographic set were loaded into dataframes, data structures from the Pandas library. A p pickup point is associated with a b block when the distance between p and b is less than the distance between p and any other block b1, b2, b3, ..., bn.

C. Data Cleaning

- Here, we will do the uni variate analysis of Jan-2015 data and remove all the erroneous/outlier points.

- Latitude and Longitude Data:
  It is inferred from [6] that New York is bounded by the location coordinates(latitude, longitude) — (40.5774, -74.15) & (40.9176,-73.7004) hence any coordinates — including both pickups and drop-offs — not within these coordinates are not considered by us as we are only concerned with pickups which originate within New York.

- Trip Duration:
  Trip Duration = Drop-off time — pickup time
  According to NYC Taxi & Limousine Commission Regulations the maximum allowed trip duration in a 24 hour interval is 12 hours. Therefore, we have removed all the points where trip duration is more than 12 hrs. This will remove all the erroneous and outlier points.

- Speed:
  We have computed speed as trip distance divided by total time taken to complete that trip. So, here we are taking average speed of a trip and checked for outlier points, where speed is in either negative or unusually very high.

- Trip Distance:
  We have just checked the trip distance and checked for outlier points in our data. We removed all the points where trip distance is either too high or in negative.

- **Fare:** We checked the fares and checked for outlier points. Here, also we removed all the points where trip fare is either extremely high or in negative

- **Removing Outliers & Erroneous Points:** Finally, we removed all the data points where pickups and drop-offs are outside of New York City area.

Fraction of points left after removing all the erroneous points and outlier points. Points where pickups and dropoffs are outside of NYC are also removed.

```
In [49]: print("Fraction of cleaned points",str(new_frame_cleaned.shape[0]/new_frame.shape[0]))

         Fraction of cleaned points 0.9680387130396095

In [50]: print("Total number of outliers and erroneous points removed = ",str(new_frame.shape[0] - new_frame_cleaned.shape[0]))

         Total number of outliers and erroneous points removed =  407474
```

Fig 6: Code snippet for removing outliers

CLUSTERING & SEGMENTATION

Here, we have divided whole NYC into regions using "**K-Means Clustering**". Here, big questions comes that what will be the optimal number of clusters. Now K- Means has a property that it creates clusters roughly of same size. Here, size refers to the number of points and not the area. Now, the Manhattan area of NYC has large number of pick up, so cluster size in Manhattan area will be small as compared to outskirts areas of NYC where cluster size will be large.

```
On choosing a cluster size of 10
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 2.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 7.0
Minimum distance between any two clusters = 0.8686191611422962
----------
On choosing a cluster size of 20
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 5.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 15.0
Minimum distance between any two clusters = 0.5854838295999788
----------
On choosing a cluster size of 30
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 8.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 22.0
Minimum distance between any two clusters = 0.4819846357949098
----------
On choosing a cluster size of 40
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 9.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 31.0
Minimum distance between any two clusters = 0.44246037062914323
----------
On choosing a cluster size of 50
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 13.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 37.0
Minimum distance between any two clusters = 0.34059618603664527
----------
On choosing a cluster size of 60
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 15.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 45.0
Minimum distance between any two clusters = 0.33397358852890063
----------
On choosing a cluster size of 70
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 19.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 51.0
Minimum distance between any two clusters = 0.30103962388132
----------
On choosing a cluster size of 80
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 17.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 63.0
Minimum distance between any two clusters = 0.2807272254597499
----------
On choosing a cluster size of 90
Avg. Number clusters within vicinity where inter cluster distance < 2 miles is 28.0
Avg. Number clusters outside of vicinity where inter cluster distance > 2 miles is 62.0
Minimum distance between any two clusters = 0.18347249496811938
----------
Time taken = 0:01:28.398460
```

Check Deployment: https://taxi-demand.herokuapp.com/

We want the minimum inter cluster distance between any two clusters to be at least 0.5 miles and when the number of clusters are 30 then this condition is almost meeting. Therefore, we are considering number of clusters to be 30.

Fig 7: Code snippet-choosing optimal cluster size

We got following points as cluster centers:

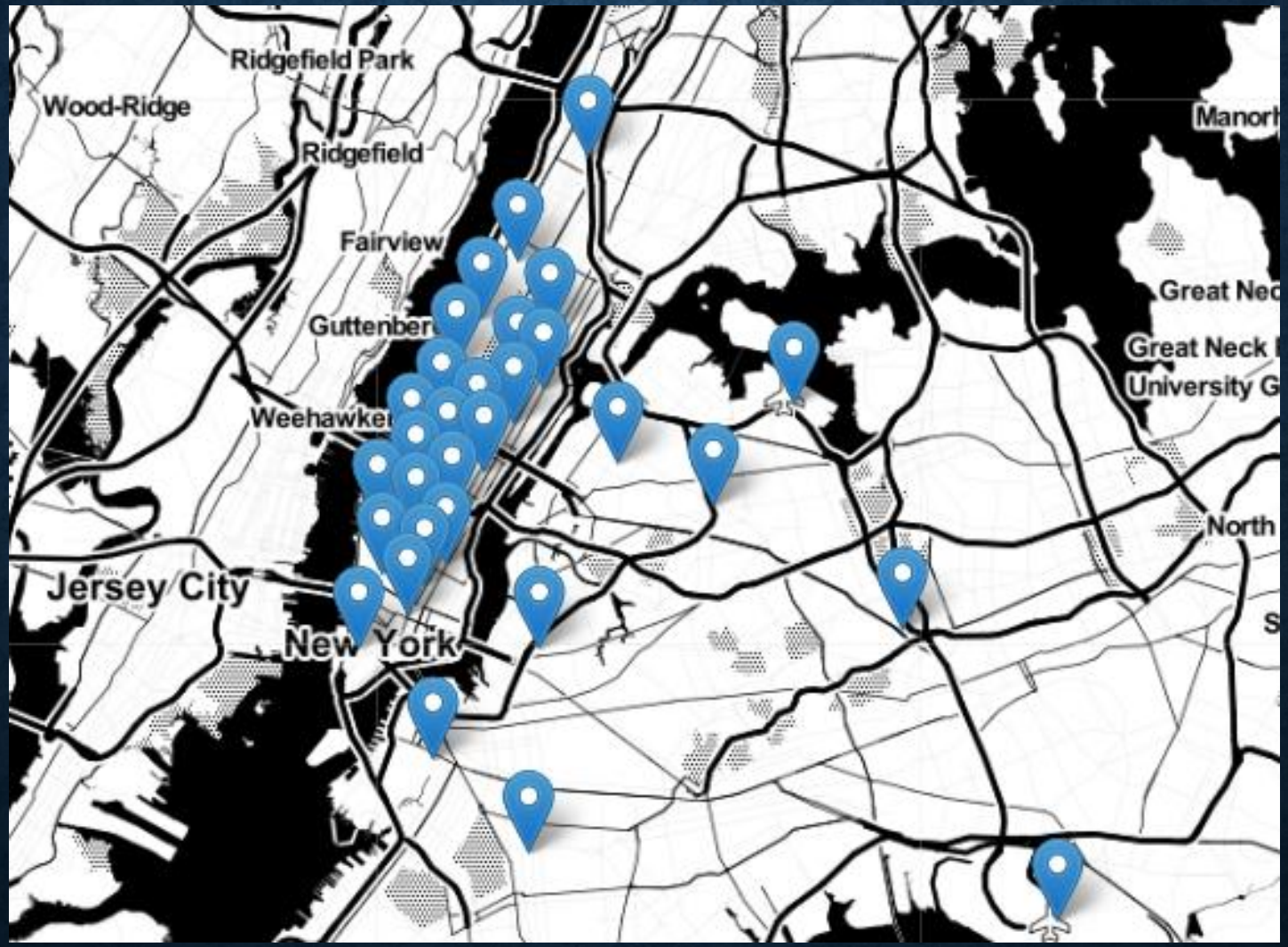


Fig 8: Cluster centers

TIME BINNING

- Now here we have taken time in regular format and converted that time into Unix time stamp then divided that time by 600 to make 10 min bins. For January 2015 data, each time bin is a 10 minute time interval which is equal to the time elapsed — in seconds — since midnight Jan 2015, divided by 600 (so as to make it in 10minute bin). Therefore, there will be a total of 4464, 10 minute time bins in the month of January-2015.

- Now we have both cluster-ID/region-ID and 10min time bins. Now with any region-ID and 10 min time bin. We have to predict the number of pickups.

- Now we will take all of the Jan 2015 and Jan 2016 data and convert all other data into cluster-ID and 10 min time bin.

DATA PREPARATION FOR JANUARY 2016 DATA.

Now doing the same operations for the month of Jan 2016.

1. Get the dataframe which includes only required colums.
2. Add trip_duration, speed, unix time stamp of pickup_time.
3. Remove the outliers based on trip_duration, speed, trip_distance, total_amount.
4. Remove all the points where pickup and dropoff are outside of New York City area.
5. Add pickup_cluster to each data point.
6. Add time_bin (index of 10min intravel to which that trip belongs to).
7. Group by data, based on ‘pickup_cluster’ and ‘time_bin’

*Now in January-2016, there are 31 days. Each day has 144, 10-min time bins. So, in whole month of January 2016, there will be 31*144 = 4464, 10-min time bins.*

```
PREPARATION OF JANUARY 2016 DATA.
-------------------------------------
Number of columns = 19
-------------------------------------
Time taken for creation of dataframe is 0:04:49.926549
New Frame for Jan 2016 creation done
-------------------------------------
Trip Duration Outliers removed
-------------------------------------
Speed Outliers removed
-------------------------------------
Trip Distance Outliers removed
-------------------------------------
Total Amount Outliers removed
-------------------------------------
Pickups outside of NYC are removed
-------------------------------------
Dropoffs outside of NYC are removed
-------------------------------------
Pickup Clusters are assigned
-------------------------------------
Pickup time bins are assigned
-------------------------------------
Pickup cluster and time bins are grouped.
-------------------------------------
Done...
-------------------------------------
Fraction of Total data left = 0.9702401919966318
Total Number of outliers removed = 324586
-------------------------------------
Total Time taken for execution of Jan 2016 data = 0:05:07.846681
-------------------------------------
```

Fig 9: Output snippet after data preparation

SMOOTHING TIME-SERIES DATA

- Now since we have divided our whole data in 10 minutes interval. Now if any of the 10 min interval contains zero pickups then it will lead to divide by zero problem later. Hence, we are adding few pickups from neighboring bins to the bin which contains zero pickups in order to make all the neighboring bin pickups equal. This is how we are smoothing our data.
- We can fill the missing values in following ways:

1. Fill the missing value with 0's
2. Fill the missing values with the avg values

- Case 1:(values missing at the start)
  Ex1: _ _ _ x =>ceil(x/4), ceil(x/4), ceil(x/4), ceil(x/4)
  Ex2: _ _ x => ceil(x/3), ceil(x/3), ceil(x/3)
- Case 2:(values missing in middle)
  Ex1: x _ _ y => ceil((x+y)/4), ceil((x+y)/4), ceil((x+y)/4), ceil((x+y)/4)
  Ex2: x _ _ _ y => ceil((x+y)/5), ceil((x+y)/5), ceil((x+y)/5), ceil((x+y)/5), ceil((x+y)/5)
- Case 3:(values missing at the end)
  Ex1: x _ _ _ => ceil(x/4), ceil(x/4), ceil(x/4), ceil(x/4)
  Ex2: x _ => ceil(x/2), ceil(x/2)

So we use smoothing for Jan 2015 data since it acts as our training data and we filled missing values with zero for Jan 2016 data.

RATIOS AND PREVIOUS TIME BIN VALUES FOR BASELINE MODELS

- We can make two types of **baseline models**, one using **ratios** and other using **previous time-bins** to predict for future time bins**.** In **ratios method** we have used both 2015 and 2016 data. In **previous time-bins method** we have used only 2016 data.

- **Ratios:** Now, let say cluster-1: R_t= Pickup_t_2016/ Pickup_t_2015. "R_t" is the ratio of the pickup of 2016 to pickup of 2015 at time 't'. Similarly R_t+1 = Pickup_t+1_2016/ Pickup_t+1_2015. Now if we have values R_t+1 and Pickup_t+1_2015 from 2015 which is our training data, then we can calculate Pickup_t+1_2016 = R_t+1 * Pickup_t+1_2015.

- **Previous Time Bins**: Now in real-production, for time series data, we can typically use our data till time 't-1' to make predictions for time 't'. As taxi drivers in New York City are using 4G internet, so they can immediately send us data till time "t-1". Then somehow we can come up with function 'f', to make predictions for time "t". Like: Pickup_t = f(Pickup_t-1, Pickup_t-2, Pickup_t-3…)

SIMPLE MOVING AVERAGE

- **Ratios:** Now as we have discussed in previous video that we will calculate Pickup_t+1_2016 as Pickup_t+1_2016 = R_t+1 * Pickup_t+1_2015. Here we already have Pickup_t+1_2015 value from training data but we do not have R_t+1 value. So, we will calculate R_t+1 as R_t+1 = (R_t + R_t−1 + R_t-2….R_t-n)/n.

- **Previous Time Bins:** Now, for pickup_t+1 which is pickup at time ‘t+1’, we can built a function as Pickup_t+1 = (Pickup_t + Pickup_t-1 + Pickup_t-2 + Pickup_t-n….)/n” for 2016 data only.

WEIGHTED MOVING AVERAGE

- Weighted moving average is a slight variation over simple moving average. here. For both **ratios** and **previous time-bins**, we have given more weight age to the values of most recent times and subsequently we have reduced our weights to the far away values.

- **Ratios**: $R_{t+1} = ((n)*R_t + (n-1)*R_{t-1} + (n-2)*R_{t-2}....R_{t-n})/(n*(n+1))/2$.

- **Previous Time Bins:** $Pickup_{t+1} = ((n)*Pickup_t + (n-1)*Pickup_{t-1} + (n-2)*Pickup_{t-2} + Pickup_{t-n}....)/(n*(n+1))/2$.

- Here, “n” is a hyper- parameter.

EXPONENTIAL WEIGHTED MOVING AVERAGE

- Through weighted averaged we have satisfied the analogy of giving higher weights to the latest value and decreasing weights to the subsequent ones but we still do not know which is the correct weighting scheme as there are infinitely many possibilities in which we can assign weights in a non-increasing order and tune the hyper-parameter window-size. To simplify this process we use Exponential Moving Averages which is a more logical way towards assigning weights and at the same time also using an optimal window-size.

- In exponential moving averages we use a single hyper-parameter alpha (α) which is a value between 0 & 1 and based on the value of the hyper-parameter alpha the weights and the window sizes are configured.

$R't = \alpha * Rt-1 + (1-\alpha) * R't-1$

$R't$ is the current predicted ratio.

$R't-1$ is the previous predicted ratio.

$Rt-1$ is the actual previous ratio.

Let say alpha = 0.7.

Now, when alpha = 0.7, then it means we are giving 70% weightage to the previous predicted ratio and 30% weightage to the previous actual ratio.

$R'0R0' = 0$

$R'1 = 0.7 * R'0 + 0.3 * R0$

$R'2 = 0.7 * R'1 + 0.3 * R1$

$R'3 = 0.7 * R'2 + 0.3 * R2$

Let's take $R'3$.

$R'3 = 0.3 * R2 + 0.7 * R'2$

$R'3 = 0.3 * R2 + 0.7 * (0.3 * R1 + 0.7 * R'1)$

$R'3 = 0.3 * R2 + 0.7 * (0.3 * R1 + 0.7 * (0.3 * R0 + 0.7 * R'0))$

$R'3 = 0.3 * R3 + 0.7 * 0.3 * R1 + 0.7 * 0.7 * 0.3 * R0 + 0.7 * 0.7 * 0.7 * R'0$

$R'3 = 0.3 * R3 + 0.7 * 0.3 * R1 + 0.7 * 0.7 * 0.3 * R0 + 0$

Previous Time Bins: Pickup_t = alpha∗Pickup_t−1+(1−alpha)∗Predicted_Pickup_t-1.

Fig 10: Computing Exponential Weighted Moving Average

BASELINE MODEL RESULT

	Model	MAPE(%)	MSE
0	Simple Moving Average Ratios	20.047669	987.182131
1	Simple Moving Average Predictions	13.688769	375.552040
2	Weighted Moving Average Ratios	19.584620	919.250478
3	Weighted Moving Average Predictions	13.402147	357.835865
4	Exponential Weighted Moving Average Ratios	22.422470	1317.632826
5	Exponential Weighted Moving Average Predictions	16.506108	495.816469

Fig 11: Baseline Model Results with MAPE & MSE

REGRESSION MODEL

- We will apply three regression models namely: Linear regression, Random Forest Regression and XGBoost Regression.
- We will be using only January 2016 data.
- But before feeding the data to the models, we have to prepare the features. We have prepared a total of 19 features in the following way:
- 1. From baseline models we have observed that the previous time-bins are quite effective in predicting the pickup for next time bin in the same cluster. Therefore, we have decided to include previous five time-bin pickups that happen in the same cluster for the prediction of pickup happen in the next time bin in the same cluster.
- 2. The sixth and seventh feature will be the latitude and longitude of the cluster center.
- 3. The eighth feature will be the day of the week on which pickup happen.
- 4. We have taken the ninth feature as the prediction result of the “Weighted Moving Average Predictions”, because we have observed that in all of the baseline models, “Weighted Moving Average Predictions” are the best in predicting the next pickup.
- 5. Rest 10 features will be the Fourier Features.

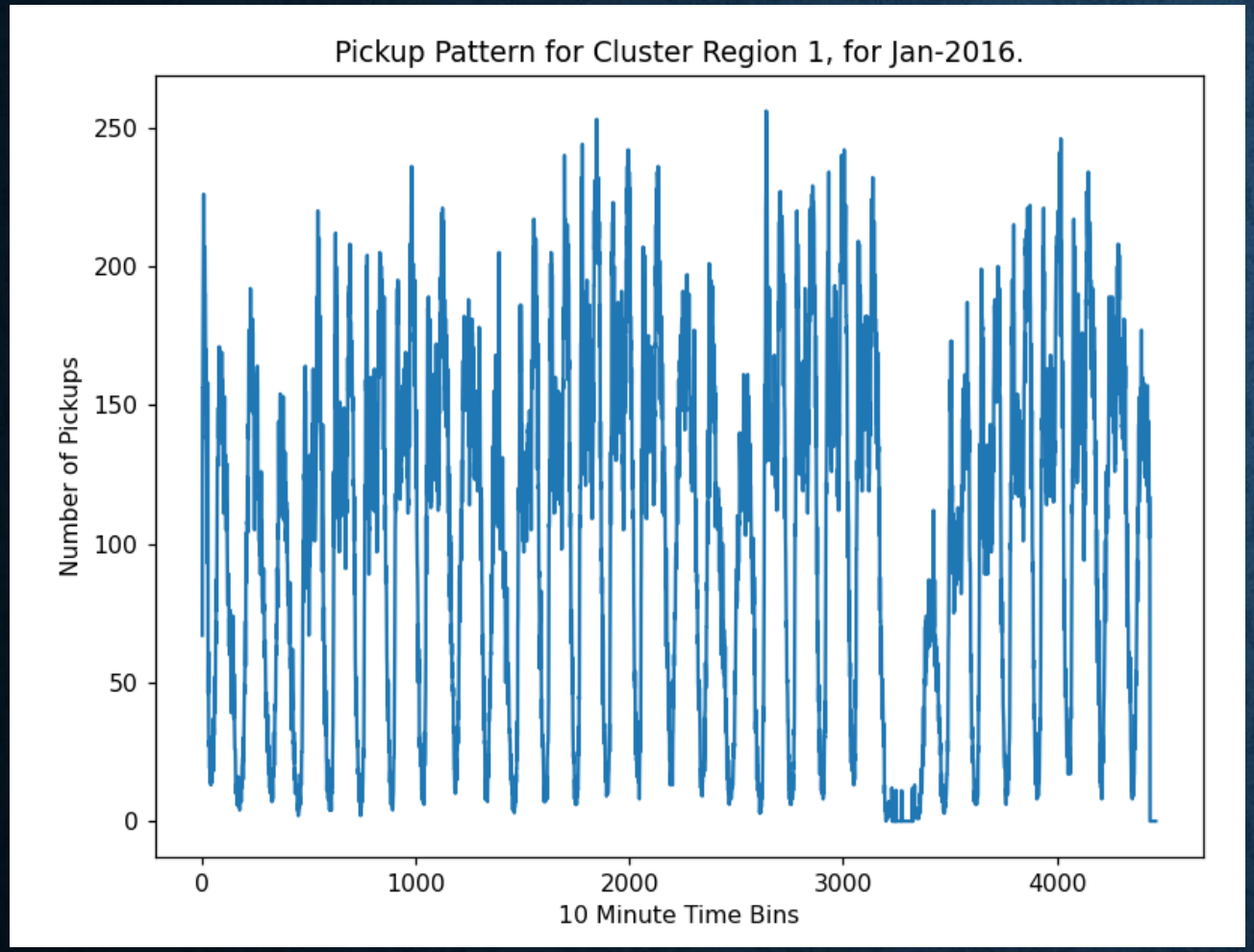
ADDING FOURIER TRANSFORM FEATURES

- Fourier Transform says that whenever we have a repeating pattern in a wave like here we have a repeating pattern of pickups in 24 hrs of time period, this repeating wave can be decomposed into sum of multiple sine waves. Each sine wave will have some frequency and amplitude. Now we can represent our original wave from time-domain to frequency-domain, where frequency will be presented on x-axis and amplitude on y-axis. In frequency domain, the x-axis frequencies will be discrete frequencies of individual sine waves and y-axis amplitudes will be their corresponding amplitude values.

- **In time-series data whenever we have a repeating pattern then the Fourier decomposed frequencies and their amplitude can be added as a features in our data.** As these features are very useful particularly when we have a repeating pattern in a time-series data.

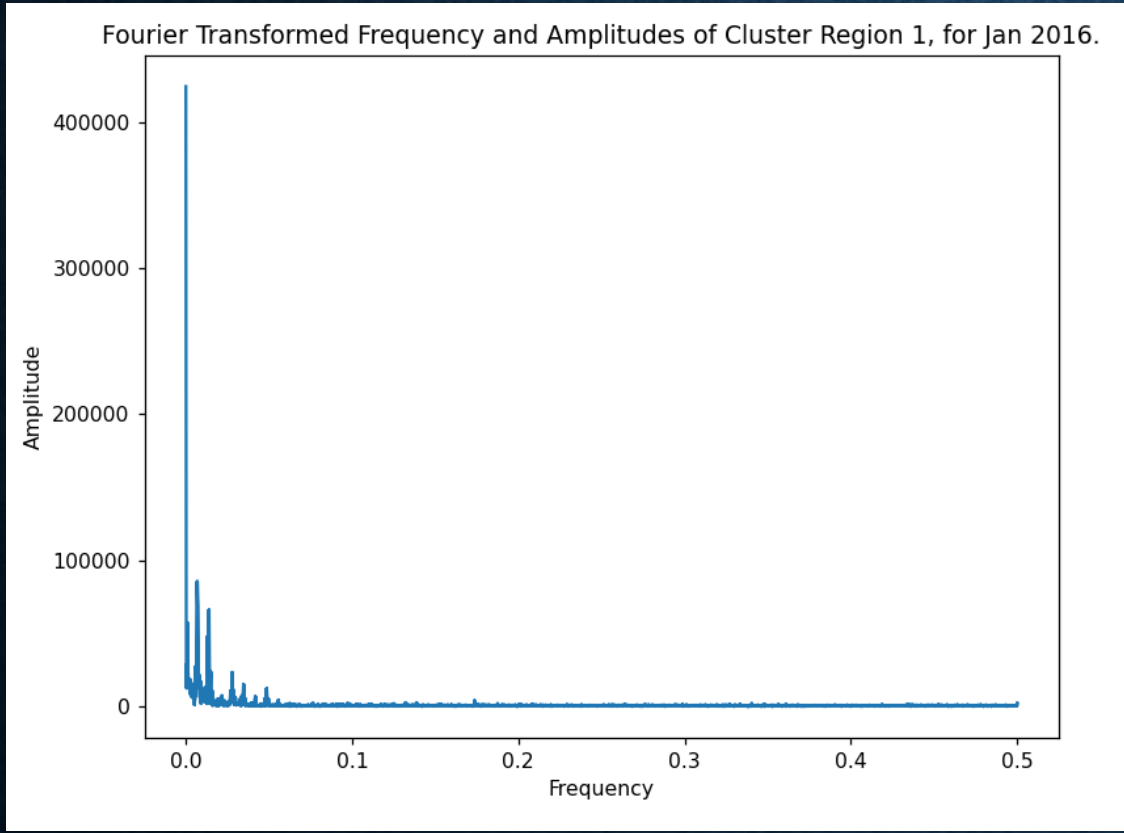
- Intuitively, amplitude measures how significant one frequency is as compared to other frequency in a relative sense. More the amplitude, more of the time-series that sine-wave can represent. So, we can simply find the frequencies with maximal amplitudes by simple sorting them to obtain the important frequencies and their corresponding amplitudes. Finally we can add them as a feature in our data.

- Now we have a total of 30 clusters in our data. In each individual cluster the pickup pattern(which is repeating) will remain same. Note that pickup pattern in different clusters will be different. So, when we plot Fourier transform of our original wave for one cluster, and let say that one cluster contains ’n’ points, and if we take top 3 frequencies and corresponding to highest amplitude values as a feature, then for all of those ’n’ points, these 3 frequencies and their corresponding amplitudes will remain same. It will change only for the points which are in another cluster because the repeating pattern in another cluster will be different.



- the pickup pattern will remain same in 24 hrs time-period. Then it repeats again in next 24 hrs. So, it is a repeating wave.
- this repeating wave can be decomposed into sum of multiple sine waves
- we have applied Fourier transform over this wave and below is the graph for the Frequency-Amplitude of the Fourier transformed wave for cluster 1

Fig 12: Pickup Pattern for Cluster Region 1



There is one peak at a frequency of 1/144 which is equivalent to 24 hrs. This peak is capturing full 24hrs pickups and since during rush hours the pickups are high so therefore peak of wave is also high at 1/144.

Another peak is at frequency of 1/72 which is equivalent to 12 hrs. Now this peak is capturing information of the hush hours,

Fig 13: Fourier Transformed Frequency & Amplitudes of cluster region 1

Total 19 features after adding the 10 fourier transform features

1. **f_t_1:** Number of pickups that are happened previous t-1st 10min interval
2. **f_t_2:** Number of pickups that are happened previous t-2nd 10min interval
3. **f_t_3:** Number of pickups that are happened previous t-3rd 10min interval
4. **f_t_4:** Number of pickups that are happened previous t-4th 10min interval
5. **f_t_5:** Number of pickups that are happened previous t-5th 10min interval
6. **Freq1:** Fourier Frequency corresponding to 1st highest amplitude
7. **Freq2:** Fourier Frequency corresponding to 2nd highest amplitude
8. **Freq3:** Fourier Frequency corresponding to 3rd highest amplitude
9. **Freq4:** Fourier Frequency corresponding to 4th highest amplitude
10. **Freq5:** Fourier Frequency corresponding to 5th highest amplitude
11. **Amp1:** Amplitude corresponding to 1st highest fourier transformed wave.
12. **Amp2:** Amplitude corresponding to 2nd highest fourier transformed wave.
13. **Amp3:** Amplitude corresponding to 3rd highest fourier transformed wave.
14. **Amp4:** Amplitude corresponding to 4th highest fourier transformed wave.
15. **Amp5:** Amplitude corresponding to 5th highest fourier transformed wave.
16. **Latitude:** Latitude of Cluster center.
17. **Longitude:** Longitude of Cluster Center.
18. **WeekDay:** Day of week of pickup.
19. **WeightedAvg:**: Weighted Moving Average Prediction values.

Fig 14: Fourier transformed features

REGRESSION MODELS(CONT.)

- **Linear Regression**-Here, we have applied linear regression and predicted the pickups for both train and test data then finally we calculated Mean Absolute Percentage Error.

- **Random Forest Regressor**-Here, we have applied Random Forest Regressor and predicted the pickups for both train and test data then finally we calculated Mean Absolute Percentage Error.

- **XGBoost Regressor**-Here, we have applied XGBoost Regressor and predicted the pickups for both train and test data then finally we calculated Mean Absolute Percentage Error.

MODEL COMPARISON

	Model	TrainMAPE(%)	TrainMSE	TestMAPE(%)	TestMSE
0	Linear Regression	13.697816	383.580467	18.577050	363.939513
1	Random Forest Regression	5.127255	66.774265	14.011983	292.947746
2	XGBoost Regressor	13.161901	312.470746	13.589307	240.435556

- The difference between train error and test error of Random Forest Regressor is high, which clearly shows that Random Forest Regressor is overfitting. Therefore, we are discarding Random Forest Regressor.

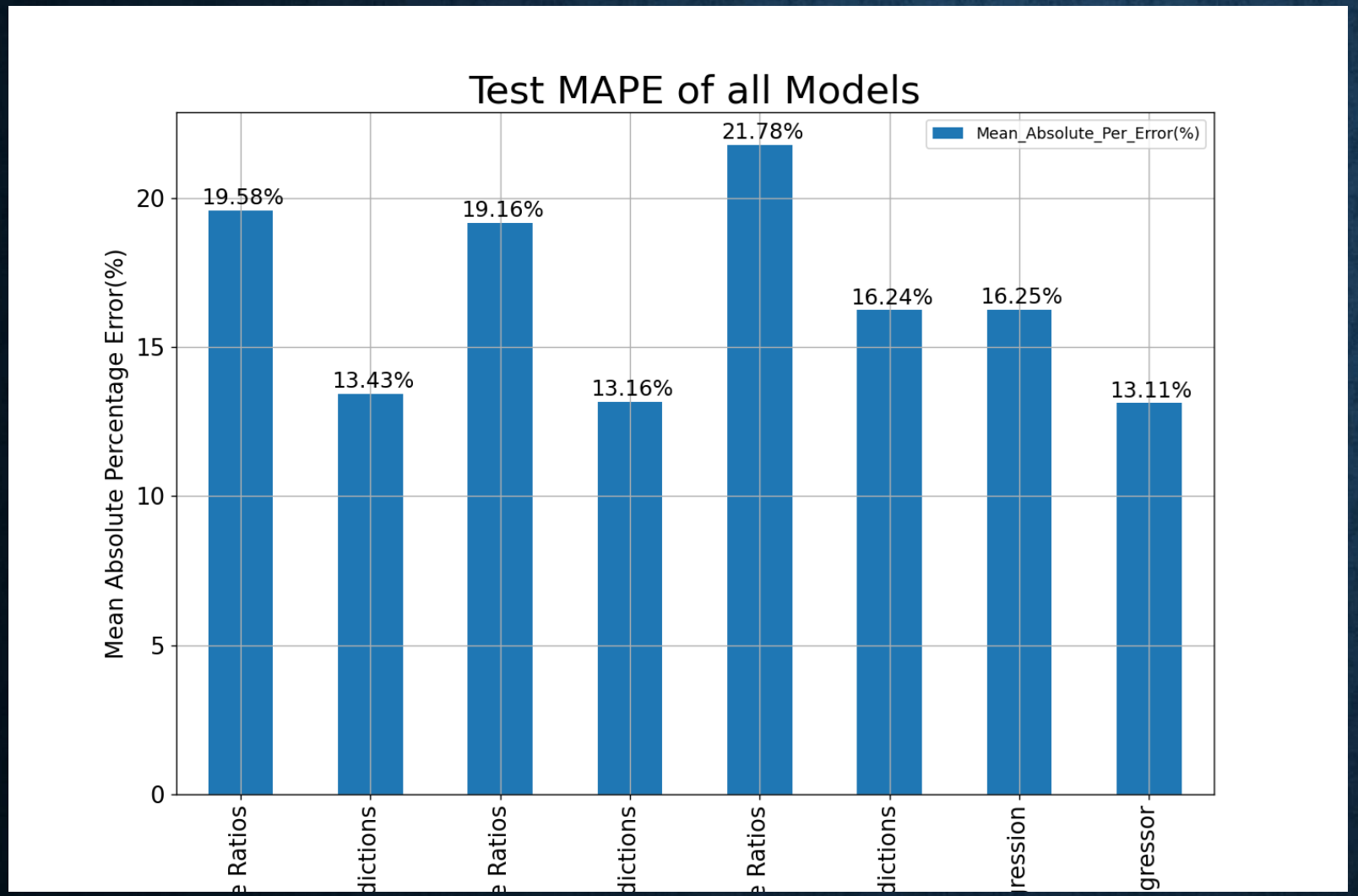
- The best model with lowest train and test MAPE error is XGBoost Regressor.

Fig 15: Model comparison between training & testing data

COMPARING TEST MAPE OF ALL THE MODELS

	Model	Mean_Absolute_Per_Error(%)
0	Simple Moving Average Ratios	19.582447
1	Simple Moving Average Predictions	13.426683
2	Weighted Moving Average Ratios	19.162557
3	Weighted Moving Average Predictions	13.163834
4	Exponential Weighted Moving Average Ratios	21.776898
5	Exponential Weighted Moving Average Predictions	16.235122
6	Linear Regression	16.252737
7	XGBoost Regressor	13.112175

Fig 16: Comparing test MAPE of all models



The data set contains dependent or target variables along with independent variables. We build models using independent variables and predict dependent or target variables. If the dependent variable is numeric, regression models are used to predict it. MSE is used to evaluate the models.

We also calculated the Mean Absolute per Error (MAPE), the sum of the individual absolute errors divided by the demand (each period separately). It is the average of the percentage errors, despite being a poor-accuracy indicator. As you can see in the formula, MAPE divides each error individually by the demand, so it is skewed: high errors during low-demand periods will significantly impact MAPE. Due to this, optimizing MAPE will result in a strange forecast that will most likely undershoot the demand.

Efficiency in terms of Time and Space

With the authentic features of this program, which operates by analyzing large amounts of verifiable data, the most important test is intensity. To reduce execution time, time complexity and house complexity should be fully assessed. There are several angles that can significantly increase the product's power: the code's norm - The equation's understanding, which is a well-practiced methodology - large amounts of data and data preparation software - data style - network capability - server capacity

RELIABILITY

This is one of the most important features that might represent the decision moment quality for any item, particularly in this practical forecasting application. In reality, in order to establish a reputation, the machine must be trustworthy from the start. The following are some examples of endowments that might be recorded:

- Adaptability (simple to grow with no outcome on the current worker) - portability (ensured to constantly be accessible)
- Safety is paramount (ready to stay away from network assault)
- Repairs and assistance (The defects in the equipment will be discarded by the organization)
- Instrument for quick failover (if the worker is down, it will rapidly be communicated to an alternate host)
- Monetary assessment (numerous proposals for clients from most help providers, especially inside the improvement interaction)

USABILITY

A human-made object's usability refers to however straightforward it's to use and learn. however, as a result of it's going to be outlined by the organization, it's AN abstract notion. Clients may make a few cash mentioned aims with viability, effectiveness, and fulfillment in an extremely declared setting of use in terms of a product package, for example.

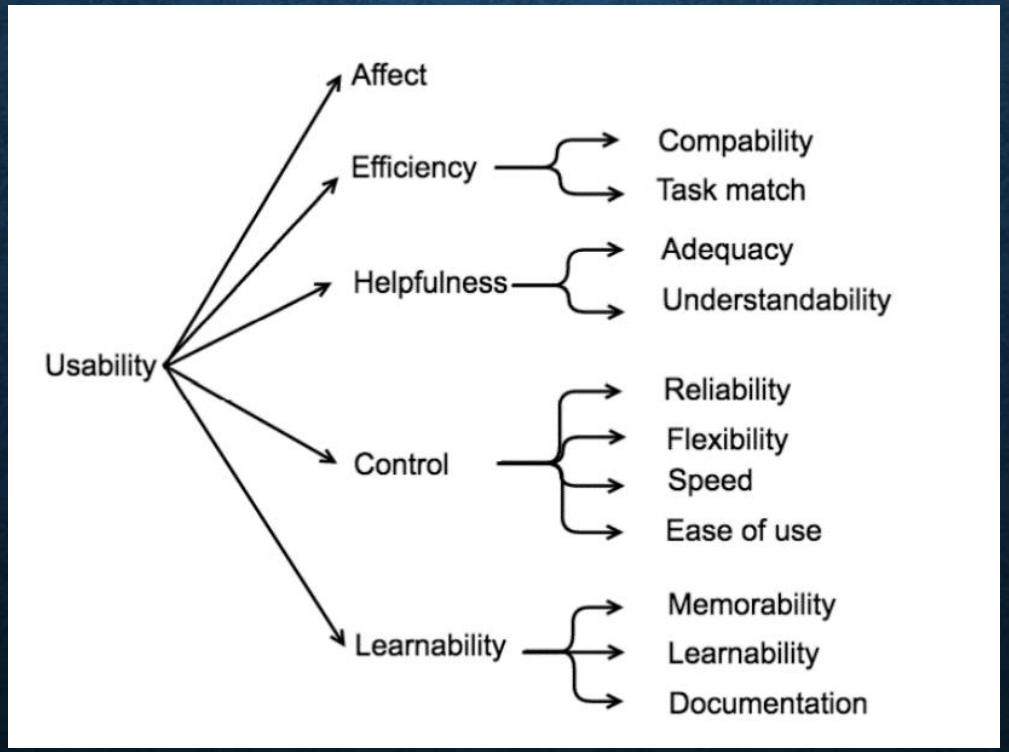


Fig 15: Reliability of system

DEPENDABILITY

Informally, dependableness refers to what quantity users will have confidence in the system's qualities. dependableness, on the opposite hand, contains qualities like dependableness, safety, security, and handiness. the subsequent diagram depicts the fundamental qualities of dependability:

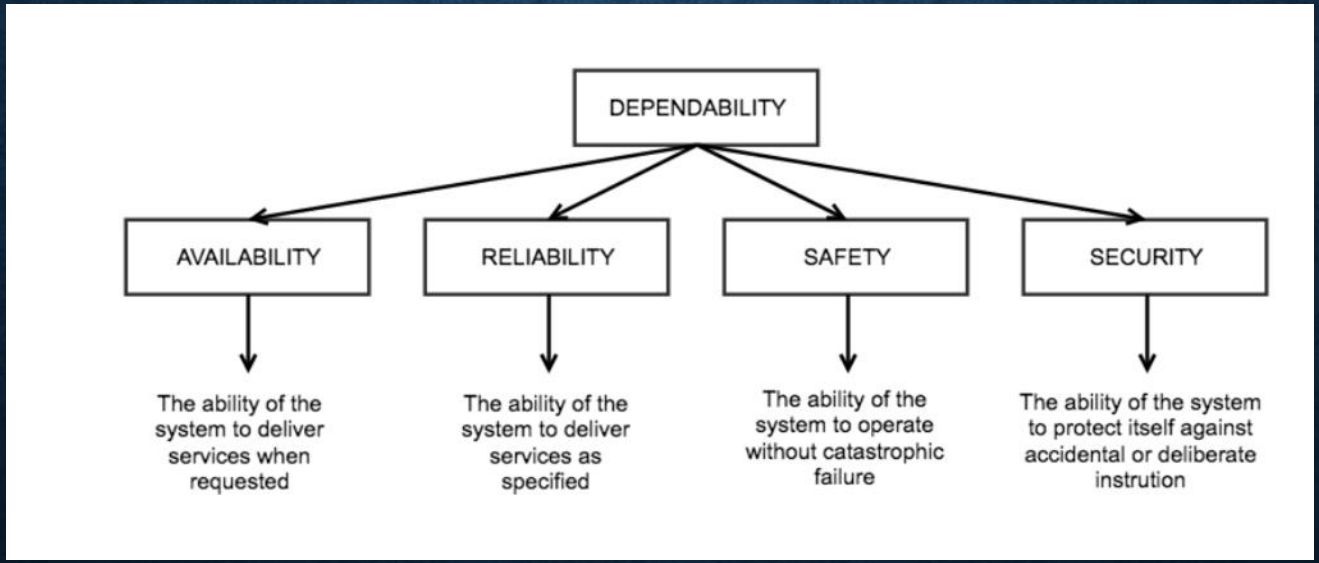


Fig 16: Dependability of system

SYSTEM REQUIREMENTS

Hardware Requirements

- Laptop with sufficient computational power i.e. CORE i7
- Laptop with enough memory i.e. at least 16GB RAM
- Windows Operating System

Software Requirements

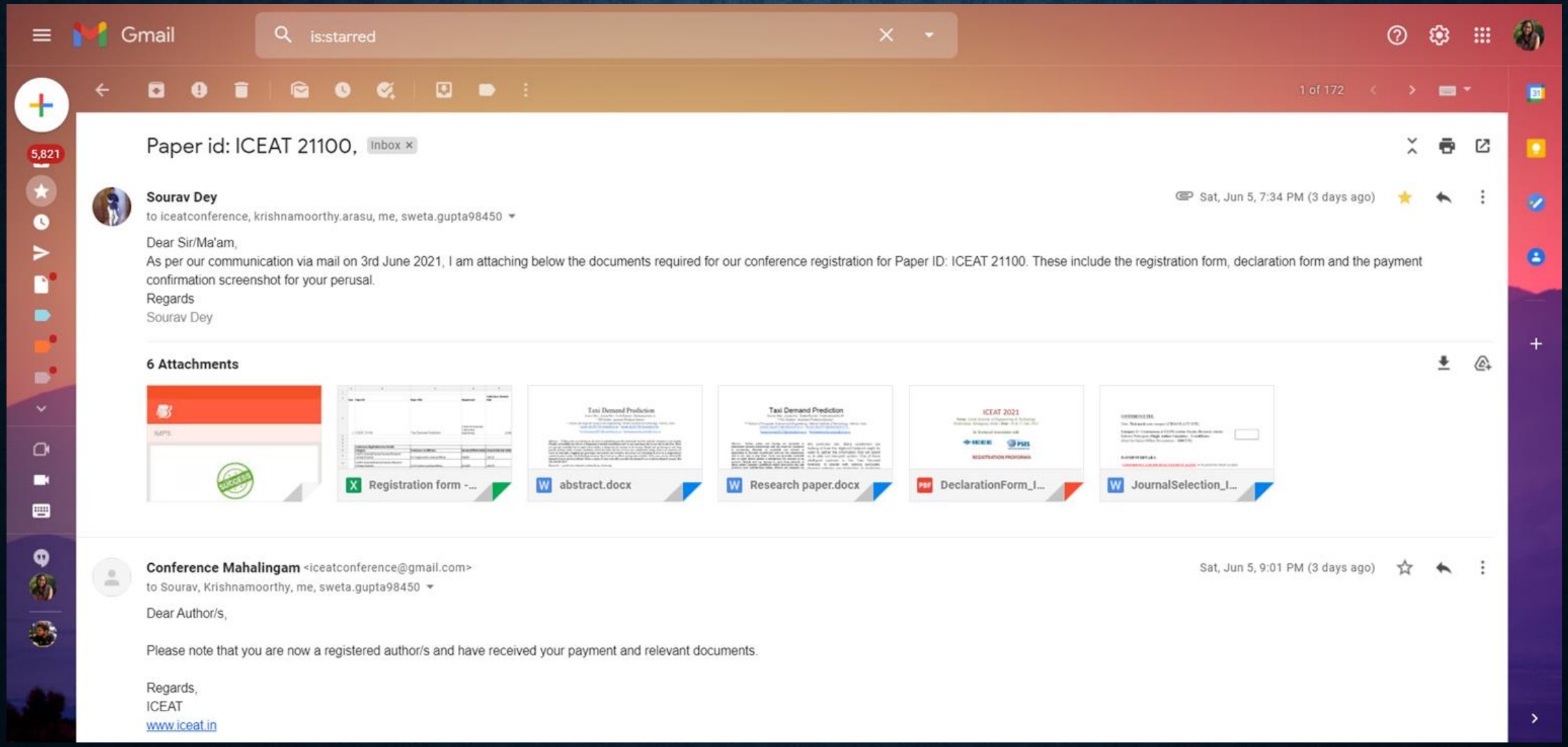
- ***Python 3.7.5****:* Python is a programming language which supports structured programming and object oriented programming. Python 3.0 was released in 2008 as a successor for Python 2.0 which was discontinued in 2020. Python 3.0 isn't backwards compatible.
- ***Anaconda3 1.7.0****:* It is an open source distributor for R and Python Programming language. It is used for scientific computing like Machine Learning, Data Science, Predictive Analysis, etc. It helps in simplifying packet deployment and management.
- ***Jupyter Notebook****:* The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. It is used for scientific computing like Machine Learning, Data Science, Predictive Analysis, etc.

CONCLUSION

By implementing our model at https://taxi-demand.herokuapp.com/ , we will be able to get insights into the data to help make effective taxi demand forecasting decisions. This study uses Clustering, XGBoost, Random Forest, Linear Regression, Simple Moving Average, Exponential Weighted Moving Average, and Weighted Moving Average to evaluate taxes and regression models and compare them to one other.

The weighted motion average technique is the smallest loss with a MAPE of 12.957807. This method has a greater accuracy than other regression-based algorithms. Future research can include a comparison of other more modern, yet less sophisticated, algorithms.

PUBLICATION



REFERENCES

Weblinks:

[1] *https://machinelearningmastery.com*

[2] *https://towardsdatascience.com*

[3] *http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml*

[4] *https://stackoverflow.com/questions/31572487/fitting-data-vs-transforming-data-in-scikit-learn*

[5] *https://colah.github.io/posts/2015-08-Understanding-LSTMs/*

[6] *https://www.flickr.com/places/info/2459115*

Journals:

[1] Zhao, K., Khrushchev, D., Freire, J., Silva, C., & Vo, H. (2016, December). Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In 2016 ieee international conference on Big data (big data) (pp. 833-842). IEEE.

[2] Seow, K. T., Dang, N. H., & Lee, D. H. (2009). A collaborative multiagent taxi-dispatch system. IEEE Transactions on Automation Science and Engineering, 7(3), 607-616.

[3] Davis, N., Raina, G., & Jagannathan, K. (2016, November). A multi-level clustering approach for forecasting taxi travel demand. In 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC) (pp. 223-228). IEEE.

[4] Lee, D. H., Wang, H., Cheu, R. L., & Teo, S. H. (2004). Taxi dispatch system based on current demands and real-time traffic conditions. Transportation Research Record, 1882(1), 193-200.

[5] Gers, F. A., Eck, D., & Schmidhuber, J. (2002). Applying LSTM to time series predictable through time-window approaches. In Neural Nets WIRN Vietri-01 (pp. 193-200). Springer, London.

[6] Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013). Predicting taxi–passenger demand using streaming data. IEEE Transactions on Intelligent Transportation Systems, 14(3), 1393-1402.

[7] De Brébisson, A., Simon, É., Auvolat, A., Vincent, P., & Bengio, Y. (2015). Artificial neural networks applied to taxi destination prediction. arXiv preprint arXiv:1508.00021.

[8] Xu, J., Rahmatizadeh, R., Bölöni, L., & Turgut, D. (2017). Real-time prediction of taxi demand using recurrent neural networks. IEEE Transactions on Intelligent Transportation Systems, 19(8), 2572-2581.

[9] Agarwal, V. (2015). Research on data preprocessing and categorization technique for smartphone review analysis. International Journal of Computer Applications, 975, 8887.

[10] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215

[11] Miao, F., Han, S., Lin, S., Stankovic, J. A., Zhang, D., Munir, S., ... & Pappas, G. J. (2016). Taxi dispatch with real-time sensing data in metropolitan areas: A receding horizon control approach. IEEE Transactions on Automation Science and Engineering, 13(2), 463-478.

[12] Zhang, D., He, T., Lin, S., Munir, S., & Stankovic, J. A. (2016). Taxi-passenger-demand modeling based on big data from a roving sensor network. IEEE Transactions on Big Data, 3(3), 362-374.

[13] Larochelle, H., & Murray, I. (2011, June). The neural autoregressive distribution estimator. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (pp. 29-37). JMLR Workshop and Conference Proceedings.

[14] Chen, Y., Li, O., Sun, Y., & Li, F. (2018). Ensemble classification of data streams based on attribute reduction and a sliding window. Applied Sciences, 8(4), 620.

[15] Balan, R. K., Nguyen, K. X., & Jiang, L. (2011, June). Real-time trip information service for a large taxi fleet. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 99-112).

[16] Markou, I., Rodrigues, F., & Pereira, F. C. (2018, November). Real-Time Taxi Demand Prediction using data from the web. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 1664-1671). IEEE.

[17] Ma, J., Chan, J., Ristanoski, G., Rajasegarar, S., & Leckie, C. (2019). Bus travel time prediction with real-time traffic information. Transportation Research Part C: Emerging Technologies, 105, 536-549.

[18] Ishiguro, S., Kawasaki, S., & Fukazawa, Y. (2018, October). Taxi demand forecast using real-time population generated from cellular networks. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (pp. 1024-1032).

[19] Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013, September). On predicting the taxi-passenger demand: A real-time approach. In Portuguese Conference on Artificial Intelligence (pp. 54-65). Springer, Berlin, Heidelberg.

THANK YOU