Taxi Demand Prediction System Using Machine Learning

Kavita Agrawal, K. Maldanna, G. Nalin Raj
Department of Computer Science and Engineering
Chaitanya Bharathi Institute of Technology, Hyderabad, India

***Abstract*—** Imbalance of availability of taxis is a big problem in Big cities. Sometimes, customers need to wait too long for the taxis or sometimes taxi drivers can't find the customers easily. Such problems can be resolved by informed driving, which is a key feature for increasing sustainability for taxi companies. Our paper mainly deals with predicting the demand of cabs in specific areas using Machine Learning algorithms based on climate ,location ,time of day,and other parameters which affect the imbalance in waiting time for customers. We are allowing users to use our prediction model on a web based platform. The user needs to specify the input of various parameters asked by the model like climate,time of day,location etc and the model attempts to predict the number of cabs needed to be distributed to that specific location based on the past data.

Keywords— Linear regression, XGBoost, ARIMA

I. INTRODUCTION

The taxicabs of New York City are widely recognized icons of the city and come in two varieties: yellow and green. Exploiting an understanding of taxi supply and demand could increase the efficiency of the city's taxi system. In New York city, people use taxis at a frequency much higher than any other cities in the US. Instead of booking a taxi by phone one day ahead of time, New York taxi drivers pick up passengers on the street. The ability to predict taxi ridership could present valuable insights to city planners and taxi dispatchers in answering questions such as how to position cabs where they are most needed, how many taxis to dispatch, and how ridership varies over time. We are focusing on predicting the number of taxi pickups given a one-hour time window and a location within New York city. Taxi service is imbalanced. While in some area's passengers experience long waits for a taxi, in others, many taxis roam without passengers. This imbalance leads to profit loss for taxi companies, since vehicles are vacant even when there is demand. Besides, it reduces the level of the passenger satisfaction due to long wait times.The ability to predict taxi demand can help address the taxi-service imbalance problem. Knowledge of where a taxi should be traveling can bring benefits to both taxi drivers and companies: taxi drivers can drive to high taxi demand areas, and taxi companies (e.g. Uber) may reallocate their vehicles in advance to meet the passenger demand.

The taxi service is an important transportation mode in urban areas. Unlike other ride sharing services like Uber where users hire a ride in advance via Internet applications, taxicabs are usually requested by pedestrians in a more spontaneous manner, which makes taxi behavior much more unpredictable. Several solutions have been proposed from many different disciplines so as to improve the quality of service and the efficiency of urban taxi rides. In that sense, a foremost course of action within the mobility data mining field has focused on predicting the taxi demand in different areas within a city. This way, it is possible to inform taxi operators in advance and minimize the amount of time that these vehicles are empty.

Rest of paper is organized as follows: Section 2 describes the related work in this area. Section 3 describes proposed algorithm in details and section 4 and 5 describes implementation details and experimental results Finally a conclusion of the work is given in section 6.

II. RELATED WORKS

One early implementation was available on taxi demand prediction problems for Beijing city in China. They used deep neural networks to implement it. Initially they tried with different algorithms but finally they ended up with DMVST-Net model which has given an accuracy of around 87%.they have used a large-scale online taxi request dataset collected from Didi Chucking, which is one of the largest online car- hailing companies in China. The dataset contains taxi requests from 02/01/2017 to 03/26/2017 for the city of Guangzhou. There are 20×20 regions in their data.

Chen, T has used different machine learning models[5]. Initially they tried using time series analysis models which are markov prediction model and Arima model but it gave very less accuracy which was around 52 to 63 % they also tried using svr (support vector regressor),Random Forest Regressor, ensemble methods which also didn't perform too good. But finally they achieved good accuracy with a gradient boosted regression tree(GBRT). The regression problem is a generalization of the classification problem, in which the model returns a continuous-valued output, as opposed to an output from a finite set. In other words, a regression model estimates a continuous-valued multivariate function.SVMs solve binary classification problems by formulating them as convex optimization problems (Vapnik 1998). The optimization problem entails finding the maximum margin separating the hyperplane, while correctly classifying as many training points as possible. SVMs represent this optimal hyperplane with support vectors. The sparse solution and good generalization of the SVM lend themselves to adaptation to regression problems. SVM generalization to SVR is accomplished by introducing an e-insensitive region around the function, called the e-tube. This tube reformulates the optimization problem to find the tube that best approximates the continuous-valued function, while balancing model complexity and prediction error. More specifically, SVR is formulated as an optimization problem by first defining a convex e-insensitive loss function to be minimized and finding the flattest tube that contains most of the training instances. Hence, a multiobjective function is constructed from the loss function and the geometrical properties of the tube.Then, the convex optimization, which has a unique solution, is solved, using appropriate numerical optimization algorithms. The hyperplane is represented in terms of support vectors, which are training samples that lie outside the boundary of the tube. As in SVM, the support vectors in SVR are the most influential instances that affect the shape of the tube, and the training and test data are assumed to be independent and identically distributed (iid), drawn from the same fixed but unknown probability distribution function in a supervised-learning context.

Jun Xu worked on Prediction applications using past taxi data .There are few previous research works conducted on taxi demand prediction[6]. Zhang proposes a passenger hotspots recommendation system for taxi drivers. By analyzing the historical taxi data, they extract hot-spots in each time-step and assign a hotness score to each of them. This hotness score will be predicted in each time-step and combined with the driver's location would be recommended. Zhao.define a maximum predictability for the taxi demand at street blocks level. They show the real entropy of past taxi demand sequences which proves that taxi demand is highly predictable. They also implement three prediction algorithms to validate their maximum predictability theory.Moreira-Matias propose a framework consisting of three different prediction models. In each time-step, the predicted demand is a weighted ensemble of predictions from three models. The ensemble weights are updated with individual prediction performances of previous time-steps in a sliding-window. Their framework can make short term demand predictions for the 63 taxi stands in the city of Porto, Portugal. Davis uses time-series modeling to forecast taxi travel demand in the city of Bengaluru, India. This information can be given to the drivers in a mobile application so that they know where the demand is higher.

B. Gupta S found that Gradient Boosting Regression predicts better results than Random Forest Regression. However, Random Forest Regression works faster than Gradient Boosting Regression as it uses parallel processing for making trees. Moreover, with increase in training data size, prediction improves further[7]. Also, with increase in training data size, there is little variation in n_estimators and random_states parameters of Random Forest Regression for result improvement. In case of Gradient Boosting Regression with increase in training data size, there is no variation in random_states and max_depth, and n_estimators decrease. We intend to run our algorithms on the complete dataset to further analyze the outcomes. We also wish to implement it in the form of an android or Web application for the convenience of the passengers booking cabs. Cab operators can use this system to improve the efficiency of the electronic dispatch system of the taxis.

The research article [8] specifies the applicability of ensemble learning and trip matching for estimating the destination and trip time for taxis in real time using Haversine distance calculated by Kernel Regression which are used as features to estimate the final destination, combined with average speed, average acceleration, and shape complexity for trip time prediction. Another work [9] partitions input into different clusters using K-Means and membership degrees to the cluster centers which are measured by "Gaussian fuzzy membership function." Authors in their work [10] tried to make passengers pick up profit by assigning a score to each road segment that determines whether it should be picked in the cruising route or not. This score depends on several factors; probability of finding a passenger on a road segment, length of the occupied paths originating from the road segment, etc. Research work took a large number of trips, and a probabilistic model is devised to detect parking places for taxis. There are a lot of techniques used in taxi demand Prediction. Some machine learning models are able to predict effectively the number of cabs required per location in a particular time based on different parameters like location id, weather conditions and some special events[1].

- **Historical average (HA)**: Historical average predicts the demand using average values of previous demands at the location given in the same relative time interval (i.e., the same time of the day).

- **Autoregressive integrated moving average (ARIMA)**: ARIMA is a well-known model for forecasting time series which combines moving average and autoregressive components for modelling time series.
- **Linear regression (LR)**: We compare our method with different versions of linear regression methods: ordinary least squares regression (OLSR), Ridge Regression (i.e., with 2-norm regularization), and Lasso (i.e., with 1- norm regularization).
- **XGBoost :** XGBoost is a powerful boosting tree based method and is widely used in data mining applications. XGBoost is an ensemble learning method. Sometimes, it may not be sufficient to rely upon the results of just one machine learning model. Ensemble learning offers a systematic solution to combine the predictive power of multiple learners. The result is a single model which gives the aggregated output from several models[13].The models that form the ensemble, also known as base learners, could be either from the same learning algorithm or different learning algorithms. Bagging and boosting are two widely used ensemble learners. Though these two techniques can be used with several statistical models, the most predominant usage has been with decision trees.Let's briefly discuss bagging before taking a more detailed look at the concept of boosting. These are some features of XGBoost that make it so interesting.
    - Regularization: XGBoost has an option to penalize complex models through both L1 and L2 regularization. Regularization helps in preventing overfitting
    - Handling sparse data: Missing values or data processing steps like one-hot encoding make data sparse. XGBoost incorporates a sparsity-aware split finding algorithm to handle different types of sparsity patterns in the data
    - Weighted quantile sketch: Most existing tree based algorithms can find the split points when the data points are of equal weights (using quantile sketch algorithm). However, they are not equipped to handle weighted data. XGBoost has a distributed weighted quantile sketch algorithm to effectively handle weighted data
    - Block structure for parallel learning: For faster computing, XGBoost can make use of multiple cores on the CPU. This is possible because of a block structure in its system design. Data is sorted and stored in in-memory units called blocks. Unlike other algorithms, this enables the data layout to be reused by subsequent iterations, instead of computing it again. This feature also serves useful for steps like split finding and column sub-sampling
    - Cache awareness: In XGBoost, non-contiguous memory access is required to get the gradient statistics by row index. Hence, XGBoost has been designed to make optimal use of hardware. This is done by allocating internal buffers in each thread, where the gradient statistics can be stored

Metrics:

1) Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

2) Rooted Mean Square Error (RMSE) : RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

III. PROPOSED SYSTEM

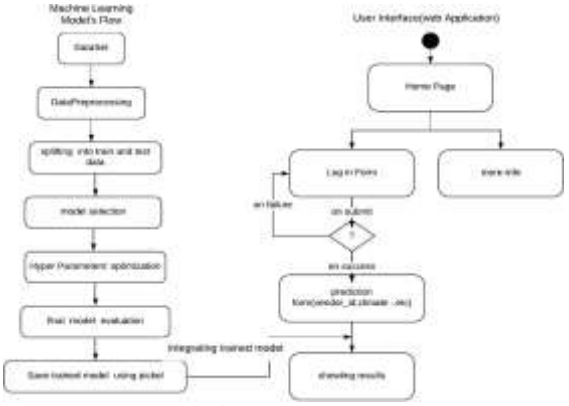

Figure 1 : Flowchart for Taxi Demand Prediction System Using Machine Learning

The above flowchart shows the working process of how we are going to implement the various modules.

There are two parts:

1) Generating machine learning models : To implement the ML model we are going to do Data Preprocessing then we split the data into a training set and testing set followed by selecting a prediction model in our case Linear Regression and Xgboost. We have also implemented Hyperparameters optimization to get suitable results.

2)Implementing the model in user interface using the django framework.: The resultant model is then taken and integrated in the web interface using django framework and the model is run online using pickle framework

IV. IMPLEMENTATION

Modules used in implementations are described below

NumPy: NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays While training machine learning models input data should be in N dimensional array i..e training and testing data should be pass to model in the form N dimensional array.Important Methods:

- numpy.mean() : to find mean of list of elements
- numpy.std(): to find standard deviation of list of elements.
- numpy.ndarray(list):to convert a normal python list into n dimensional numpy array.
- numpy.arange(start,end,step) :it will give a list of elements .

Pandas: Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns. pandas.DataFrame (): which converts a list of elements into 2 dimensional tabular data.Methods used on DataFrame:

- DataFrame.drop(column name) : by using this we can drop any column from dataframe
- DataFrame.join(DataFrame) : by using this we can join a dataframe to another dataframe
- DataFrame.iloc(a:b, c:d) by using this we selected particular cell from tabular data where a : b to select rows ' range and c:d to select columns range.

Sklearn :Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.Models used:

- Linear regression
- RandomForestRegression
- Support Vector Regression
- Xgboost

These regression models will be useful in cases where the end response of the system is in continuous values i.e output is in a certain range. ex: RankPrediction System : which is going to predict a student's rank in a certain range based on his/her marks after training using previous data.

Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.Hence we have used the following methods.

Scaling : to visualize given data we should bring out data into one common scale .To do this we have 2 power scalers in sklearn

StandardScaler: Standardize features by removing the mean and scaling to unit variance The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

Where x is actual value and u is the mean of the training samples or '0' if with_mean=False, and s is the standard deviation of the training samples .

Output range: [-1,1]

method:sklearn.preprocessing.StandardScaler(copy=True,with_mean=True,with_std=True)

MinMaxScaler: It Transforms features by scaling each feature to a given range.This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.The transformation is given by:

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0)) to scale data where X is a list of elements.

method: sklearn.preprocessing.MinMaxScaler(feature_range=(0, 1), copy=True)

Saving a machine learning Model: In machine learning, while working with a sklearn library, we need to save the trained models in a file and restore them in order to reuse it to compare the model with other models, to test the model on new data. The saving of data is called Serialization, while restoring the data is called Deserialization.Also, we deal with different types and sizes of data. Some datasets whose size is large (more than 1GB) can take very large time to train on a local machine even with GPU. Hence we store this trained data in serialised format so we can use it in future by deserializing it.For this purpose we are using a module called Pickel.

Pickel : module implements a fundamental but powerful algorithm for serializing and de-serializing a Python object structure. Pickle model provides the following functions

- **pickle.dump** to serialize an object hierarchy, you simply use dump().
- **pickle.load** to deserialize a data stream, you call the loads() function.

1) Django Framework:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. While working on the project we had to implement the following code and file structure after completing the implementation is explained below.

2) File Structure:

Project Folder

1. taxi_application
   - **admin.py** : this python file useful to store data admin and user details intodb.sqlite3 or any other database
   - **forms.py** : this file is useful to get data from user by providing some user form to user
   - **models.py** : this python file is useful to create database tables in the database
   - t**emplates** : these files contain html css javascript content to appear on browser as webpage
   - **Urls.py** : this python file to handle the user request from one page to another page through urls
   - **views.py** : this python file is important where our entire business logic implementation will be here
2. **manage.py** : this file useful to interact with in built django server through command line
3. **db.sqlite3** it is in built database providing by django only
4. **Settings.py** : these file will contains all our system configurations

3) Weather-API:

We used weather api to make this system automate i.e using api we are able to predict number of cabs to each location for every 15 minutes time interval with any human interaction (i..e with out this weather api user(admin) should enter inputs for every 25 minutes) So using this api we are fetching current time and weather conditions these are inputs in machine learning model.

4) Steps to use weather api :

1. **Create weather api account:** Firstly we have to create a account using mail which is open source(need not pay anything to use it)

2. **copy-link:** It will give a unique link for every individual which we can use in our code to fetch weather data.
3. **fetch required field**:This api will give the entire weather data in JSON object form ,so can get the required field using its keys.

V. RESULTS

index	vendor_id	time	weather_id		sunday	special	total_cabs
0	1	0.00	4	...	1	0	8
1	2	0.00	4	...	1	0	8
2	1	0.00	7	...	1	0	1
3	2	0.00	7	...	1	0	2
4	1	0.00	13	...	1	0	8
...	...	...	...	...	...	...	...
444427	1	23.75	264	...	1	0	21
444428	2	23.75	264	...	1	0	19
444429	1	23.75	265	...	1	0	2
444430	2	23.75	265	...	1	0	1
444431	2	0.00	79	...	1	0	1
[444432	rows x 15	columns]>					

Table 1:Data format during training

Observations :

- We have used 4 models to train the dataset and the results are as follows: Linear regression model is not performing well on this New York 2018 data where it is giving mean absolute error around 574.32 which means it will predict values with approximately 574 cabs variance from actual values.
- The Svr model somewhat performed well on this dataset but it is taking a lot of time to train the model and also it is giving 17.28 error which means it will predict values approximately with a variance of 17 cabs from actual values.
- The RandomForest Regression model is somewhat performing well on the previous 2 models where it is giving error around 6.2 which means it will predict values with approximately 6 cabs variance from actual values.

```
...  max_depth   error
     7   10.755175340569753
     max_depth   error
     8   9.424009045624889
     max_depth   error
     9   8.38079645837943
     max_depth   error
     10   7.69076088410607
     max_depth   error
     11   7.12491920270804
     max_depth   error
     12   6.632325476991428
     max_depth   error
     13   6.406911329036137
     max_depth   error
     14   6.273594208013961
     max_depth   error
     15   6.207109936222227
     max_depth   error
     16   6.194605903659446
     max_depth   error
     17   6.212692702931968
```

Figure 2: Random forest regression model error

- When compared with above 3 methods xgboost is predicting efficiently where it is giving an error of 3.4 which is a variation of 3.4 from the actual data.This xgboost model is taking a lot of time to train because we have implemented hyper parameters to get better accuracy.

```
start time is : 2020-01-21 14:12:18.073443
mean_absolute_error is : 3.4193293874512
how muc %this Xgboost model suitable for given dataset is : 86.189437539224317
end time is : 2020-01-21 14:39:18.343934
```

Figure 3:Xgboost model error

- The table below is showing the 4 different models performance after getting optimised hyper parameters using grid search. where among 4 models Xgboost gives less error i.e

3.419 MAE(mean absolute error) and this model is almost 86 percent suitable for available NYC2018 dataset, so based on this table we have selected Xgboost model for future predictions on unseen data**.**

model	error	score
LinearRegression	574.3293726345128	8.633922347007694
SVR	17.283461284627155	72.62321456274636
RandomForestRegressor	6.212669270293197	81.23729402842784
xgboost	3.4193293874512	86.31734927498248

Figure 4: Error comparison table

- After training the Xgboost model with optimised hyper parameters we have saved them in the form of pickle model which will be used to predict the future output.In this way we have done for all 12 months and saved them using python pickle module . These files contain a trained model which contains a prediction method to predict the result on given user input. So based on input date we will allocate respective month module.

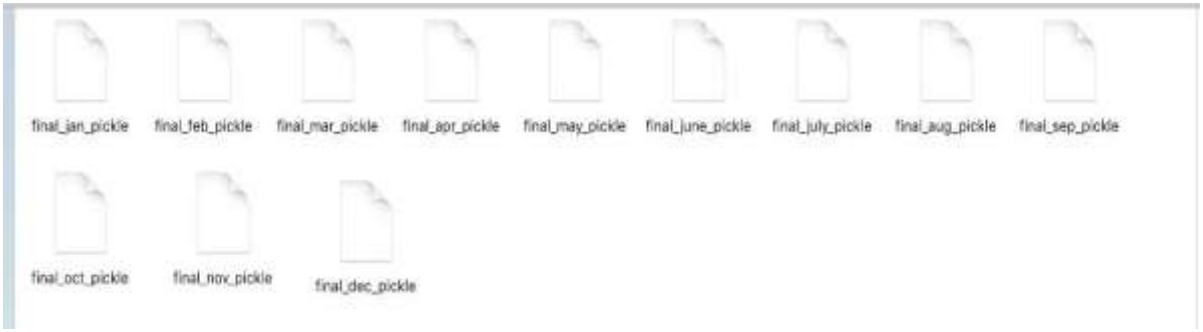

Figure 5: Saved Xgboost models

User Interface:

- This form would allow the user to enter vendor_id, date,time,sunday special day and climate ,on clicking the predict(submit) button it will show results in another web page. But it is involving lot of manual work (i.e for every 15 minutes user have to enter all required details in prediction form).So to overcome this we have added the AutoPrediction option by this one-click user will able to get results automatically for every 15 minutes this web page will refresh to show new results on itself.

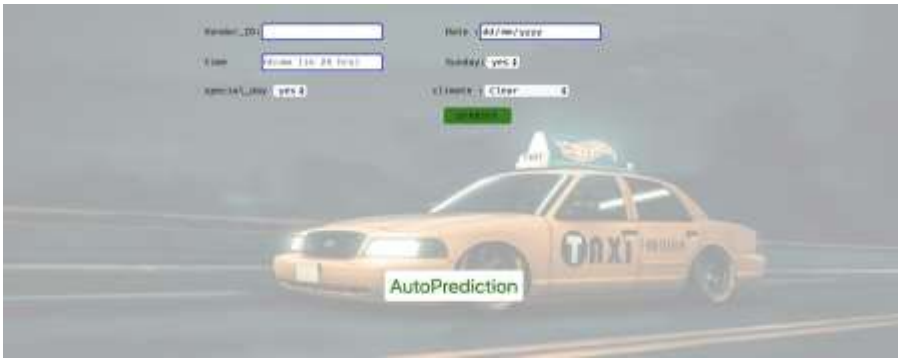

Figure 6: User Interface

- After the user clicks on the predict or AutoPrediction button it will show the result in the above form where for each location it will predict the number of cabs required in that particular time based on user inputs. NYC is divided into 265 locations and each of them have mainly two vendors (1. Yellow taxi cab vendor 2.Green Taxi cab Vendor) so if a user clicked on the predict button it will show the result based on respective user input along with vendor id if the user clicked on AutoPrediction it will go to predict for both cab vendors.

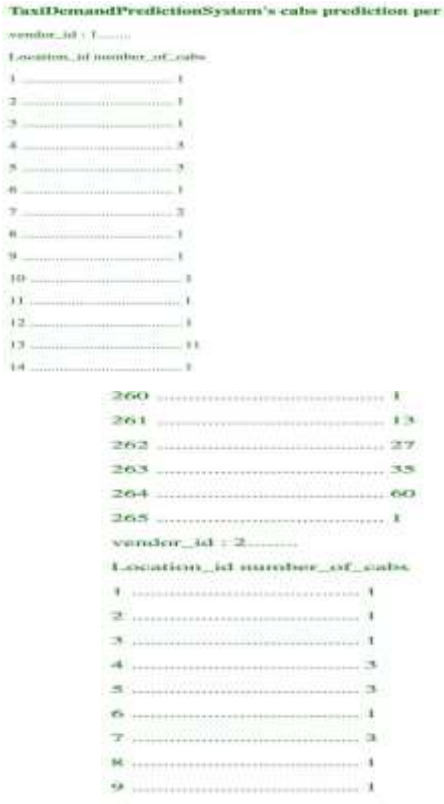

Figure 7:Final output of Taxi Demand Prediction

VI.CONCLUSION

Our actual data had inconsistent format for date and time so we have used a python builtin time module to get format for our convenience also the weather data had multi attributes so we used weather api to make a single column where the attributes are given an index value.In this way we had reduced the manual work.Then we had trained and tested the data using different models .Overall our models for predicting taxi pickups in New York City performed decently.The xgboost regression model using hyperparameters performed the best at a MAE value of 3.4 likely due to its unique ability to capture complex feature dependencies.Also we can say from our observation that hyperparameters played a crucial role in getting better accuracy. The next closest is RandomForest regression model achieving a MAE value of 6.19. So finally we are considering XGboost.We also observed that when the model is complex it takes more time to train. Also the models accuracy and time to train has improved for scaled data rather than unscaled data. Then we have implemented the model into a web application successfully with the help of Django framework and we also added an additional Auto predict feature to automatically update the result for every 15 minutes time interval.

REFERENCES

[1] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, 2018 “Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction”AAAI Publications.

[2] Yellow Taxi Trip Records https ://s3.amazonaws.com/nyc-tlc/trip+data/yellow_tripdata_2019-01.csv

[3] G. Niemeyer. (2008) Tips & tricks about geohash. [Online]. Available: http://geohash.org/site/tips.html

[4] NYC Taxi Limousine Commission. Taxi and limousine commission (tlc) trip record data. [Online]. Available: http://www.nyc.gov/html/tlc/html/about/trip record data.shtml

[5] Chen, T., and Guestrin, C. 2016. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. ACM.

[6] . Jun Xu , Rouhollah Rahmatizadeh, Ladislau Bölöni, and Damla Turgut, “Real-Time Prediction of Taxi Demand Using Recurrent Neural Networks”, IEEE Transaction on Intelligent transport system, vol. 19, no. 8, pp. 2572-2581,Aug.2018.

[7] B. Gupta S. Awasthi R. Gupta 2018“Taxi Travel Time Prediction Using Ensemble-Based Random Forest and Gradient Boosting Model” Springer Nature Singapore Pte Ltd. 978-981-10-7200

[8] ZHANG, D., LI, N., ZHOU, Z.-H., CHEN, C., SUN, L., AND LI, S., "IBAT: detecting anomalous taxi trajectories from GPS traces.," In the 13th conference on ubiquitous computing, UbiComp, 2011.

[9] Y. Lv, Y. Duan, "Traffic flow prediction with big data: A deep learning approach," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 865-873, 2015.

[10] Moreira-Matias, L., et al., "Predicting Taxi–Passenger Demand Using Streaming Data," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 3, pp. 1393-1402,

[11]Metrics and Python https://towardsdatascience.com/metrics-and-python-850b60710e0c [12]Regression XGboost Classifier https://www.learnbay.co/data-science-course/tag/regression/

[12]Vanichrujee, Ukrish, Horanont, Teerayut, Pattara-atikom, Wasan,Theeramunkong, Thanaruk,Shinozaki, Takahiro, 2018 “ Taxi Demand Prediction using Ensemble Model Based on RNNs and XGBOOST” 10.1109/ICESIT-ICICTES.2018.8442063