New York city taxi fare prediction

Project progress report

1. Introduction

1.1. Motivation

In the New York city, people use taxi in a frequency much higher than any other cities of US, there are roughly 200 million taxi rides in New York City each year. The ride-hailing companies like Uber, Lyft and others are giving ability for users to plan their trips ahead which is not possible for city taxi riders. It would be very helpful for city taxi riders if they could plan their taxi rides ahead just like as in any other online taxi hailing apps.

1.2. Problem Statement

The objective of the problem is to predict the taxi fare by analyzing the previous taxi rides behavior in New York City, given the pickup and drop-off locations using efficient machine learning models.

1.3. Assumptions

We are making the following assumptions in our project:

- Considering Haversine distance as the distance between pickup and drop-off coordinates.
- Not considering additional charges like delay or cancellation charges of trips.
- Assuming that the passenger count is between 1 to 6.

1.4. Related work:

- Fare and Duration Prediction: A Study of New York City Taxi Rides, by Christophoros Antoniades, Delara Fadavi, Antoine Foba Amon Jr., December 16, 2016. http://cs229.stanford.edu/proj2016/report/AntoniadesFadaviFobaAmonJuniorNewYorkCityCabPricing-report.pdf

2. Exploring Data Set and Visualization

The dataset in Kaggle is having 55M taxi rides from which we randomly selected 2M trips as training dataset and around 10k trips as test dataset. Further we have split the training data set into 80% training and 20% as validation set.

2.1. Features

- pickup_datetime - timestamp value indicating when the taxi ride started.
- pickup_longitude - longitude coordinate of where the taxi ride started.
- pickup_latitude - latitude coordinate of where the taxi ride started.
- dropoff_longitude - longitude coordinate of where the taxi ride ended.
- dropoff_latitude - latitude coordinate of where the taxi ride ended.
- passenger_count - number of passengers in the taxi ride.
- fare_amount - dollar amount of the cost of the taxi ride.

2.2. Initial observations and Data cleaning

- The minimal \`fare_amount\` is negative. As this does not seem to be realistic, we will drop them from the dataset.
- Is there some missing data? if so than those doesn't make sense. Removing them impacts very little affect to our model.
- The minimum passenger count for some of the taxi trips are having 0 value. Zero passengers don't make any sense, so we are dropping those data.
- We have noticed maximum passenger count in training set more than 200, which is unreasonable, so we are dropping all the taxi trips with more than 6 passenger count.
- There are some pick and drop-off locations which falls outside New York and some pick and drop-off locations are not valid coordinates, so we are dropping those trips.
- The average \`fare_amount\` is about $11.4 USD with a standard deviation of $9.9 USD. When building a predictive model, we want to be better than $9.9 USD.

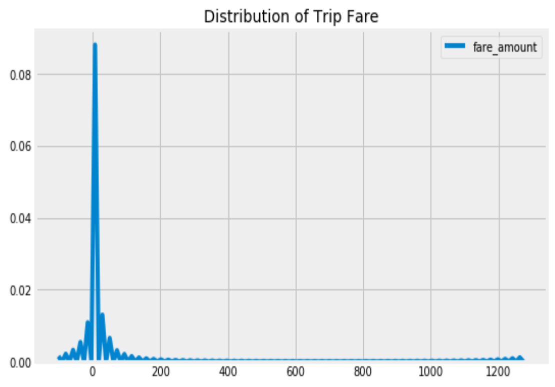
3. Exploratory Data Analysis (EDA)

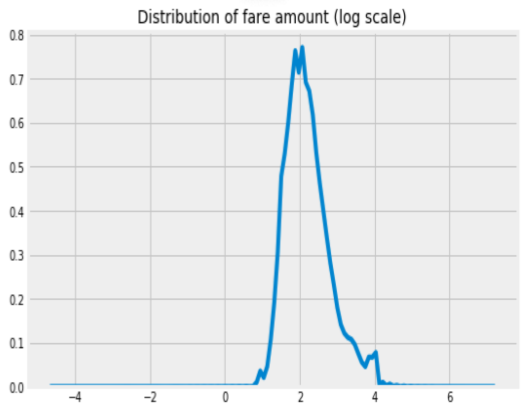
We performed feature engineering on the following hypothesis which could affect the prediction of taxi fare:

- Time of Travel: The taxi fare might be higher during peak traffic hours.
- Day of Travel: Weekday, weekend and specific hour of the day might impact fare amount.
- Trip distance**:** distance and the fare amount are directly proportional.
- Pickup or Drop-off Neighborhood: Kind of neighborhood impacts fare amount.
- is it a trip to/from airport: Trips to/from airport generally have a fixed fare.

3.1. Fare amount distribution

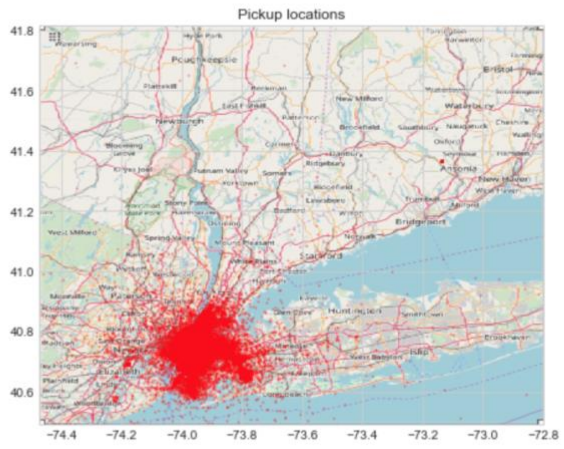
When we observed fare amount distribution as per following graph, we found that there are around 300 records having negative fares. Since, cost of a trip cannot be negative we removed such data points from the data. Also, fare amount follows long tail distribution. We understood the distribution of fare amount better by taking a log transformation after removing the negative fares- this makes the distribution close to normal.

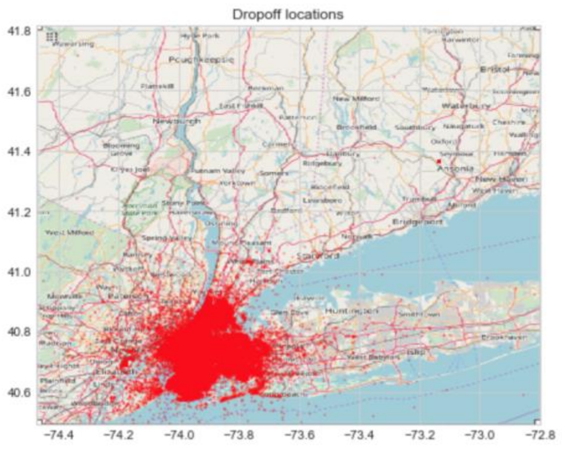




3.2. Distribution of Geographical Features

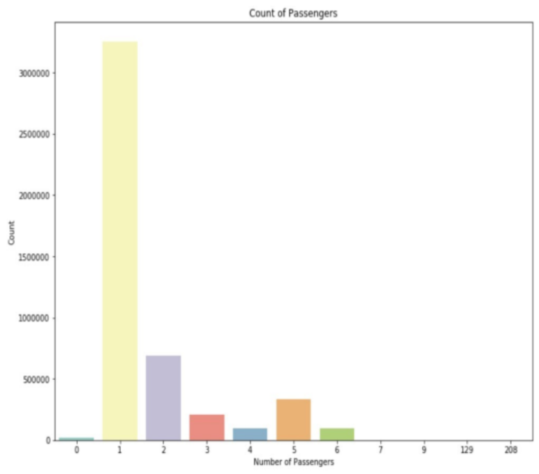
The valid range of latitudes and longitudes of earth are between -90 to 90 and -180 to 180 respectively. But in the training data set we observed latitudes and longitudes in range of (-3488.079513, 3344.459268) which are invalid. On further analysis, we also identified a set of 114K records which had both pickup and drop-off coordinates at the Equator. Since, this data is for taxi rides in New York, we remove these rows from our analysis. Such errors where not found in the test data. The following graph describes pickup and drop-off coordinates with respect to locations on map.

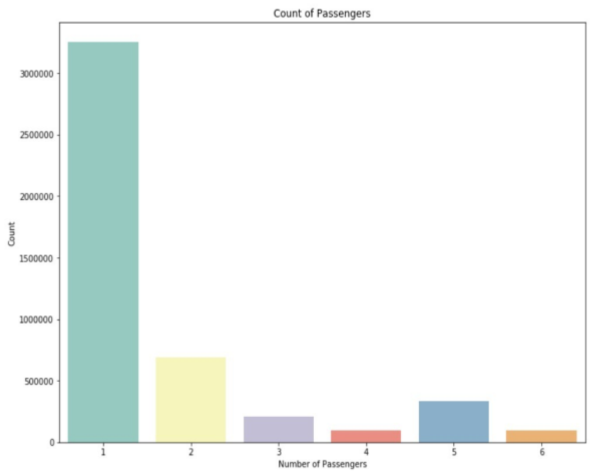




3.3. Passenger Count distribution

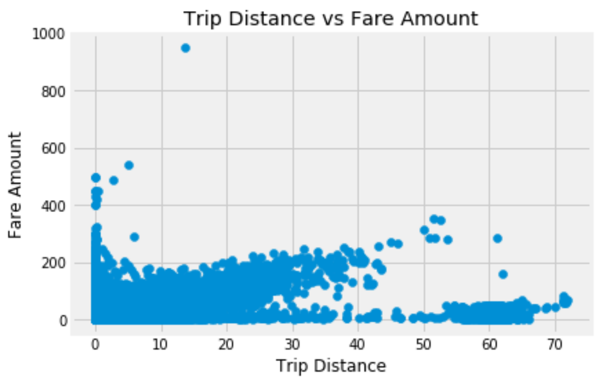
When we plotted graph of distribution of passenger count, we observed that there are data points with range of 7 to 208 passengers. These data points do not make any sense; hence we considered the data points with passenger count range from 1 to 6 and removed all other data.

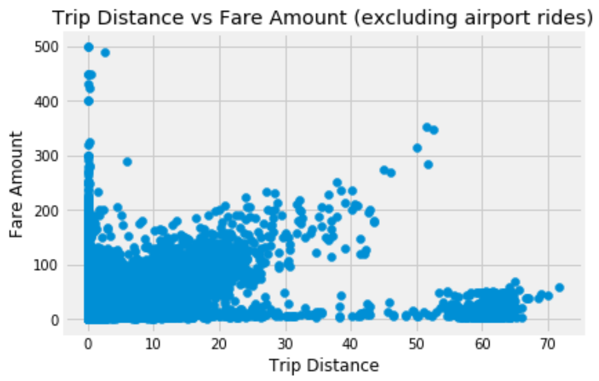




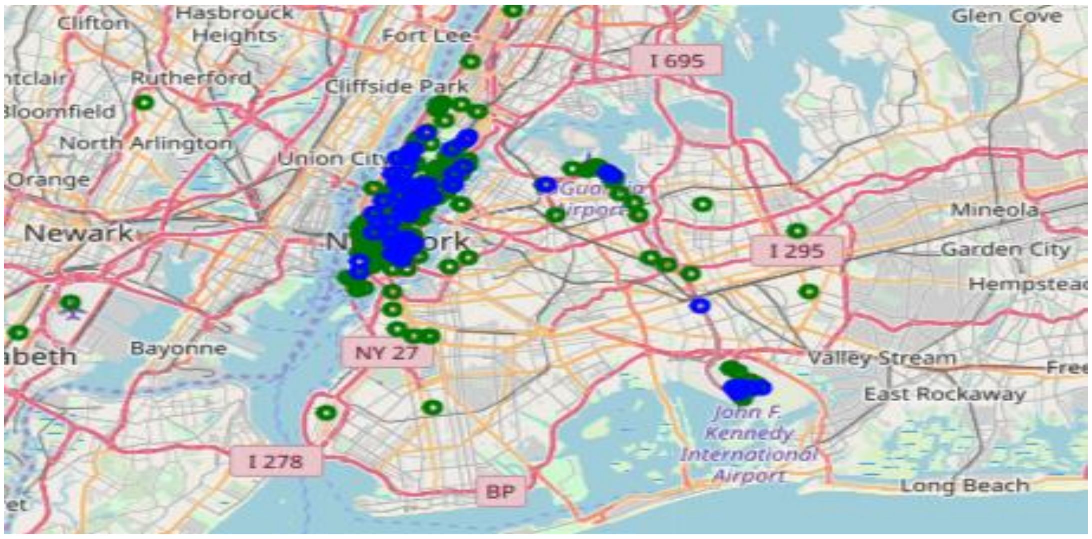
4. Feature Engineering:

One of our hypotheses was just the fare amount should ideally increase with trip distance. A scatter plot between trip distance and fare amount showed that though there is a linear relationship, the fare per mile (slope) was lower, and there were a lot of trips whose distance was greater than 50 miles, but fare was very low. To check if this was the case because of airport trips, we removed the airport trips and plotted the distribution. We then observed that fare per mile was higher and another small cluster with trip distance >50 miles was observed.

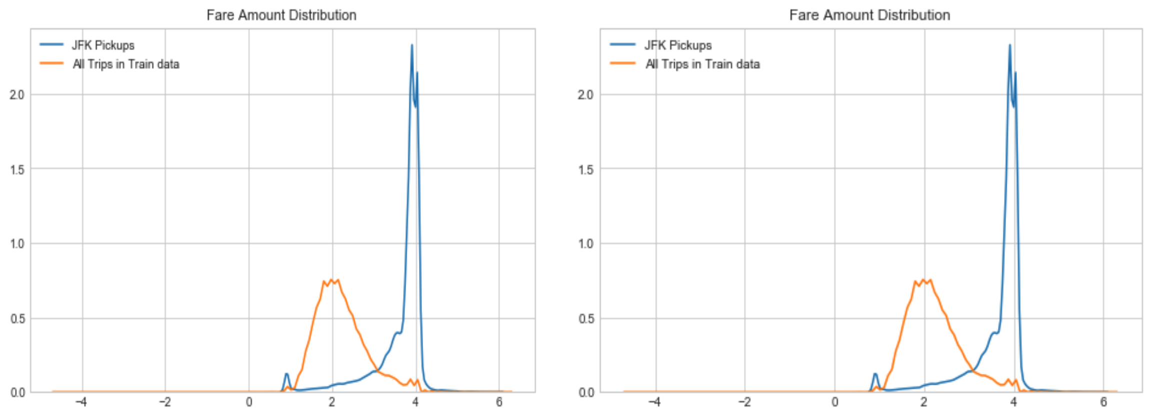




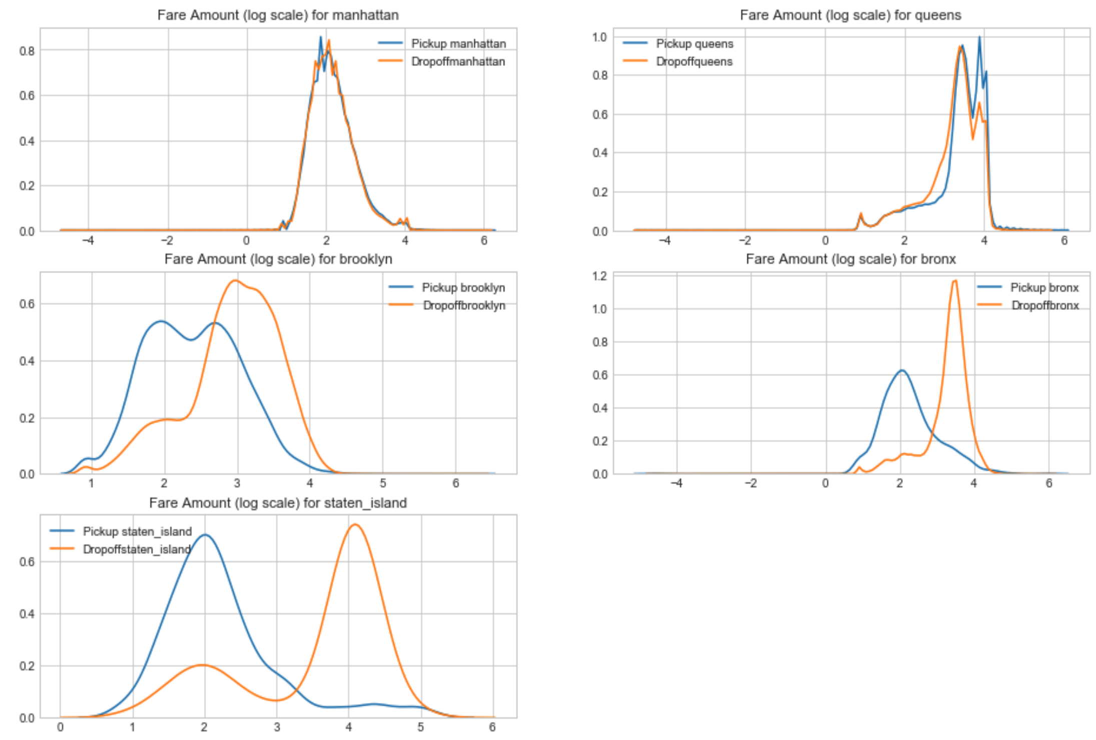
The next step was to see if there was a particular region where the trip distance>50 miles was observed. This showed that, there were a lot of pickups and drop-offs from lower Manhattan. This led to a new feature — pickup_is_lower_manhattan and dropoff_is_low_manhattan.



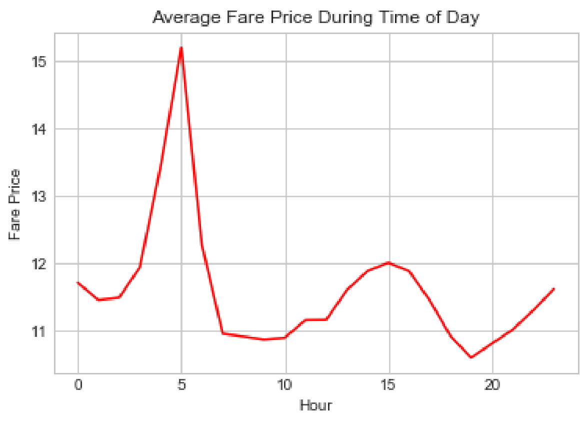
We can see that there is a high density of pickups near JFK and La Guardia Airport. We then looked at what is the average fare amount for pickups and drop offs to JFK, compared to all trips in the train data and observed that fare was higher for airport trips. Based on this observation, we created features to check whether a pickup or a drop-off was to any one of the three airports in New York — JFK, EWR or LaGuardia.



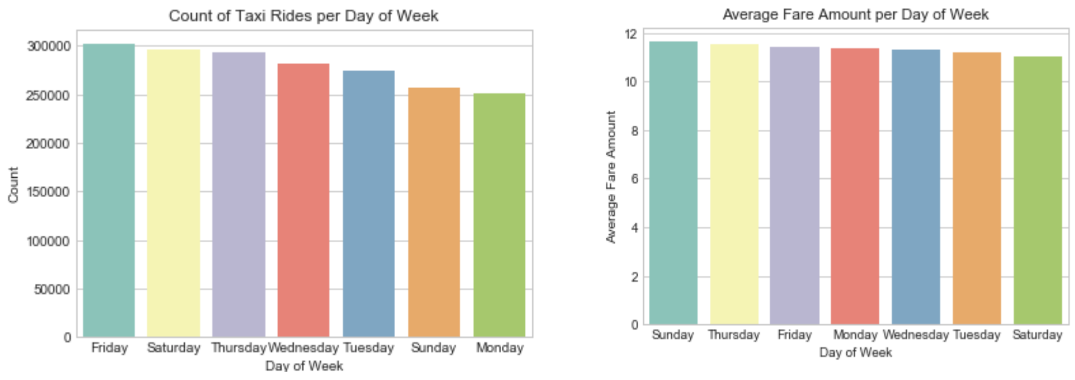
The next step was to check whether our hypothesis of fare from certain neighborhoods are higher than the rest, based on the 5 Boroughs New York city is divided — Manhattan, Queens, Brooklyn, Staten Island and Bronx, each pickup and drop off location was grouped into these 5 neighborhoods. And yes, our hypothesis was right- except for Manhattan which had most of the pickups and drop offs, for every other neighborhood, there was a difference in the pickup and drop off fare distribution. Also, Queens had a higher mean pickup fare compared to other neighborhoods.



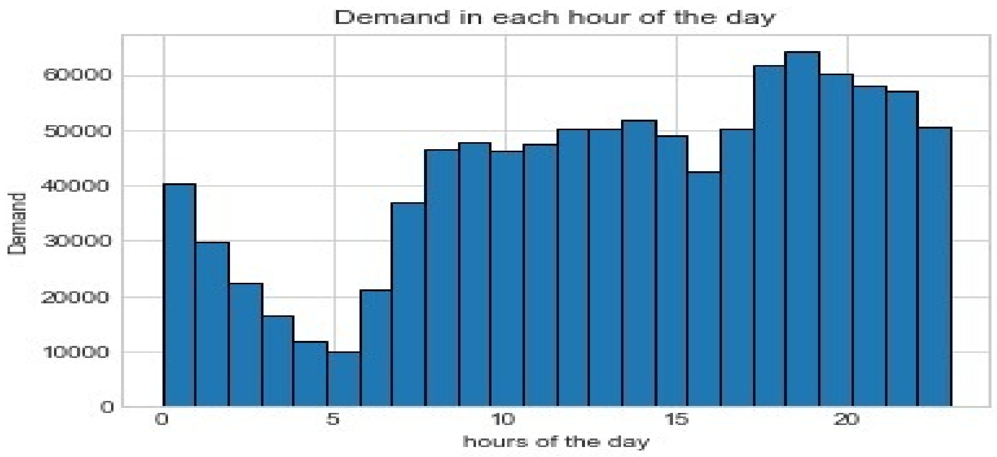
The next step was to analyze how the fares have changed over time, is to create features like hour, day of the week, day, month, year from pickup datetime. So, we plotted graphs for each of them with respect to the fare amount.



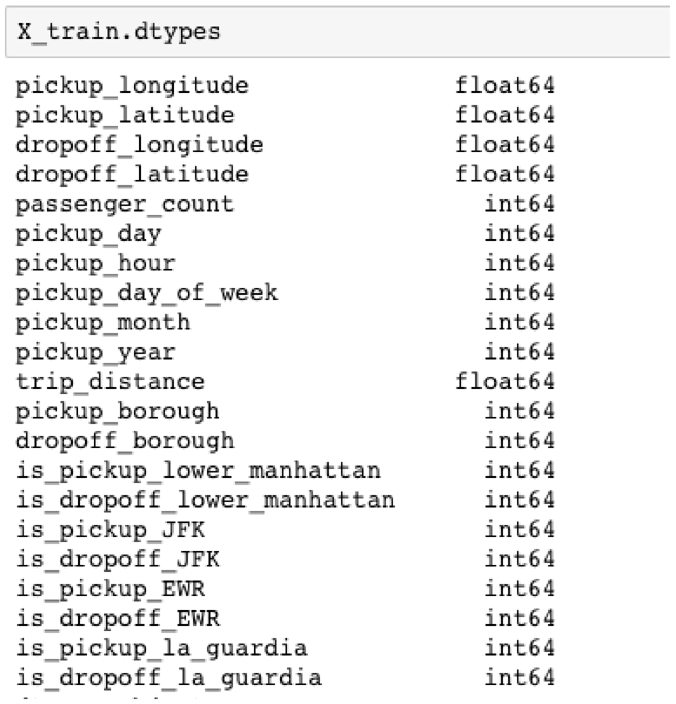
From the above graph, we can see that the fare amount is at its peak at 5 in the morning. And during the analysis, we could find that at 5 AM the greatest number of trips are to/from the airport.



From the above graph, we can see that number of pick ups are higher on Saturday but still fare amount is relatively low on Saturday. While on Sunday and Monday, number of trips are lower while fare amount is relatively high. Also, From the below graph we can infer that taxi demand is the highest at 7 PM and the lowest at 5 AM given that there are constant number of taxis available any time for 24 hours.



Finally, we considered the following 21 features for our Machine Learning model implementation:



```
X_train.dtypes
```

```
pickup_longitude              float64
pickup_latitude               float64
dropoff_longitude             float64
dropoff_latitude              float64
passenger_count                 int64
pickup_day                      int64
pickup_hour                     int64
pickup_day_of_week              int64
pickup_month                    int64
pickup_year                     int64
trip_distance                 float64
pickup_borough                  int64
dropoff_borough                 int64
is_pickup_lower_manhattan       int64
is_dropoff_lower_manhattan      int64
is_pickup_JFK                   int64
is_dropoff_JFK                  int64
is_pickup_EWR                   int64
is_dropoff_EWR                  int64
is_pickup_la_guardia            int64
is_dropoff_la_guardia           int64
```

5. Models and Prediction

We wanted to have one solution without applying any machine learning solution so that we can test and compare that baseline model in order to improve our subsequent solution (A baseline model is a solution to a problem without applying any machine learning techniques). The most common way of creating a baseline model is taking the most common value in case of classification and calculating the average in a regression problem. In this analysis, since we are predicting fare amount (which is a quantitative variable)— but we will predict the fare amount using linear regression machine learning algorithm. This resulted in an RMSE of 8.35. So, any machine learning algorithm we implement in this problem, should have an RMSE less than 8.35.

Further, for our regression problem, we considered the following machine learning models to better predict the fare amount and ultimately decrease the error rate:

- Random Forest
- Light GBM
- Neural Network

Random Forest and light GBM are ensembles of the decision trees. The only difference is that they both use different techniques to build decision tree. Random Forest uses bagging while Light GBM uses boosting for building decision trees. Boosting and Bagging are described briefly as below:

- **Bagging:**

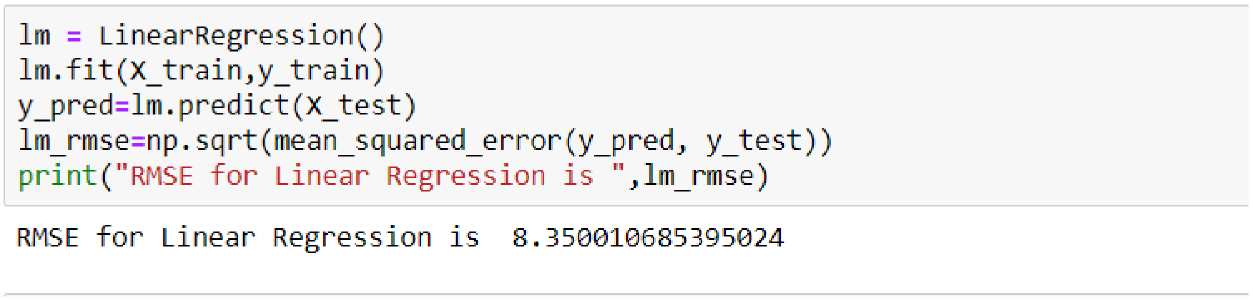
  In bagging (also called bootstrap aggregating), multiple models are created, and the final output is an average of all the different outputs predicted by multiple decision models. Here, bootstrap samples are taken and each sample trains a weak learner. This technique is mainly used to reduce the high variance which is usually seen in decision trees.

- **Boosting:**

  In boosting, multiple weak learners are ensembled to create a strong learner. Unlike bagging, Boosting uses all the samples to train every learner. Light GBM is one which uses boosting for creating decision trees. Boosting is mainly used to reduce the high bias. There are some variants of gradient boosting decision trees e.g. Xgboost and light GBM which are based on this method. There are two strategies using which the decision trees grow: leaf wise growth and level wise growth. The algorithm which we are taking into consideration i.e. Light GBM follows the leaf wise growth while training the decision tree.

5.1. Linear Regression:

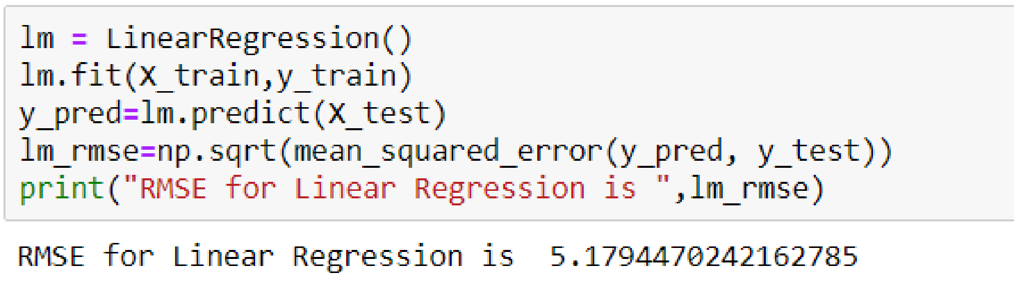
Linear regression is the most basic regression algorithm in machine learning. It has a high bias and not very flexible algorithm. It simply predicts the line where the prediction error is the least. Also, as it tries to minimize the squared error, Linear regression is highly susceptible to outliers.



```
lm = LinearRegression()
lm.fit(X_train,y_train)
y_pred=lm.predict(X_test)
lm_rmse=np.sqrt(mean_squared_error(y_pred, y_test))
print("RMSE for Linear Regression is ",lm_rmse)
```

```
RMSE for Linear Regression is  8.350010685395024
```

As you can see, RMSE for Linear Regression without feature engineering is 8.35 while with feature engineering, RMSE is 5.18 which is a significant improvement from the baseline model.

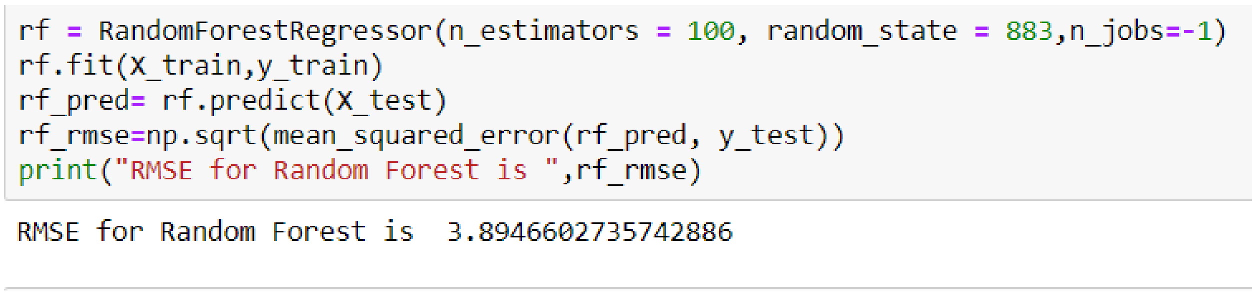


```
lm = LinearRegression()
lm.fit(X_train,y_train)
y_pred=lm.predict(X_test)
lm_rmse=np.sqrt(mean_squared_error(y_pred, y_test))
print("RMSE for Linear Regression is ",lm_rmse)
```

```
RMSE for Linear Regression is  5.1794470242162785
```

5.2. Random Forest:

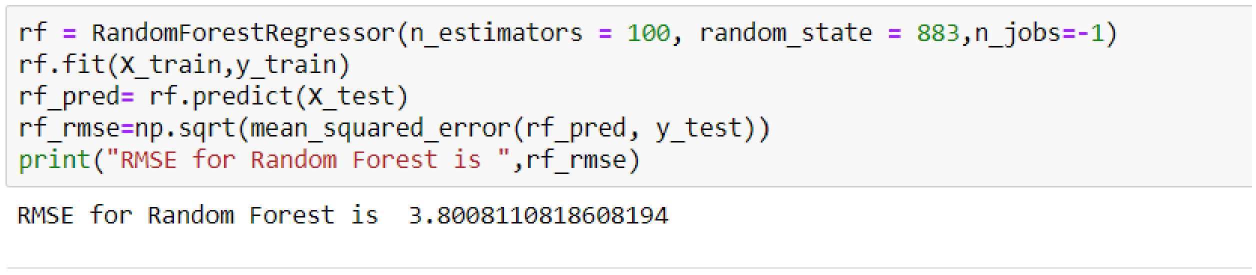
Random Forest is far more flexible than a Linear Regression. So, it means that it has lower bias, and it can fit the training data more efficiently. Complex models can often memorize the underlying data and so there's a high chance of overfitting. Parameter tuning is used to avoid the problem of overfitting.



```
rf = RandomForestRegressor(n_estimators = 100, random_state = 883,n_jobs=-1)
rf.fit(X_train,y_train)
rf_pred= rf.predict(X_test)
rf_rmse=np.sqrt(mean_squared_error(rf_pred, y_test))
print("RMSE for Random Forest is ",rf_rmse)
```

```
RMSE for Random Forest is  3.8946602735742886
```

As you can see, RMSE for Random forest without feature engineering is 3.89 while with feature engineering, RMSE is 3.80 which is not a significant improvement but there's still some.

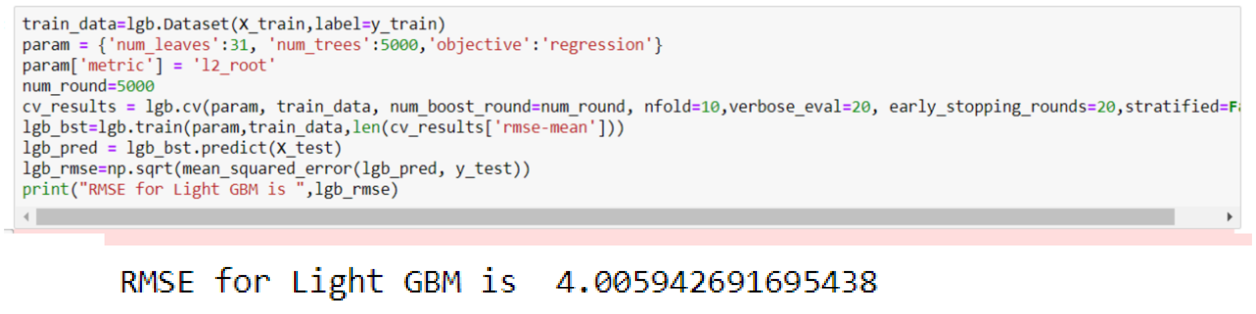


```
rf = RandomForestRegressor(n_estimators = 100, random_state = 883,n_jobs=-1)
rf.fit(X_train,y_train)
rf_pred= rf.predict(X_test)
rf_rmse=np.sqrt(mean_squared_error(rf_pred, y_test))
print("RMSE for Random Forest is ",rf_rmse)
```

```
RMSE for Random Forest is  3.8008110818608194
```

5.3. Light GBM:

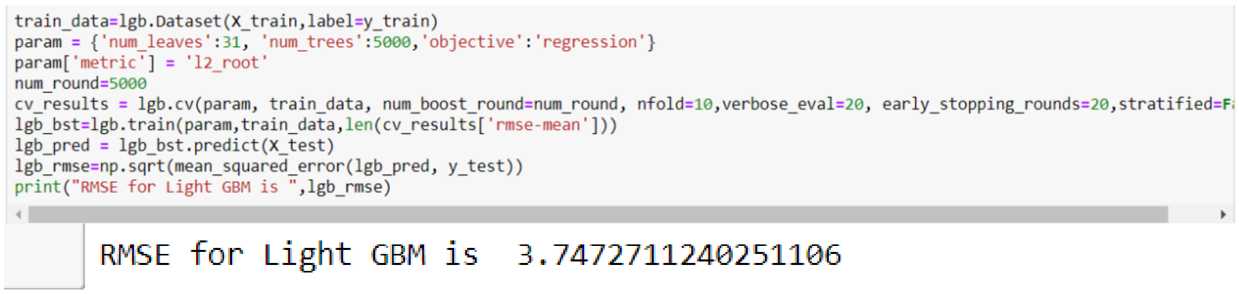
LightGBM is decision tree-based machine learning algorithm which uses boosting method for building the decision tree. The Light GBM grows leaf-wise instead of level-wise which happens to be the case for other decision tree-based algorithms. This algorithm chooses the node which will result in maximum delta loss to split. Light GBM is very fast, takes quite less RAM to run, and focuses on the accuracy of the result.



```
train_data=lgb.Dataset(X_train,label=y_train)
param = {'num_leaves':31, 'num_trees':5000,'objective':'regression'}
param['metric'] = 'l2_root'
num_round=5000
cv_results = lgb.cv(param, train_data, num_boost_round=num_round, nfold=10,verbose_eval=20, early_stopping_rounds=20,stratified=F
lgb_bst=lgb.train(param,train_data,len(cv_results['rmse-mean']))
lgb_pred = lgb_bst.predict(X_test)
lgb_rmse=np.sqrt(mean_squared_error(lgb_pred, y_test))
print("RMSE for Light GBM is ",lgb_rmse)
```

```
RMSE for Light GBM is  4.005942691695438
```

As you can see, RMSE for Light GBM without feature engineering is 4.005 while with feature engineering, RMSE is 3.74 which shows the improvement.



```
train_data=lgb.Dataset(X_train,label=y_train)
param = {'num_leaves':31, 'num_trees':5000,'objective':'regression'}
param['metric'] = 'l2_root'
num_round=5000
cv_results = lgb.cv(param, train_data, num_boost_round=num_round, nfold=10,verbose_eval=20, early_stopping_rounds=20,stratified=F
lgb_bst=lgb.train(param,train_data,len(cv_results['rmse-mean']))
lgb_pred = lgb_bst.predict(X_test)
lgb_rmse=np.sqrt(mean_squared_error(lgb_pred, y_test))
print("RMSE for Light GBM is ",lgb_rmse)
```

```
RMSE for Light GBM is  3.7472711240251106
```

Model Name	RMSE without feature Engineering	RMSE with normalizing fare
Linear regression (baseline model)	8.35	5.18
Random forest	3.89	3.80
Light GBM	4.00	3.74

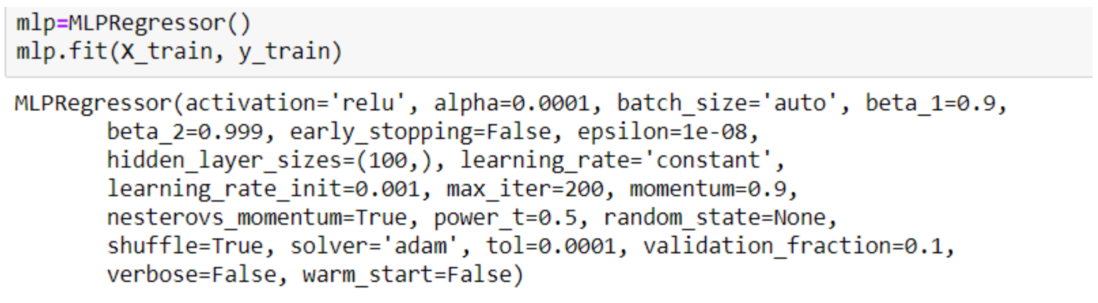
5.4. Neural Network:

Artificial Neural network is a collection of nodes which are used to predict the output for classification problems as well as regression problems. Here, neural network consists of one or more input layer and one output layer. Each layer consists of nodes and each node is called the activation unit. Here, we have used two variants of NNs:

a. Multilayer Perceptrons
b. Sequential Model

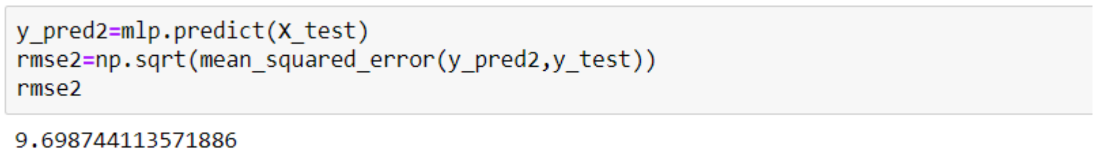
a. Multilayer Perceptrons:

Multilayer Perceptrons are the classical type of neural network. It consists of at least three or more layers of activation units (We used RELU).



```
mlp=MLPRegressor()
mlp.fit(X_train, y_train)
```

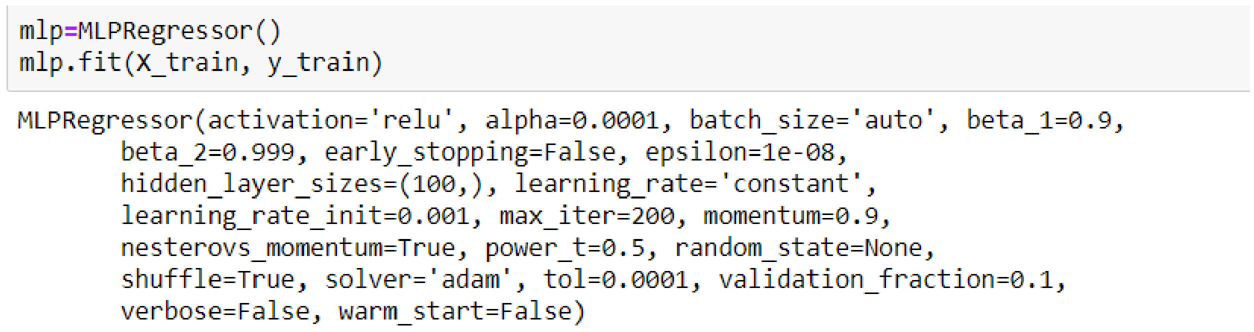
```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```



```
y_pred2=mlp.predict(X_test)
rmse2=np.sqrt(mean_squared_error(y_pred2,y_test))
rmse2
```

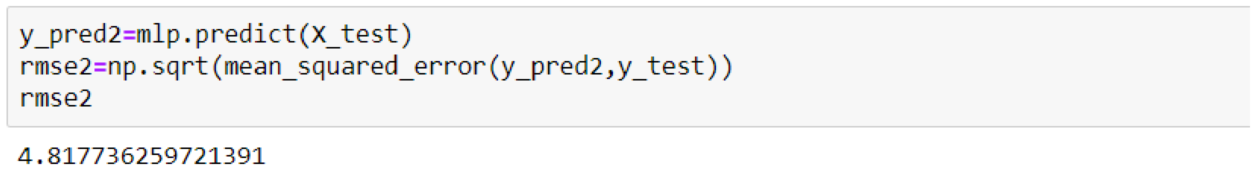
```
9.698744113571886
```

As you can see, RMSE for MLP without feature engineering is 9.69 while with feature engineering, RMSE is 4.81 which shows significant improvement. However, it still lags behind Light GBM.



```
mlp=MLPRegressor()
mlp.fit(X_train, y_train)
```

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
       beta_2=0.999, early_stopping=False, epsilon=1e-08,
       hidden_layer_sizes=(100,), learning_rate='constant',
       learning_rate_init=0.001, max_iter=200, momentum=0.9,
       nesterovs_momentum=True, power_t=0.5, random_state=None,
       shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
       verbose=False, warm_start=False)
```

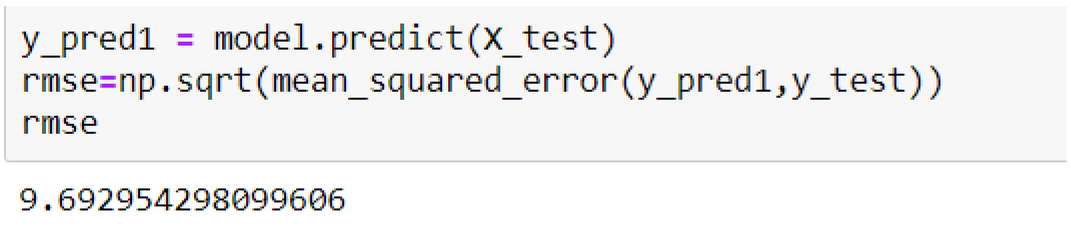


```
y_pred2=mlp.predict(X_test)
rmse2=np.sqrt(mean_squared_error(y_pred2,y_test))
rmse2
```

```
4.817736259721391
```

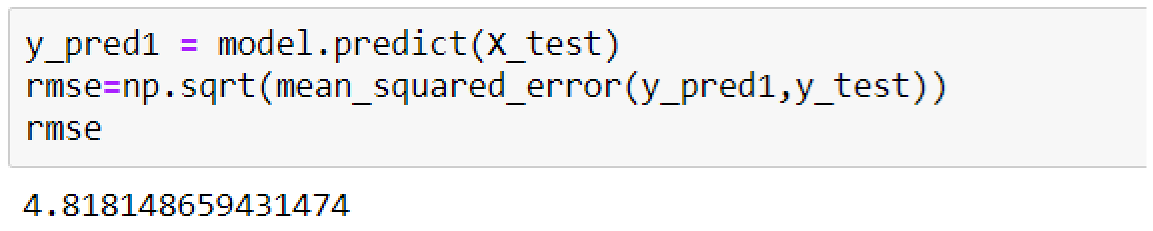
From the above observations, one could use random forest model to make the prediction but as the light GBM is has

b. Sequential Model:
This type of neural network is similar to traditional neural network. There are few exceptions: In sequential Model, all the layers are stacked sequentially. In keras, we can implement this with defining the parameters such as input, output, input shape and output shape.



```
y_pred1 = model.predict(X_test)
rmse=np.sqrt(mean_squared_error(y_pred1,y_test))
rmse
```

```
9.692954298099606
```



```
y_pred1 = model.predict(X_test)
rmse=np.sqrt(mean_squared_error(y_pred1,y_test))
rmse
```

```
4.818148659431474
```

As you can see, RMSE for Sequential Model without feature engineering is 9.69 while with feature engineering, RMSE is 4.82 which shows significant improvement. However, it still lags behind

Light GBM but similar to MLP. And we can see further improvements in the RMSE upon tuning the neural networks.
In our results, Random Forest and Light GBM do significantly better than other regression models. But as Light GBM has less variance, it is faster and more focused on accuracy compared to Random Forest, it is wise to choose Light GBM over Random Forest.

6. Lessons learned

We learned the importance of EDA and feature engineering as it helped us in breaking down the project into small chunks and perform microanalysis. This resulted in reduction of error by ~8%. We also explored unknown ML models Light GBM and neural networks. We further plan to improve the performance of neural networks by tuning the hyper-parameters and building accurate models. We predict a further reduction using neural networks because the jump in error was the highest using feature engineering and adding further features to these networks might only improve the performance.

7. Conclusion

We came to the conclusion that Trip distance was found to be the most important and Passenger count to be least important feature in determining the fare amount. There was a significant drop in RMSE after feature engineering. Also, we found that Fare amount and demand were inversely proportional in relation. We could also state that final model was a tradeoff between loss, time taken and variance while taking into account the model complexity.

8. Division of labor

Akash Tonne	EDA and Data Preprocessing prior to cleaning the data using bar plots. Normalized the data as a part of Data cleaning. Removed all the taxi trips having the negative fare amount as a part of data cleaning. Plot the location data (both training and test data) i.e. coordinates on a map. Implemented Linear Regression as a baseline model and calculated RMSE for it. Did the feature engineering on Trip distance and pick up/drop off co-ordinates. Ran the Light GBM, Linear Regression and Neural networks sequential models after doing the feature engineering. Prepared the Final report (Models and Prediction)
Amogh Raj	EDA and Data Preprocessing prior to cleaning the data using bar plots. Removed taxi trips having the missing data as a part of missing data as a part of data cleaning. Plot the location data (both training and test data) i.e. coordinates on a map. Wrote a function to encode a particular day to its numeric value. Implemented Random Forest and Linear Regression models and calculated RMSE for those models. Did the feature engineering in Trip distance and dividing the New York City into five boroughs. Ran the Light GBM and MLP models after doing the feature engineering. Prepared the Final report (Models and Prediction as well as conclusion, EDA)

Gautham J	EDA and Data Preprocessing prior to cleaning the data using bar plots. Normalized the data as a part of Data cleaning. Removed taxi trips having passenger count less than zero and greater than six as a part of data cleaning. Implemented Light GBM and Random Forest models and calculated RMSE for those models. Did the feature engineering on pick up/ Drop off co-ordinates and fare amount distribution. Ran the Sequential Model and Linear Regression models after doing the feature engineering. Prepared the Final report (Models and Prediction, Data Exploration)
Parth Shah	EDA and Data Preprocessing prior to cleaning the data using bar plots. Removed the taxi trips having zero feature value which does not make sense in this dataset. Wrote a function to select the location data only which is within the boundary of the city. Implemented Light GBM and Random Forest models and calculated RMSE for those models. Did the feature engineering on fare amount distribution and trip distance. Ran the Random Forest and MLP models after doing the feature engineering. Prepared the Final report (Models and Prediction, Data Visualization)

9. References

Haversine formula:
https://stackoverflow.com/questions/1502590/calculate-distance-between-two-points-in-google-maps-v3

Data Set:
https://www.kaggle.com/c/new-york-city-taxi-fare-prediction

Calculate distance between locations:
https://www.travelmath.com/flying-distance/

Open street map to grab using bounding box a map:
https://www.openstreetmap.org/export#map=8/52.154/5.295

we have taken the boundary box [long_min, long_max, latt_min, latt_max] of New York city using website
https://www.mapdevelopers.com/geocode_bounding_box.php

GitHub Links:
https://github.com/parthberk/Projects/tree/master/4th%20Quarter/PRDM

https://github.com/akashct/New-York-city-taxi-fare-prediction