Cab Fare Prediction Project

25.05.2019

Kushagra Raj Tiwari

Third Year, B.Tech (CSE)

Maulana Azad National Institute of Technology, Bhopal

Table of Contents

Problem Statement

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

Data Details

- pickup_datetime - timestamp value indicating when the cab ride started.
- pickup_longitude - float for longitude coordinate of where the cab ride started.
- pickup_latitude - float for latitude coordinate of where the cab ride started.
- dropoff_longitude - float for longitude coordinate of where the cab ride ended.
- dropoff_latitude - float for latitude coordinate of where the cab ride ended.
- passenger_count - an integer indicating the number of passengers in the cab ride.

Problem Analysis

Above variables are given with data and with these variables along with other features which will be introduced in feature engineering I will predict fare of a cab ride. This is a regression problem under supervised machine learning. After data cleaning, exploration and feature engineering, a machine learning model will be build and further optimized.

Metrics

I will use Root Mean Square Error (RMSE) error metrics for this regression problem. The reason for using this error metrics is that RMSE gives more weight to big errors and also relates with frequency distribution of errors. This metric assumes that the error is unbiased and follows a normal distribution. The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values.As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how

accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Business Hypothesis for further Analysis

Let us built some general business hypothesis for this project. These are some factors which decide fare amount for cab rides.

Trip distance

Generally for taxi rides the cab fare is directly proportional to trip distance i.e. if the distance to be traveled is more, then fare should be higher

Time of Travel

Due to more demands of taxi rides on particular hours of day i.e. during peak traffic hours, the taxi fare may be higher.

Day of Travel

Due to weekend plans of public, traffic may increase on those days thus fare amount may differ on weekday and weekends.

Weather Conditions

Due to extreme weather conditions there may be lower availability of cabs and hence higher fares.

Pickup or drop-off near airport

Trips to/from airport generally have a fixed and higher fare.

Pickup or Drop-off Neighborhood

Fare may be different based on the kind of neighborhood.

Availability of taxi

If a particular location has a lot of cabs available, the fares may be lower.

Based on these hypothesis we will visualise data and extract important insights from it which is important before modelling. Above hypothesis are based on general observations, these is not based on data. For example, hypothesis regarding weather conditions can only be used when we are having appropriate and correct weather data with us. Hence we will use only those hypothesis for which we are having data.

Data Cleaning and Exploration

Check for missing values

Prior to further analysis first check for missing values in loaded data. Following are missing values corresponding to each variable-

```
fare_amount          24
pickup_datetime       0
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     0
dropoff_latitude      0
passenger_count      55
dtype: int64
```

Check and change variable data types

Before dealing with missing values each variable should be in proper data type.

Initial data types.

```
fare_amount          object
pickup_datetime      object
pickup_longitude    float64
pickup_latitude     float64
dropoff_longitude   float64
dropoff_latitude    float64
passenger_count     float64
dtype: object
```

Change fare_amount to float and pickup_datetime to datetime.

After changing.

```
fare_amount                      float64
pickup_datetime      datetime64[ns, UTC]
pickup_longitude                 float64
pickup_latitude                  float64
dropoff_longitude                float64
dropoff_latitude                 float64
passenger_count                  float64
dtype: object
```

NOTE- passeger_count will be treated later in analysis.

Missing value analysis

Some values are misformatted and cannot properly converted in desired types and hence converted to NaN.These misformated values are very less and can be treated as NaN. Following are missing values and and missing value percentage-

```
fare_amount          25
pickup_datetime       1
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     0
dropoff_latitude      0
passenger_count      55
dtype: int64
```

```
fare_amount          0.155598
pickup_datetime      0.006224
pickup_longitude     0.000000
pickup_latitude      0.000000
dropoff_longitude    0.000000
dropoff_latitude     0.000000
passenger_count      0.342317
dtype: float64
```

Values which are missing in fare_amount (dependent variable) should be imputed with mean, median or knn (whichever is suitable) because deleting these missing values can be a loss of influential data.

Delete the missing value of pickup_datetime because we know that it is due to misformatted value and such value cannot be imputed also due to datetime type of variable.

Impute missing values of passenger_count with median of column because mean and knn imputation may give decimal results. However in further analysis we will see that this variable is not important for our business goal.

After performing above operations. Missing values corresponding to each variable is-

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
dtype: int64
```

Outlier Analysis and Data Extraction.

At first we will confine our train data within boundary box of locations in test data i.e. we will choose only those observations which are within extreme locations of test data. Following are extreme coordinates of test data. Locations outside these values will be considered as outlier locations and must be removed.

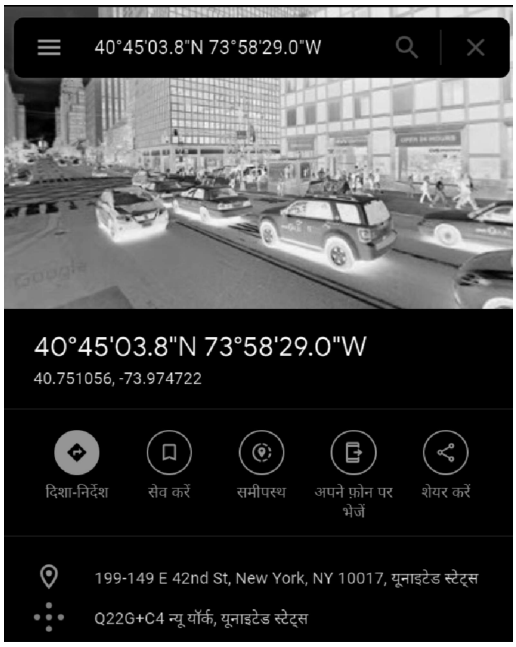
```
Minimum Latitude:     40.568973
Maximum Latitude:     41.709555
Minimum Longitude:  -74.263242
Maximum Longitude: -72.986532
```

Also we identify city with test data. Locating mean locations of test data.

```
(-73.97472222393064, 40.75104072348803)
```

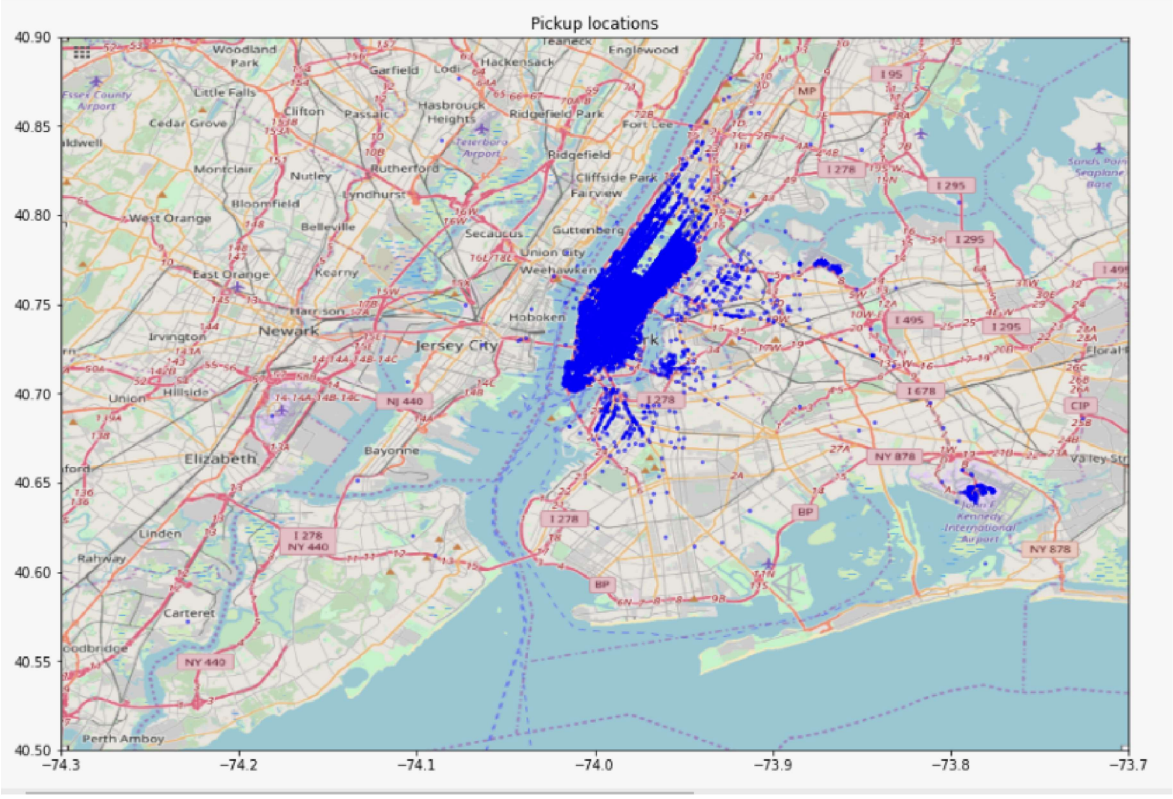
Locating above coordinates on google map, we know that city is NEW YORK.

We are going to predict cab fares for rides in New York. nyc = (-74.0063889, 40.7141667) #New York City center

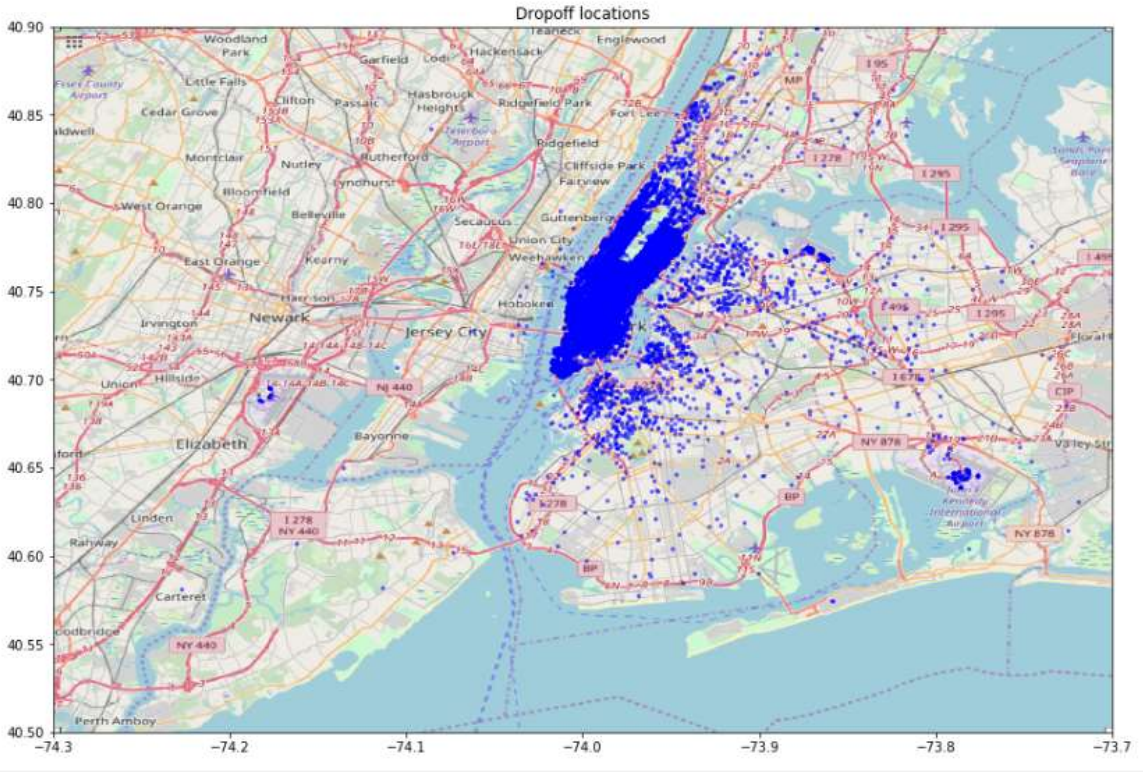


Negative fare_amount for rides is purely outlier value. Remove these observations from train data.

Plot pickup locations on city map.



Plot drop-off locations on map.

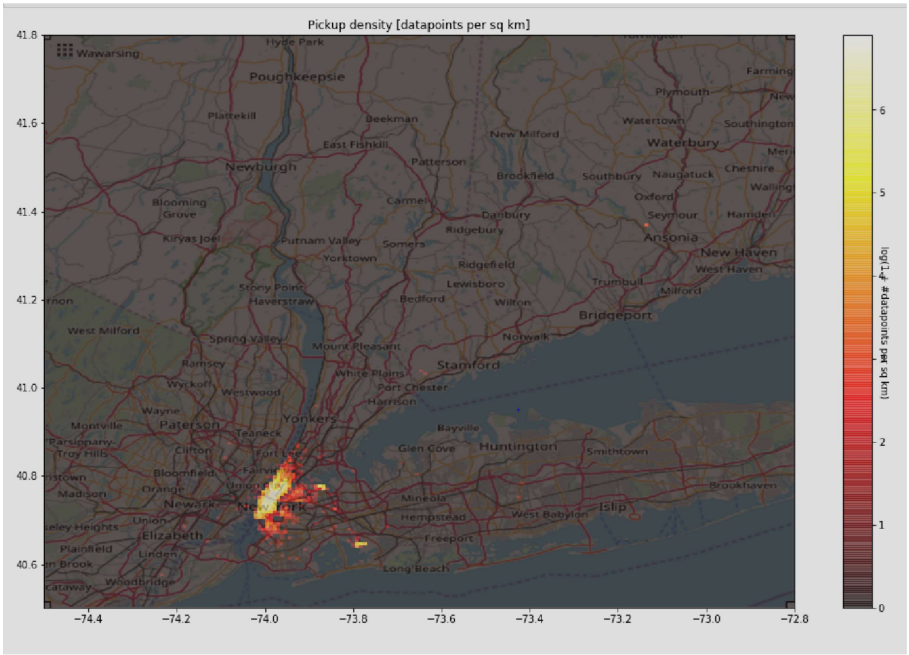


Extract important insights from pickup_datetime like pickup_hour, pickup_day, pickup_hour, pickup_day_of_week, pickup_month, pickup_year.

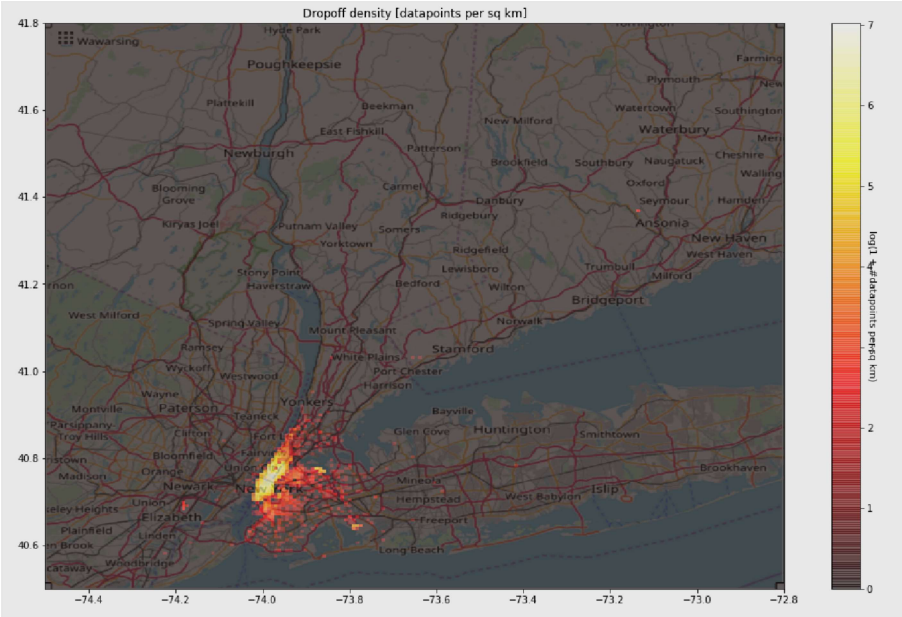
Calculate geographical trip distance between locations.

Next step is to know main pickup and dropoff locations of New York city . this can be done by plotting density of pickup and dropoff per sq. km. on city map.

Plot pickup density per sq. km.



Plot drop-off density per sq. km



Notice the scale on the side, which is log(1 + # of datapoints), meaning that it takes around 100-1000 rides within a given square KM to start making any kind of visible impact on this map.

Clearly observing above two images there are some hotspots in the visualisations. These may be the busiest travel locations like airports.

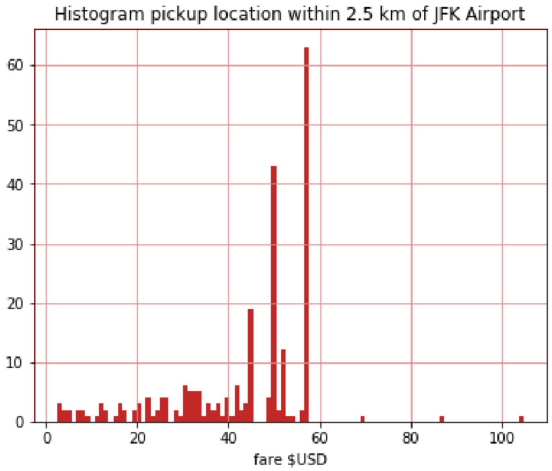
There are 3 major airports in New York.

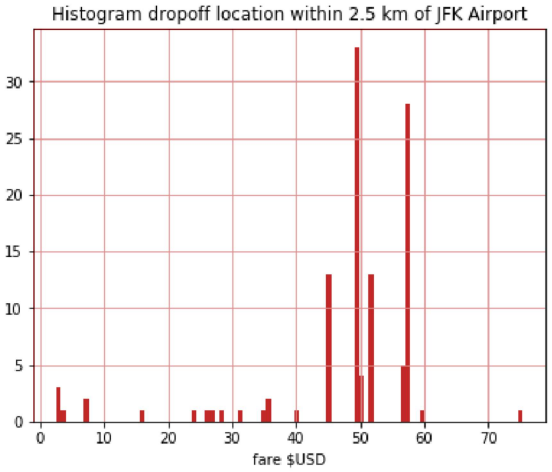
- JFK airport (-73.7822222222, 40.6441666667)
- LaGuardia Airport (-73.8719444444,40.7747222222)
- Newark Liberty International Airport (-74.175, 40.69)

Thus we look at fare_amount of rides which are having either drop off or pick up within 2.5 km range of airport.

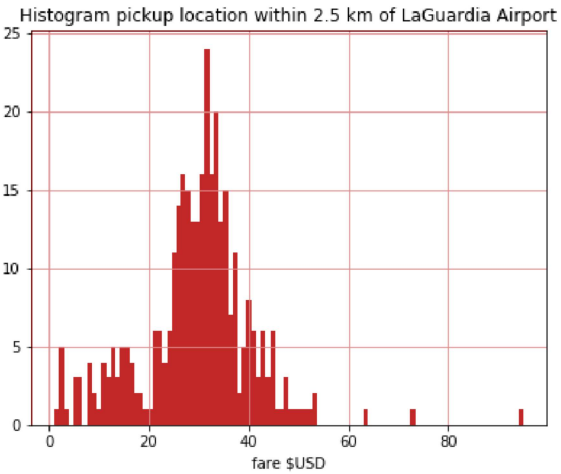
Plot Histograms for fare_amount of airport rides.

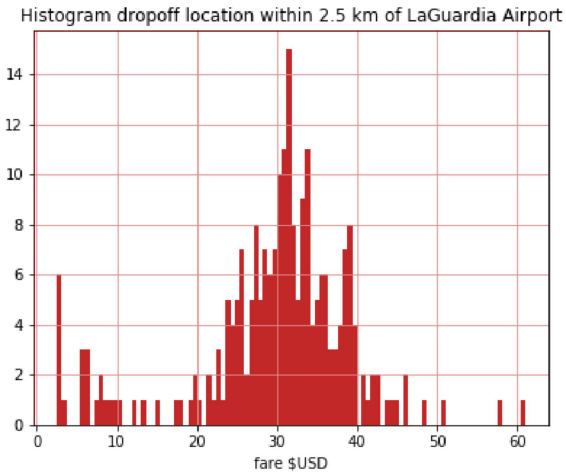
JFK AIRPORT.



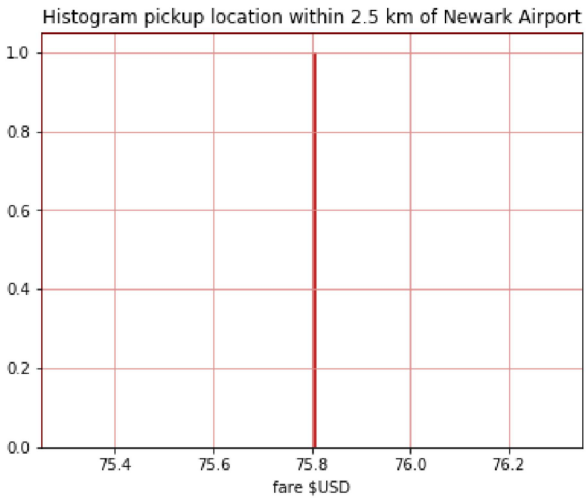


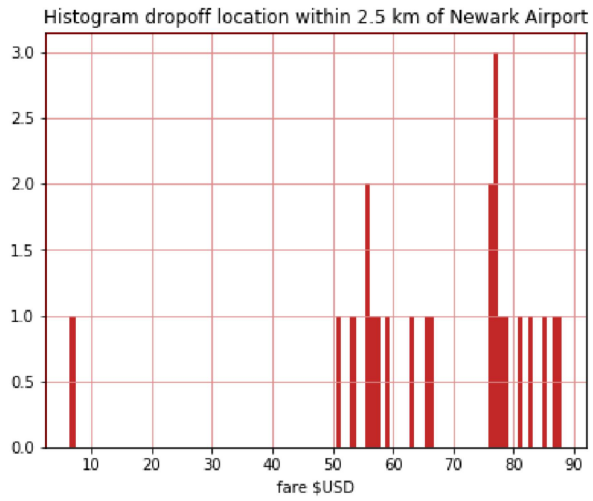
LAGUARDIA AIRPORT





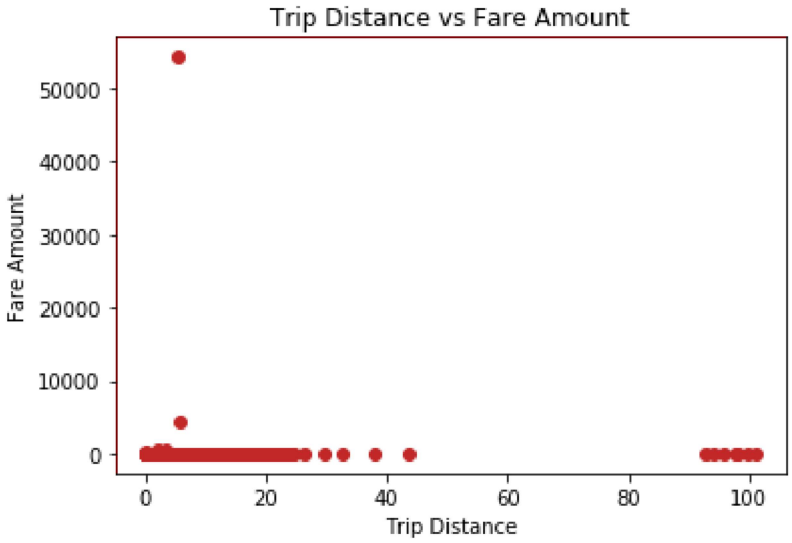
NEWARK AIRPORT





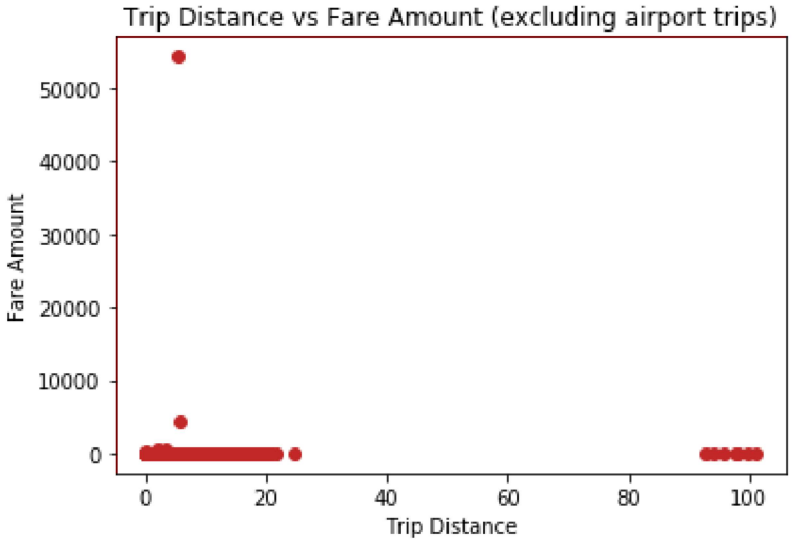
Airport rides have generally higher and fixed fare.Thus we create a new variables for dropoff and pickup to/from rides near range of 2.5km near any airport.

Plot trip distance v/s fare amount plot.



Closely observing on above plot we observed that some values of fare amount are more than 100$USD for non airport rides this is only possible for airport rides in New York. But If they are non-Airport rides than either we have to delete them or impute depending on their importance and no. of such trips

Checking whether they are airport trips or not.

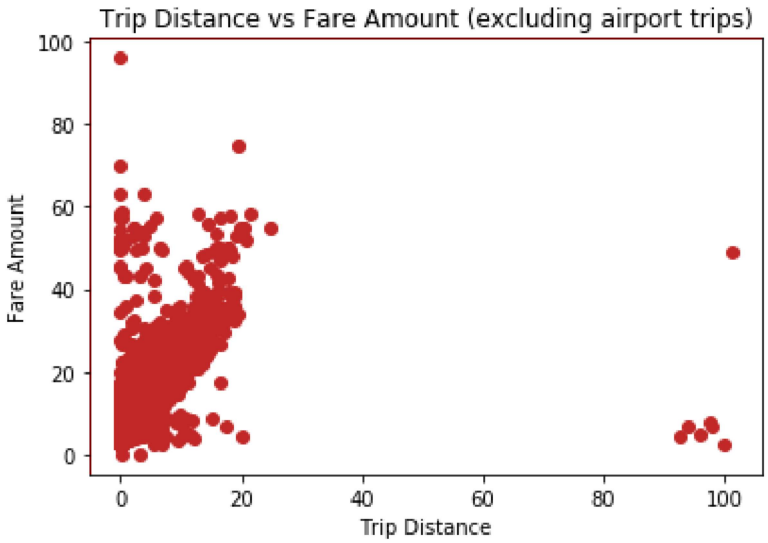


Looking on plot of non airport rides, we observed that these high fare rides are city rides and in New York City fare_amount>100$USD is not possible because in city fare rides are quite low than airport rides.

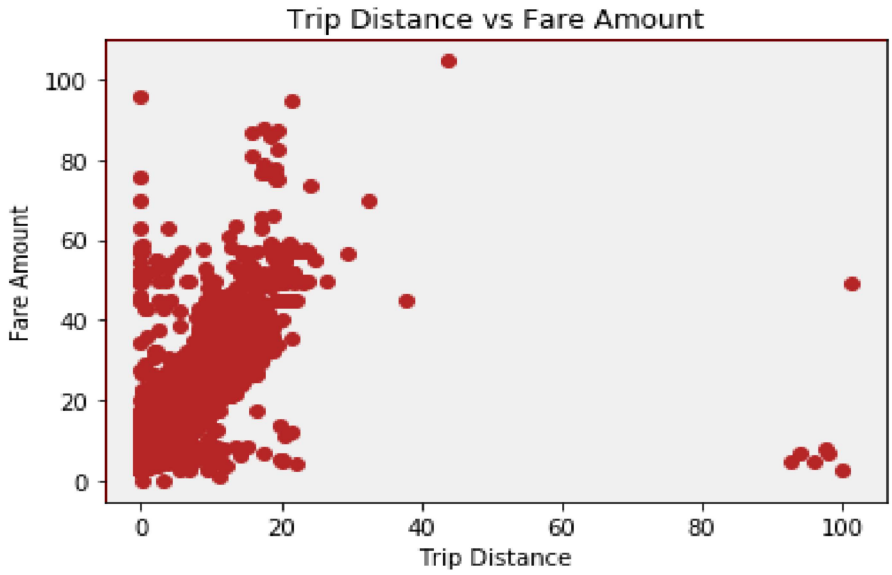
	fare_amount	distance_km
607	453.0	1.932338
980	434.0	3.295400
1015	54343.0	5.252040
1072	4343.0	5.744321
1483	165.0	0.028498
14142	108.0	3.829811

These non airport rides with abnormal fares for normal distance should be removed because they are only six such rides and fare_amount for these normal distance rides can be calculated with other available distances Hence these rides are very less and not important. This clearly states that these are outliers and I decided to remove them. Remove these abnormal values.

After removing these abnormal rides we plot a scatterplot between trip distance and fare amount of non airport rides.



After removing these abnormal rides we plot a scatterplot between trip distance and fare amount of all rides.



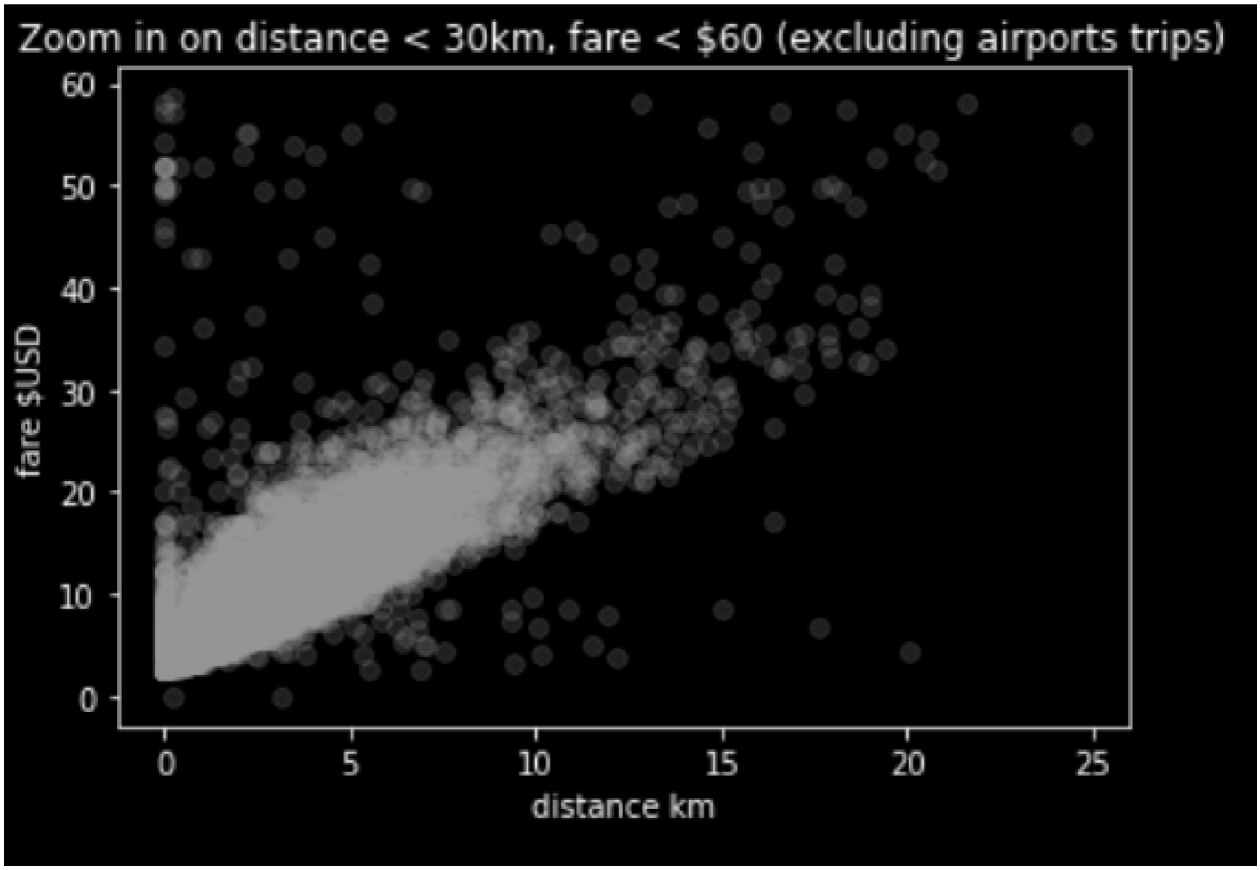
Observing above plots we observed some rides with longer distance but fer low fare.

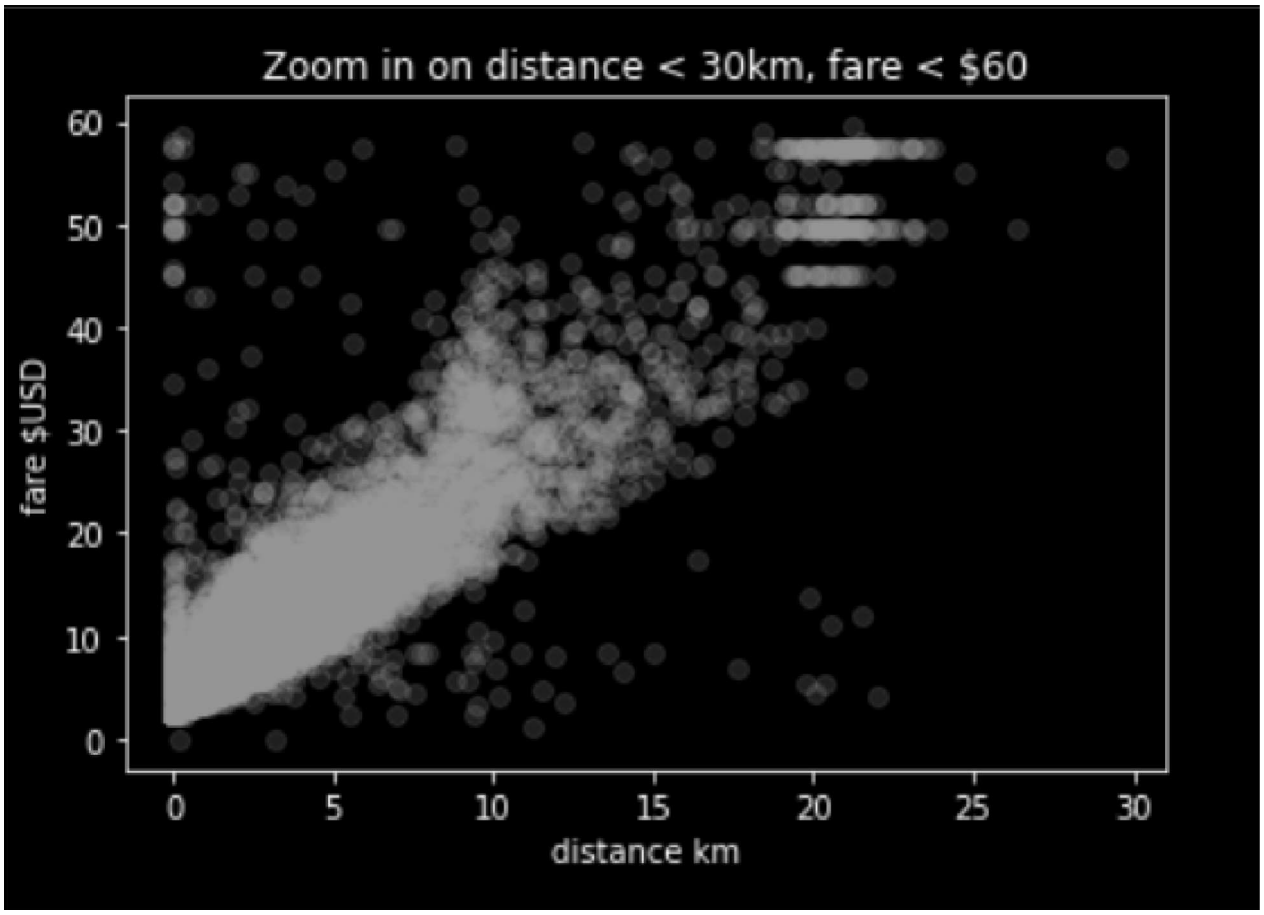
Now lookup on these rides which are longer than 80km and having fare less than 20$USD.

	fare_amount	distance_km	pickup_datetime
1684	2.5	99.802900	2009-05-02 19:01:01+00:00
3075	6.9	98.015848	2009-01-06 10:53:36+00:00
4487	4.9	95.882126	2009-08-26 07:43:16+00:00
7401	4.5	92.634919	2009-07-16 09:41:26+00:00
9808	6.9	93.955084	2009-08-13 23:15:28+00:00
9899	7.7	97.701251	2009-08-12 19:04:53+00:00

These are probably discounted trips by company. In Our hypothesis set we do not make any assumption about discount scheme. Still I am observing for other factors like special place, month and special occasions these points are very less that a firm result cannot be made. Thus I concluded to remove them from data.

Observing plots of distance v/s fare amount for fare<60$USD and distance<30 km





WE CAN CLEARLY SEE THE CONSTANT AND LARGE FARES FOR AIRPORT RIDES.

Treatment oF passenger_count

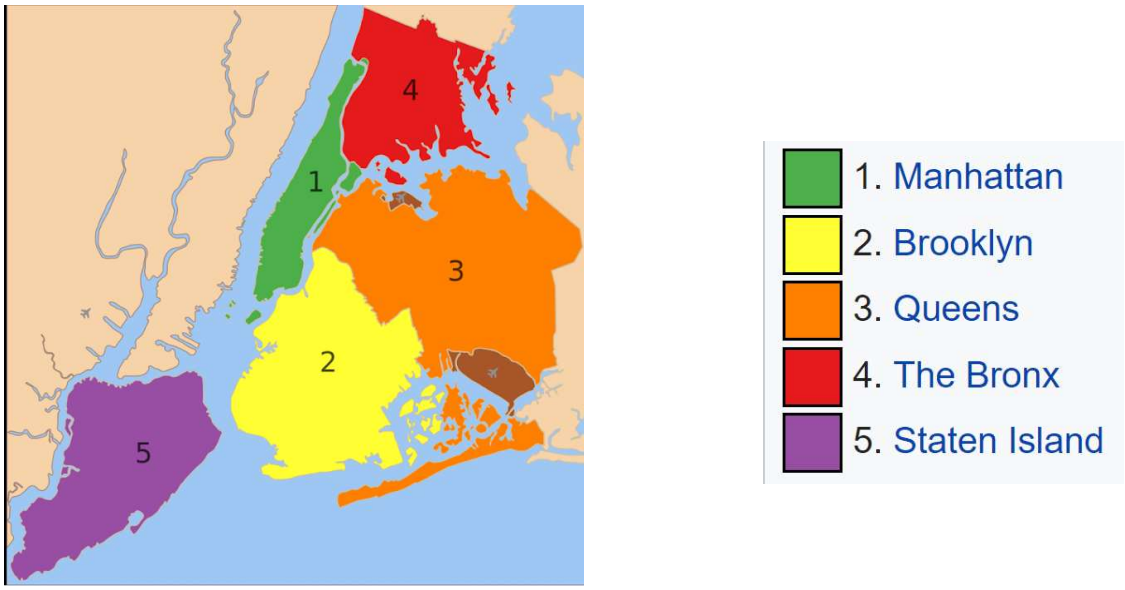
a normal cab company has minimum 1 and maximum 6 passengers in ride. Same here but according to our hypothesis set fare_amount does not depends on shared rides.Thus it is better to remove passenger count.

CONSIDERING CITY BOROUGHS IN MIND

5 city boroughs

New York city is divided into 5 Boroughs. Let us calculate which borough pickup and dropoff points are. And whether that affects the fare or not.

New York City encompasses five county-level administrative divisions called boroughs: The Bronx, Brooklyn, Manhattan, Queens, and Staten Island

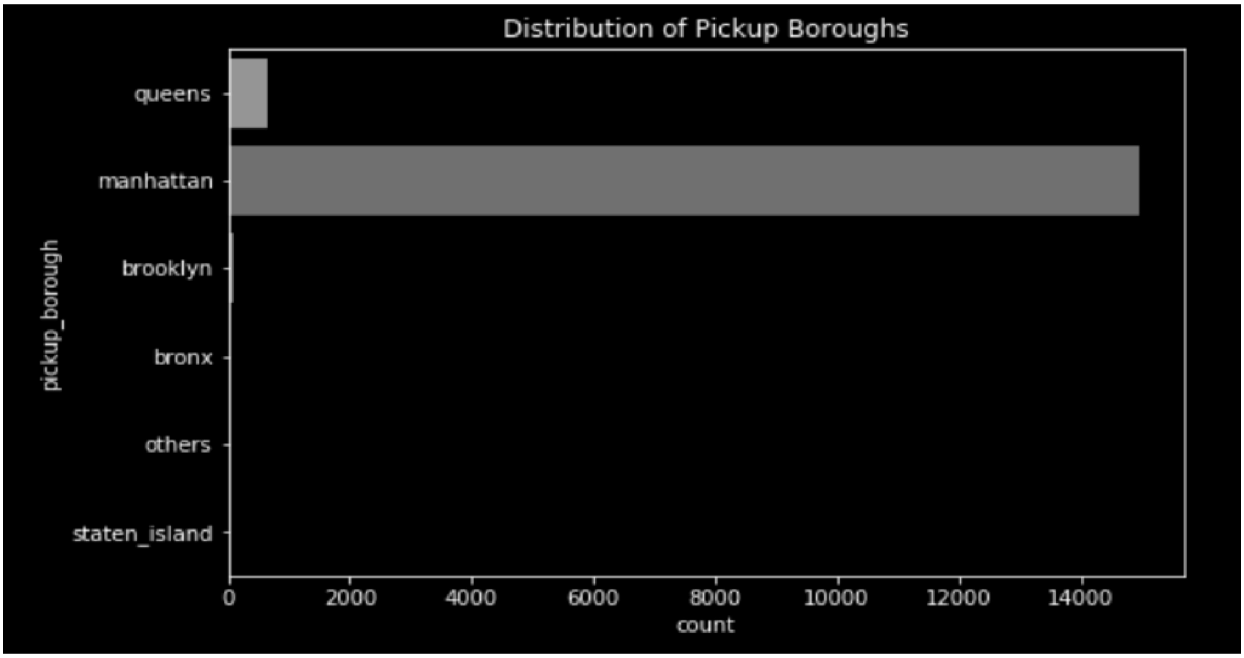


Boroughs	Minimum Longitude	Minimum Latitude	Maximum Longitude	Maximum Latitude
Manhattan	-74.0479	40.6829	-73.9067	40.882
Queens	-73.963	40.5431	-73.7004	40.8007
Brooklyn	-74.0421	40.5707	-73.8334	40.7395
Bronx	-73.9339	40.7855	-73.7654	40.9176
Staten_island	-74.2558	40.496	-74.0522	40.649

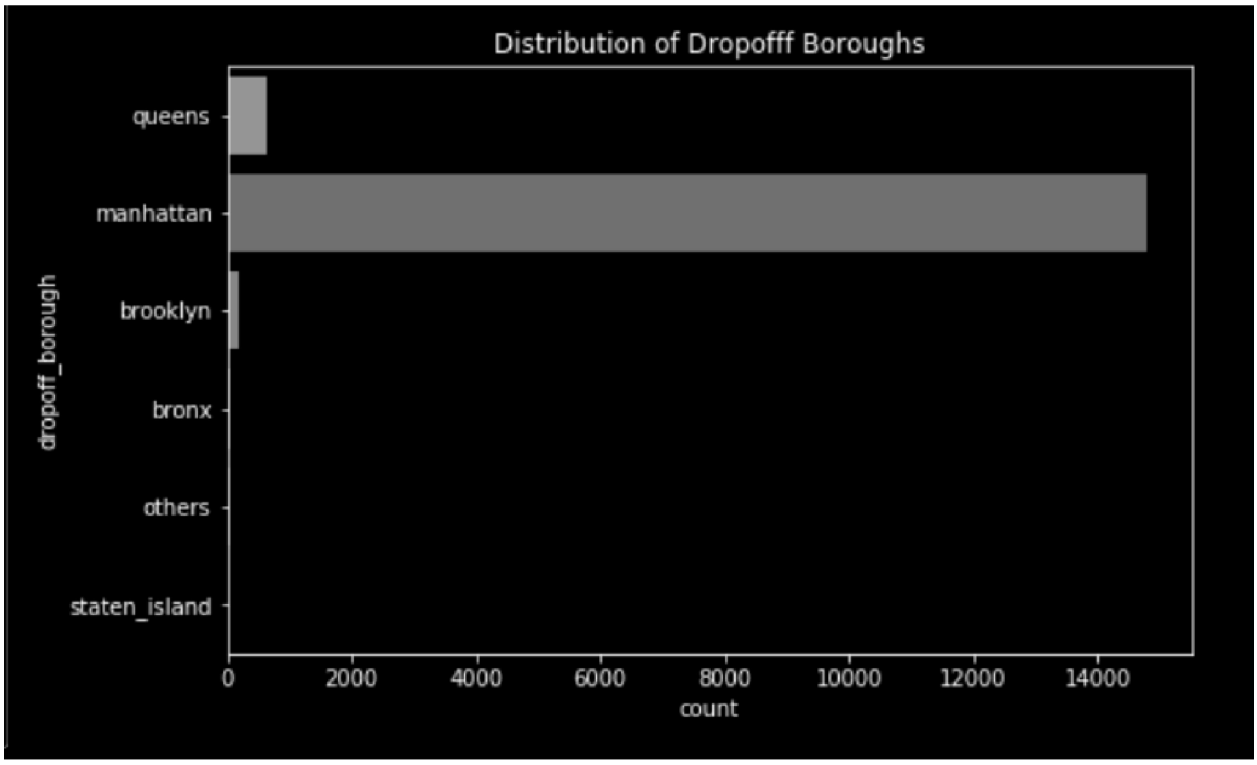
Create new variable for pickup_borough and dropoff_borough.

We are looking for these boroughs because these are the locations which decide fares of cab rides and have their respective traffic timings which affect fare of cab rides at different times.

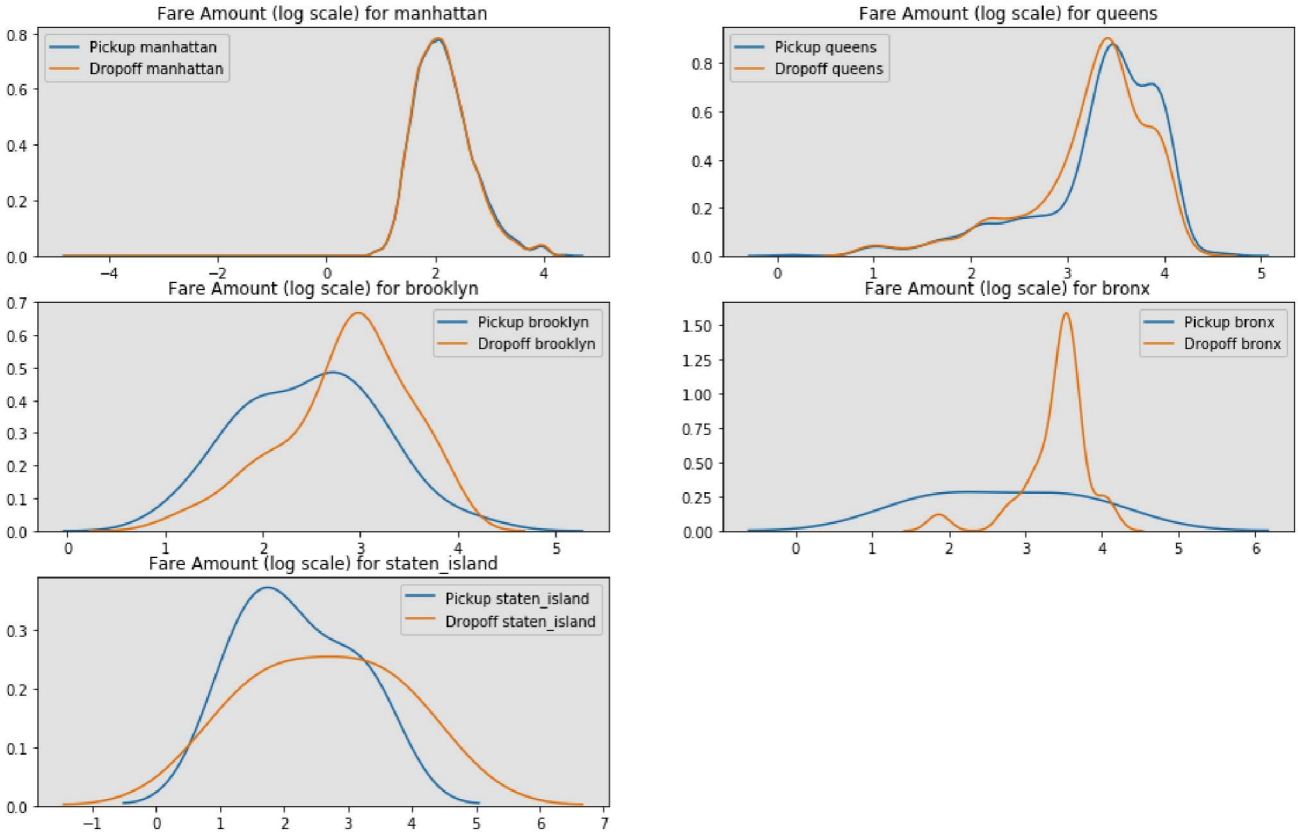
Distribution of Pickup boroughs.



Distribution of Drop Off boroughs.



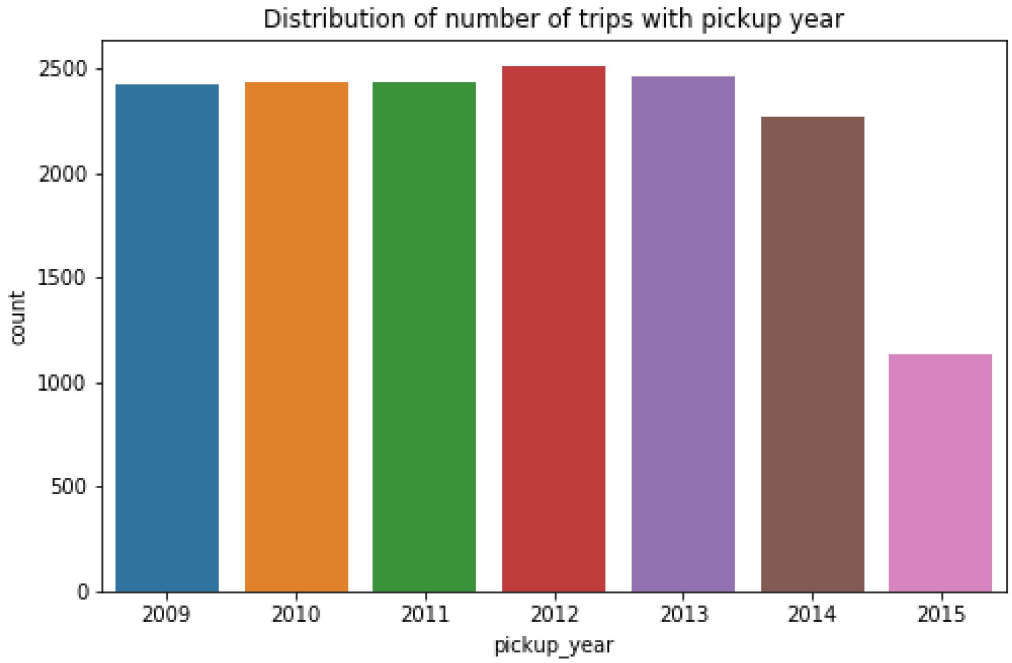
Plotting kernel density plots for fare amount in different boroughs.

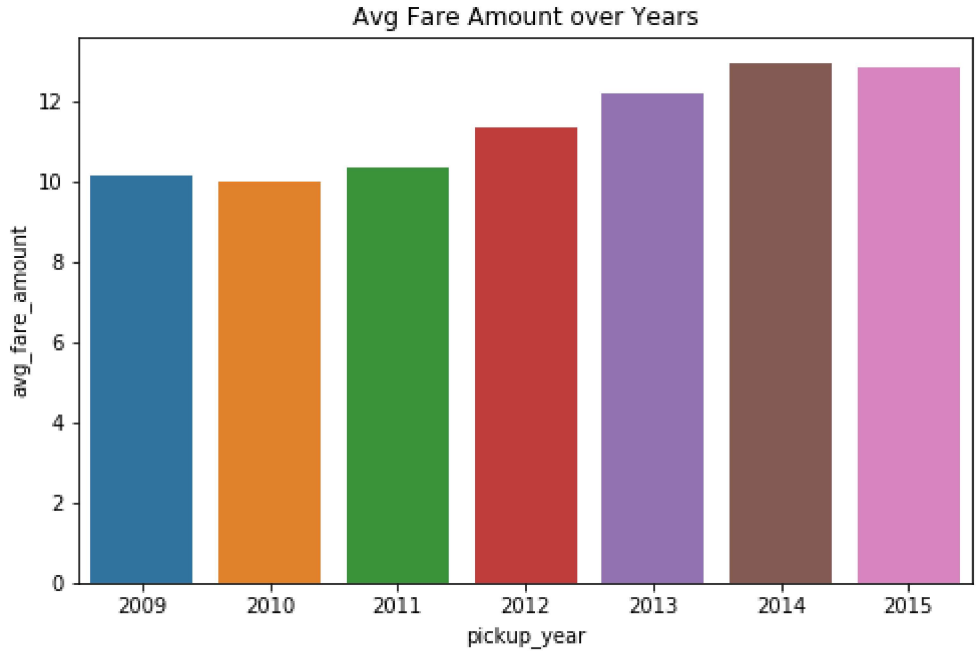


The next step was to check whether our hypothesis of fare from certain neighborhoods are higher than the rest, based on the 5 Boroughs New York city is divided each pickup and drop off location was grouped into these 5 neighborhoods. And yes our hypothesis was right- except for Manhattan which had most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

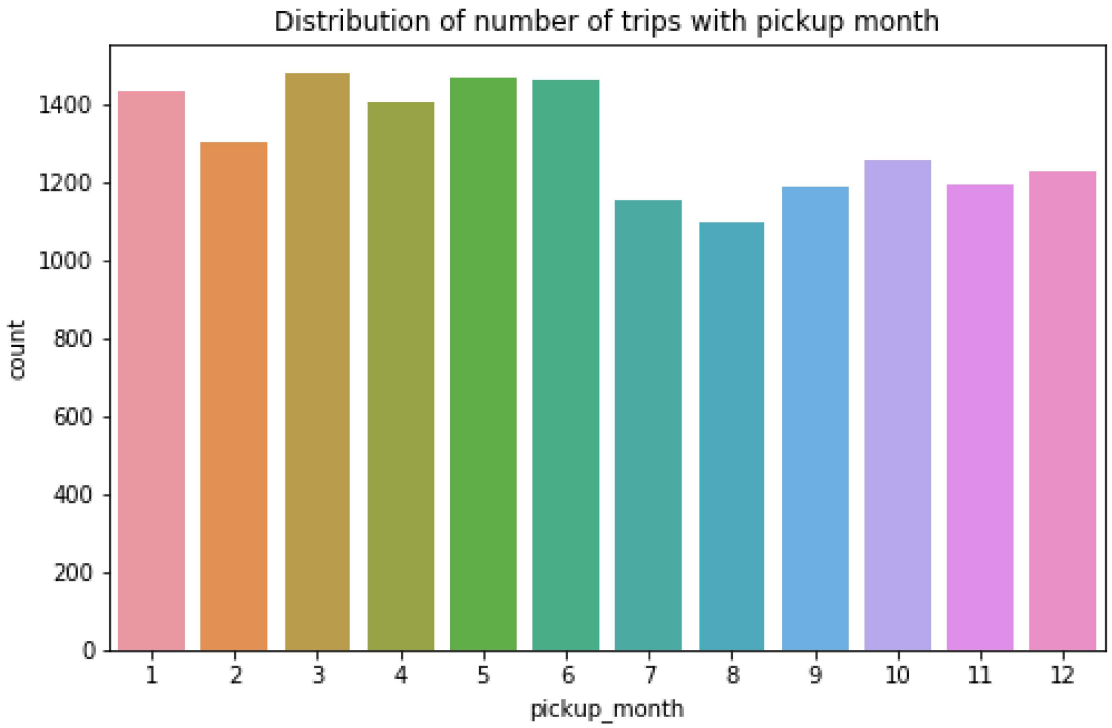
PLOTS FOR TRIP COUNTS AND AVERAGE FARE OVER YEARS, MONTHS, WEEKDAYS, HOURS.

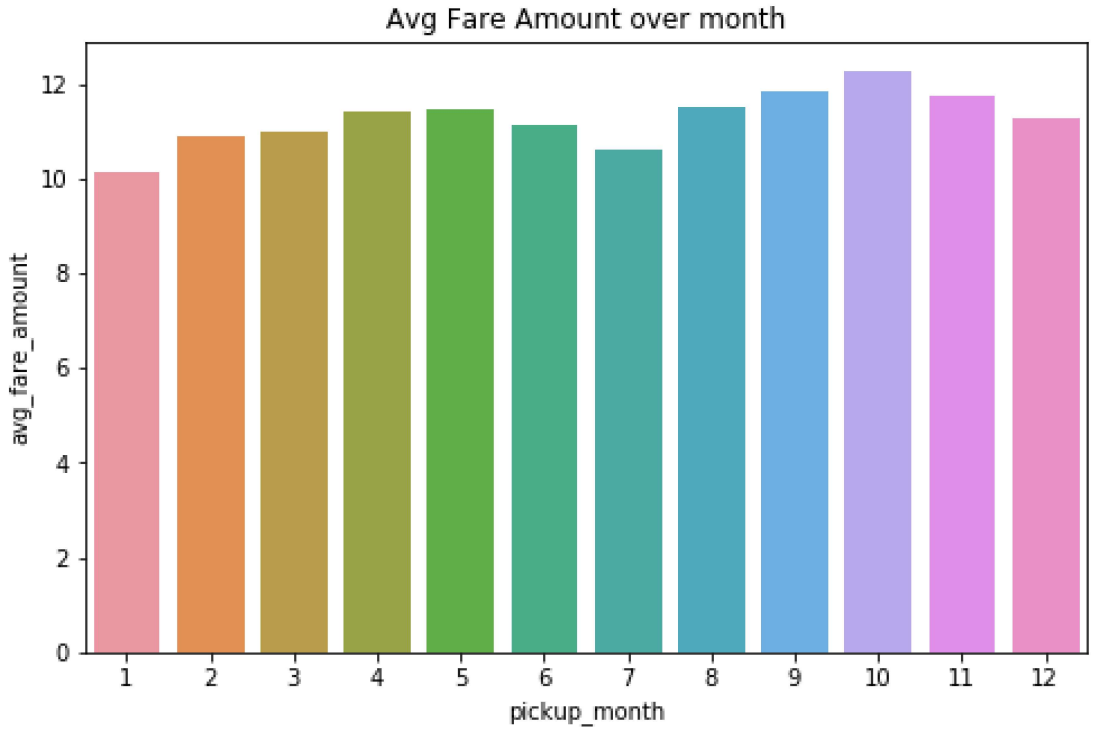
Distribution of number of trips and average amount with pickup year. Avg Fare amount has been increasing over the years



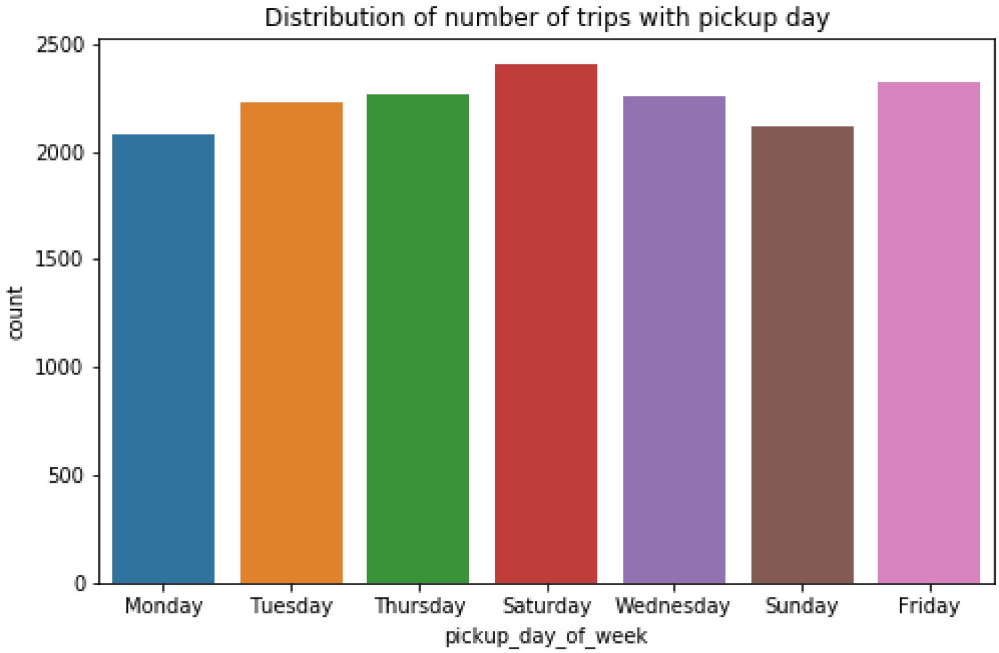


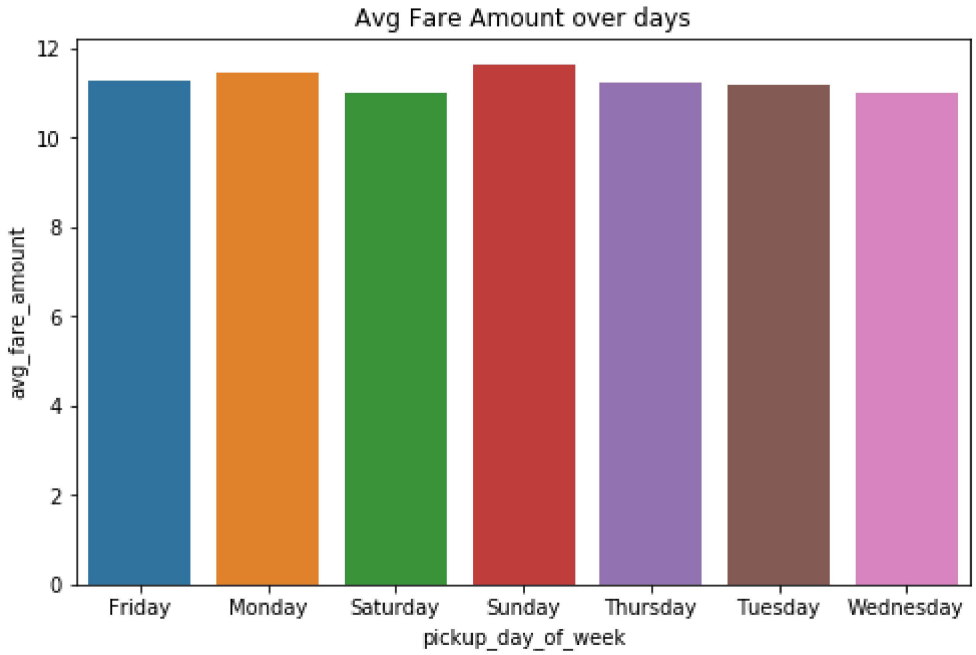
Distribution of number of trips and average amount with pickup month. Fares across months are fairly constant, though number of trips are lower from july to december.



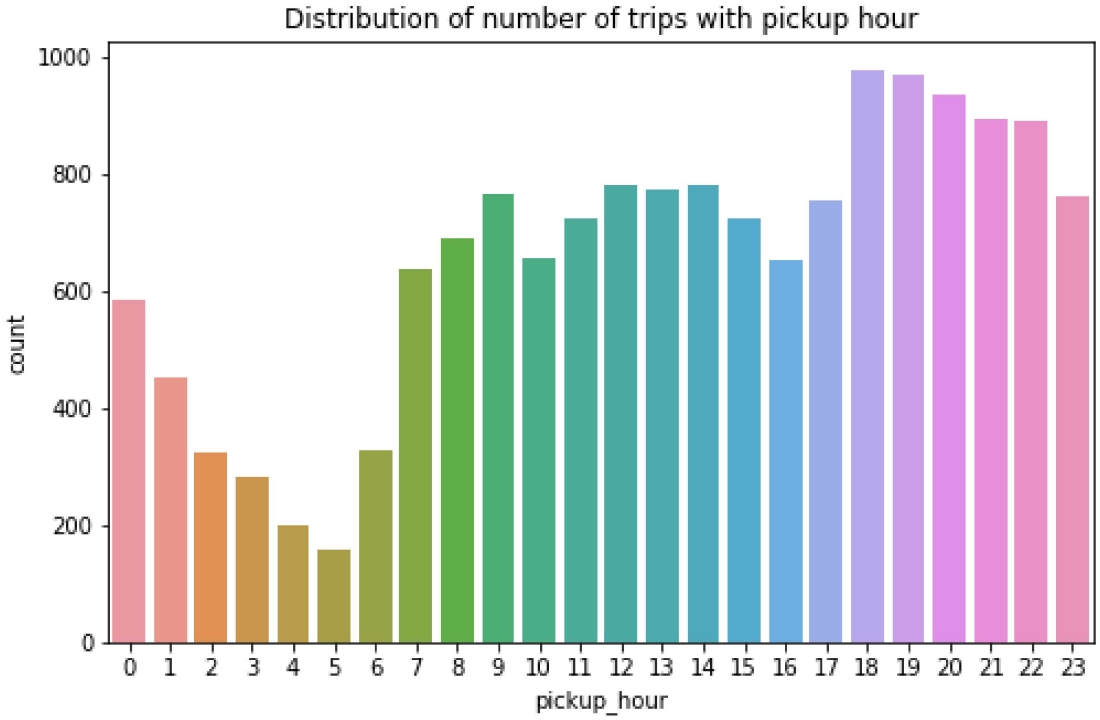


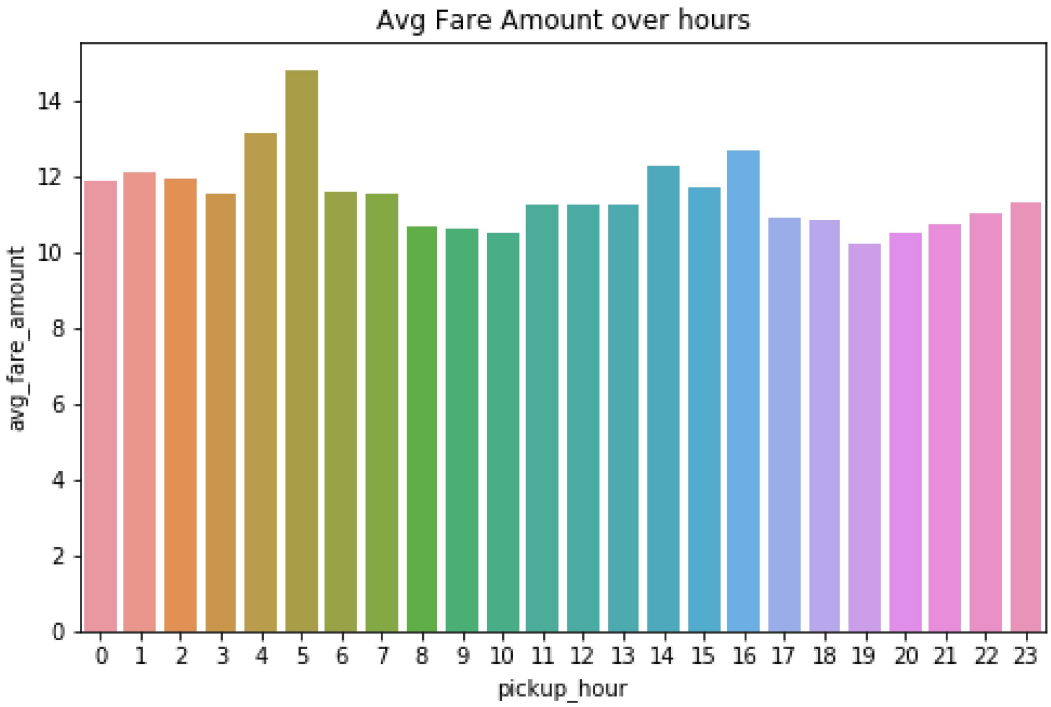
Distribution of number of trips and average amount with pickup day of week.Saturday has low avg fare amount, compared to other days though there are a lot of trips of saturday. On sunday and monday though the number of trips are lower, avg fare amount is higher.





Distribution of number of trips and average amount with hours.The avg fare amount at 5am is the higher while the number of trips at 5 am are the least. The number of trips are highest in 18 and 19 hours.





Encode weekdays in increasing order with Sunday as 0.

Finally train data is having 15698 observations and 21 variables.

Save this file to working directory as cleaned_train.csv

Now considering test data, perform following tasks-

- Change data types of variables if required.
- Extract date, day, hour, weekdays, month, year from pickup datetime.
- Encode weekdays in increasing order with Sunday as 0.
- Calculate trip distance in km.
- Extract pickup and drop off boroughs.
- Check for rides which are near airport i.e. within range of 2.5 km from airport.
- Delete passenger_count variable.
- Save this file to working directory as cleaned_test.csv.

Finally test data is having 9914 observations and 20 variables.

Here our part one of exploratory analysis and data cleaning is completed.

Feature Engineering, Modelling and Tuning

Building a Baseline Model

First we will design a baseline model for problem. A baseline model is a solution to a problem without applying any machine learning techniques. Any model we build must improve upon this solution. Some ways of building a baseline model are 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)— we will predict the average fare amount. For this baseline model we will only use following features.

```
pickup_longitude      float64
pickup_latitude       float64
dropoff_longitude     float64
dropoff_latitude      float64
pickup_day              int64
pickup_hour             int64
pickup_day_of_week      int64
pickup_month            int64
pickup_year             int64
dtype: object
```

This resulted in an RMSE of 9.812. So any model we build should have an RMSE lower than 9.812.

Building Model Without Feature Engineering

In this step, we will use only DateTime features, without including any of the additional features like the trip distance, or distance from airports, or pickup distance from a borough. To understand and evaluate the models, we will consider the following ML algorithms:

- Linear Regression
- Random Forest
- Light GBM

Except for Linear Regression, the rest of the models are ensembles of decision trees, but they differ in the way the decision trees are created.

Bagging: In this method, multiple models are created, and the output prediction is the average predictions of the different models. In Bagging, you take bootstrap samples (with replacement) of your data set and each sample trains a weak learner. Random Forest uses this method for predictions. In Random Forest, multiple decision trees are created, and the output is the average prediction by each of these trees. For this method to work, the baseline models must have a lower bias (error rate).

Boosting: In this method, multiple weak learners are assembled to create strong learners. Boosting uses all data to train each learner. But instances that were misclassified by the previous learners are given more weight, so that subsequent learners can give more focus to them during training. XGBoost and Light GBM are based on this method. Both of them are variations of the Gradient Boosting Decision Trees (GBDTs). In GBDTs, the decision trees are trained iteratively—i.e., one tree at a time. XGBoost and Light GBM use the **leaf-wise** growth strategy when growing the decision tree. When training each decision tree and splitting the data, XGBoost follows level-wise strategy, while Light GBM follows leaf-wise strategy.

Model 1: Linear Regression. It is used to find a linear relationship between the target and one or more predictors. The main idea is to identify a line that best fits the data. The best fit line is the one for which the prediction error is the least. This algorithm is not

very flexible, and has a very high bias. Linear Regression is also highly susceptible to outliers as it tries to minimize the sum of squared errors.

The test RMSE for Linear Regression model was 7.937, and the training RMSE was 7.856. This model is an improvement on the baseline prediction. Still, the error rate is very high in this model, though the variance is low (0.080695)

Reason for failure of logistic regression model, is that it tries to fit a linear line between the variables and the target. But, as we saw in the Exploratory analysis phase this is not true.

Model 2: Random Forest is far more flexible than a Linear Regression model. This means lower bias, and it can fit the data better. Complex models can often memorize the underlying data and hence will not generalize well. Parameter tuning is used to avoid this problem.

The Random Forest model gave an RMSE of 4.469 on validation data and train RMSE of 1.526.

There is a huge variation in the training and validation RMSE, indicating overfitting. To reduce overfitting, we can tune this model.

Model 3: LightGBM is a boosting tree based algorithm. The difference between Light GBM and other tree-based algorithms, is that Light GBM grows leaf-wise instead of level-wise. 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.

This model gave an RMSE of 4.866 on validation data

But the bias is higher than Random Forest. On the other hand, the variance of this model was 0.890 as compared to 2.943 in Random Forest model.

	model_name	test_rmse	train_rmse	variance
0	Linear Regression	7.937118	7.856423	-0.080695
1	Random Forest	4.469489	1.526057	-2.943432
2	Light GBM	4.866929	3.976304	-0.890625

Since Light GBM has a slightly more error rate than that of Random Forest, but as a lower variance and runs faster than the latter, we will use LightGBM as our model for further analysis. (Very high variance is a sign of overfitting).

Feature Engineering is the process of transforming raw data into features that are input to the final models. The aim is to improve the accuracy of the models. Having good features means we can use simple models for producing better results. Applying the same lightgbm model we get following results.

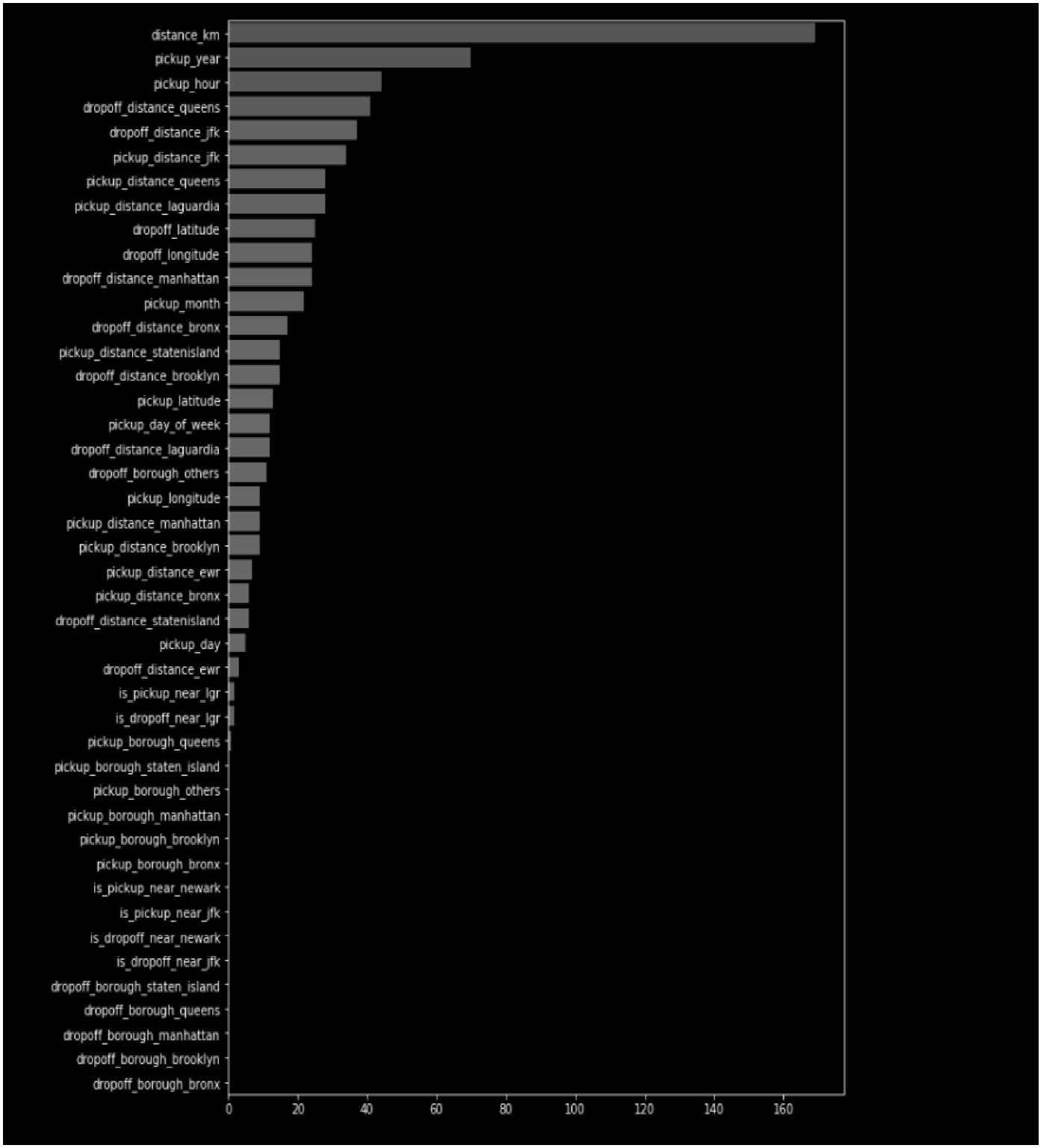
Light GBM model discussed above on data with Feature Engineering, the RMSE has decreased from 4.866 to 4.026. The variance of the model has also come down from 0.890 to 0.790.

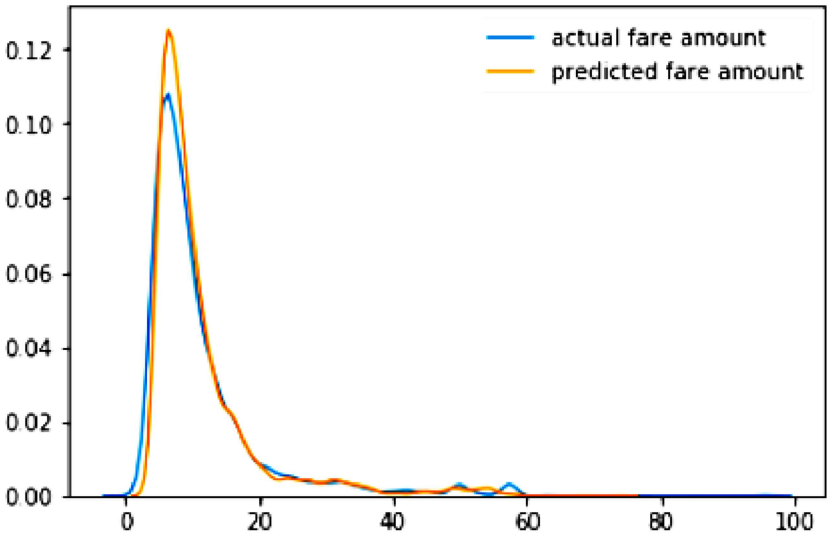
Tuning Light GBM

- **num_iterations**:It defines the number of boosting iterations to be performed.
- **num_leaves** :This parameter is used to set the number of leaves to be formed in a tree.In case of Light GBM, since splitting takes place leaf-wise rather than depth-wise, num_leaves must be smaller than 2^(max_depth), otherwise, it may lead to overfitting.
- **min_data_in_leaf** :A very small value may cause overfitting. It is also one of the most important parameters in dealing with overfitting.
- **max_depth**:It specifies the maximum depth or level up to which a tree can grow. A very high value for this parameter can cause overfitting.
- **Bagging_fraction**: It is used to specify the fraction of data to be used for each iteration. This parameter is generally used to speed up the training.
- **max_bin** : Defines the max number of bins that feature values will be bucketed in. A smaller value of max_bin can save a lot of time as it buckets the feature values in discrete bins which is computationally inexpensive.
- I am using gridsearchCV for obtaining best parameters.

After tuning best parameters we obtain RMSE 4.070 and variance 0.721.

Importance of features in tuned lightgbm.





Another way to improve the model's accuracy is to increase the amount of training data, and/or building ensemble models and tuning with hyperopt. If we have a lot of dimensions (features) in the data, dimensionality reduction techniques can also help improve the model's accuracy.