A

PROJECT REPORT

ON

"CAB Fare Prediction"

DESIGN AND DEVELOPED

BY

HARSH.J.JOSHI

UNDER THE GUIDANCE OF

PROF. Muquayyar Ahmed

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Now a day's cab rental services are expanding with the multiplier rate. The ease of using the services and flexibility gives their customer a great experience with competitive prices. By doing this we can help the customer with the fares from place to place.

1.1 Problem Evaluation

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.

So let's first understand the given problem, we have the historical data from pilot project and we have data analysis on that data for fare prediction of CAB in the city.

1.2 Understanding Data

Understanding the data is one of the most important and crucial task to perform. It is the first step for every Data Science and Data Analytic projects because by doing so we can easily analyse and give answers to the questions like what is data, Is data proper or not, Is data enough for performing any analysis methodologies.

Here in our case our company has provided a data set with following features, we need to go through each and every variable of it to understand data and for better functioning. We have been provided with two datasets train.csv and test.csv. As we see in the train data set we have 6 independent and 1 target variable

Size of Dataset provided: - 16067 rows, 7 Columns (including dependent variable)

Missing Values: Yes

Outliers Presented: Yes

Below mentioned is a list of all the variable names with their meanings:

Variables	Description
fare_amount	Fare amount
pickup_datetime	Cab pickup date with time
pickup_longitude	Pickup location longitude
pickup_latitude	Pickup location latitude
dropoff_longitude	Drop location longitude
dropoff_latitude	Drop location latitude
passenger_count	Number of passengers sitting in the cab

Fig 1.1:- Variable Data Chart

These are the variables given in dataset from which we have to come to a conclusion to predict CAB fares in a city.

Pickup_datetime telling us when the journey was stared like **2009-06-15 17:26:21 UTC** so, what we can do is we differentiate the above attribute in year, month, day of month, hour, minutes and second.

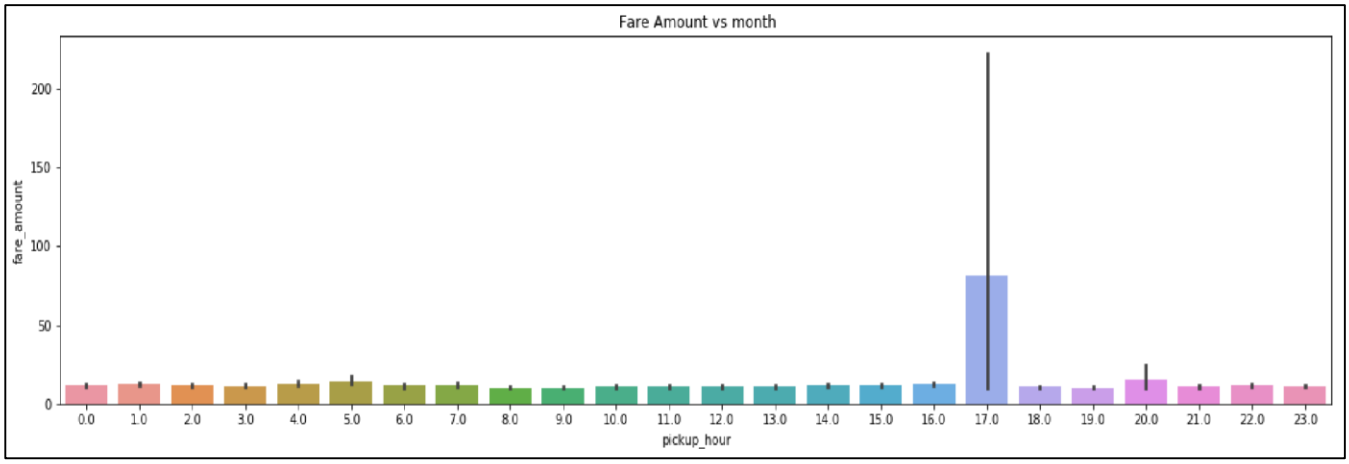


Fig 1.2 Pickup_hour_vs fare_amount

As we can see hour 17 is impacting more on the target variable. In this particular time people use more cab service. As we can notice in Fig 1.3 March, April, May, June fare amount is more as compare to other month and the Google survey says that those month have best weather to travel in the New York City. So, may be people from overall world and New York resident are use more cab service.

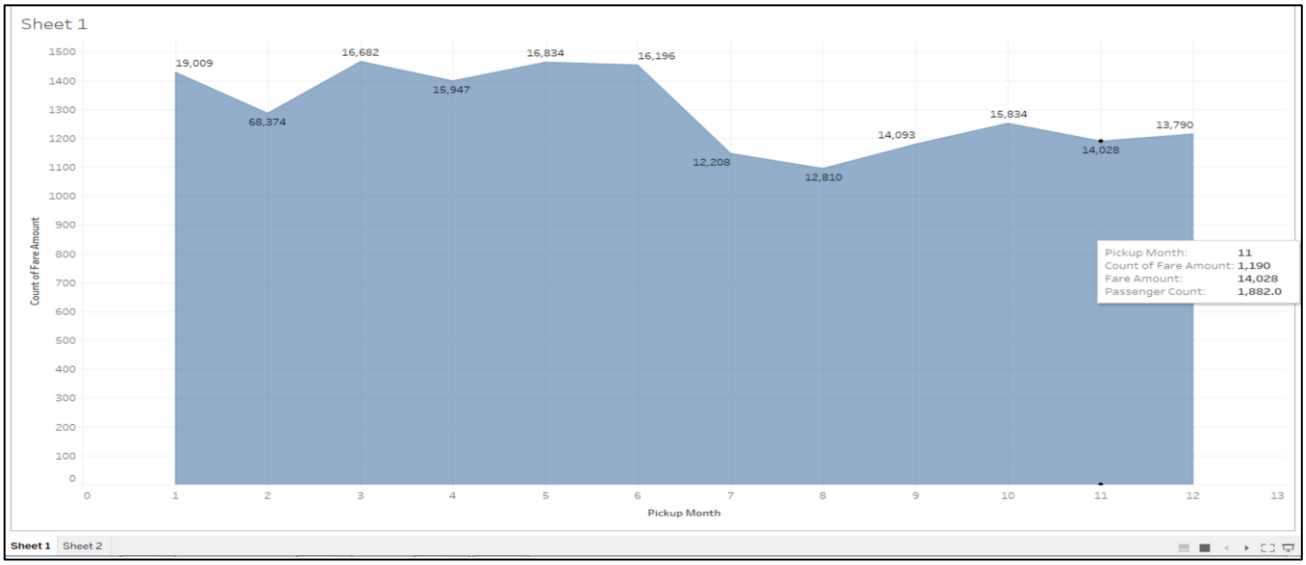


Fig 1.3 Pickup_year vs. Fare_amount

The pilot project data which we have received it is from 2009 to 2015 and as we can see in the below Fig.1.4 year 2015 has more fare_amount as compare to other year.

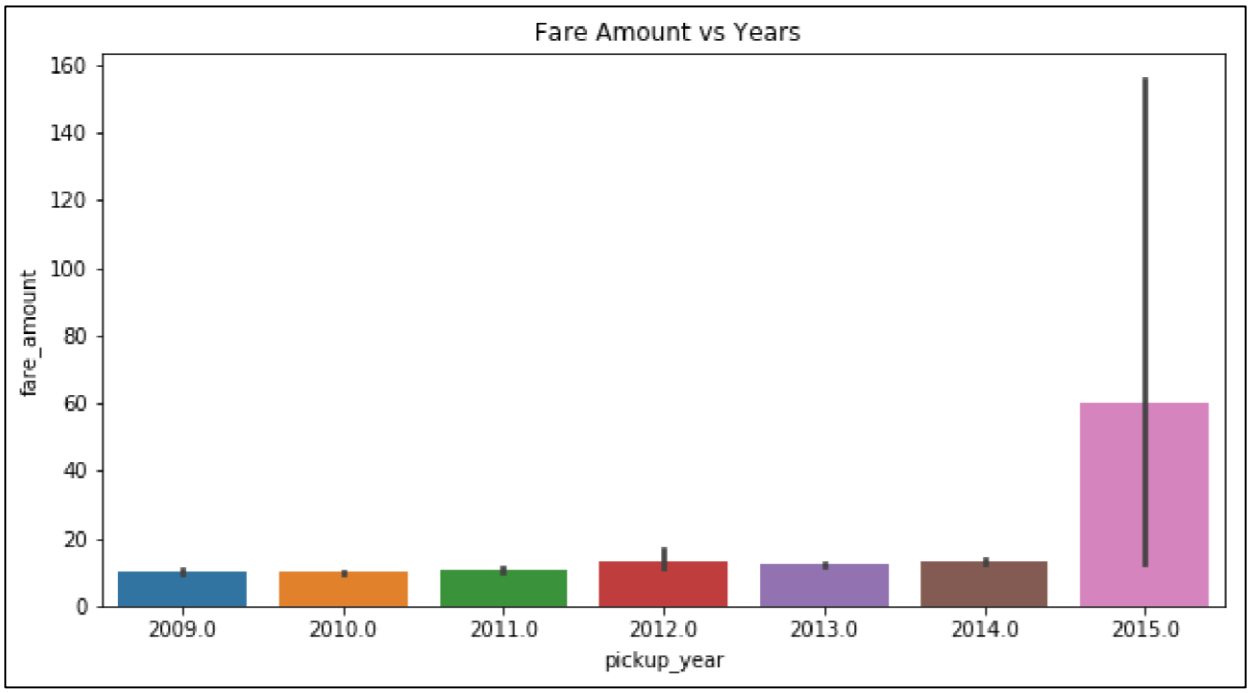


Fig 1.4 fare_amount vs years

Pickup and Dropout Location in our data set pickup latitude, pickup longitude, drop-off latitude and drop-off longitude telling us the pickup and drop-off location. With the help this

attributes here we can find the distance of trip. As we know the earth share is spherical or elliptical in nature so the 'haversine' formula help us to calculate the great-circle distance between two points that is, the shortest distance over the earth's surface distance. The formula to calculate the distance is shown below. :

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)})$$

$d = R \cdot c$ (φ is latitude in radian, λ is longitude, R is earth's radius (6,371km))

The latitude and longitude point has to be entered within the range latitude: [−90, 90] longitude: [−180, 180]. But in the below Fig.1.5 we can see latitude is 401.1 and longitude -73.9 also in some cases values of latlong is 0 it lies inside Atlantic ocean and value in the range of latitude 39.61607524 longitude 73.9644596 are also lies within water. Practically speaking, it is not possible because cab service is not available inside the water and ocean those values are nothing but outliers.

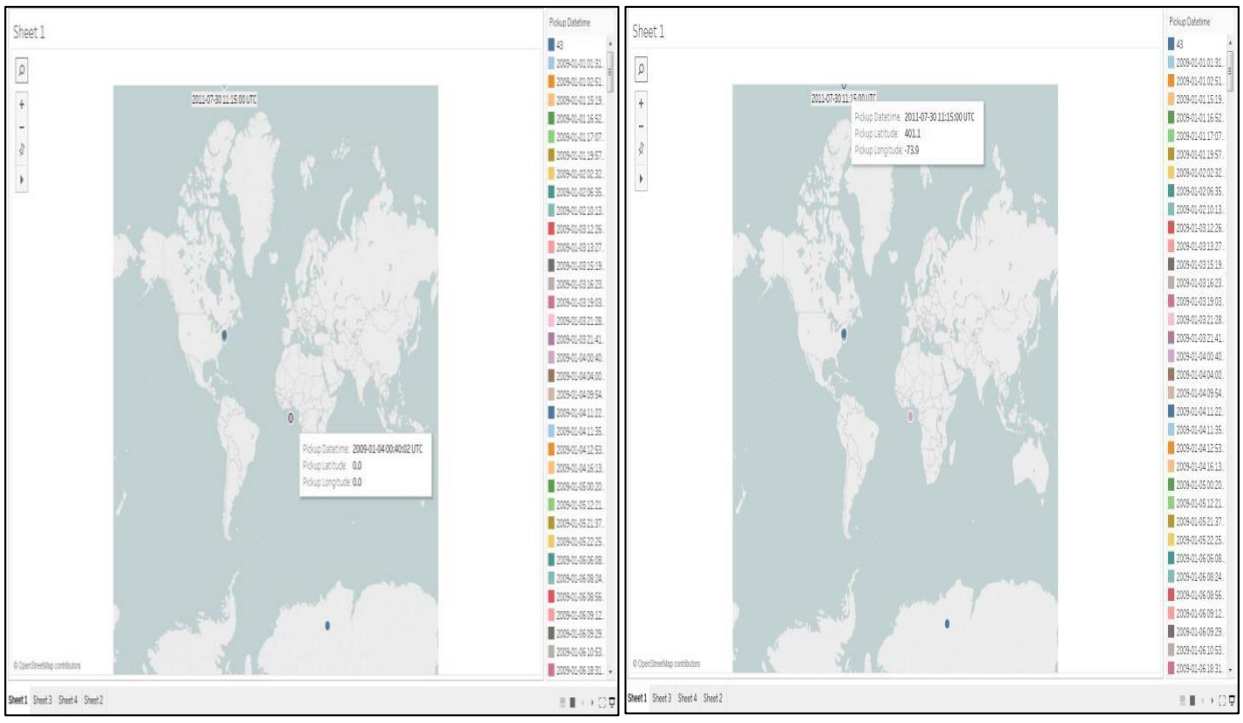


Fig 1.5 Latlong Visualization

In Fig.1.6 We plotted pickup time vs. location and we notice most frequent pickup is in airport area (**John Fitzgerald Kennedy and LaGuardia airport**). It means within the range of airport area cab frequency is more.

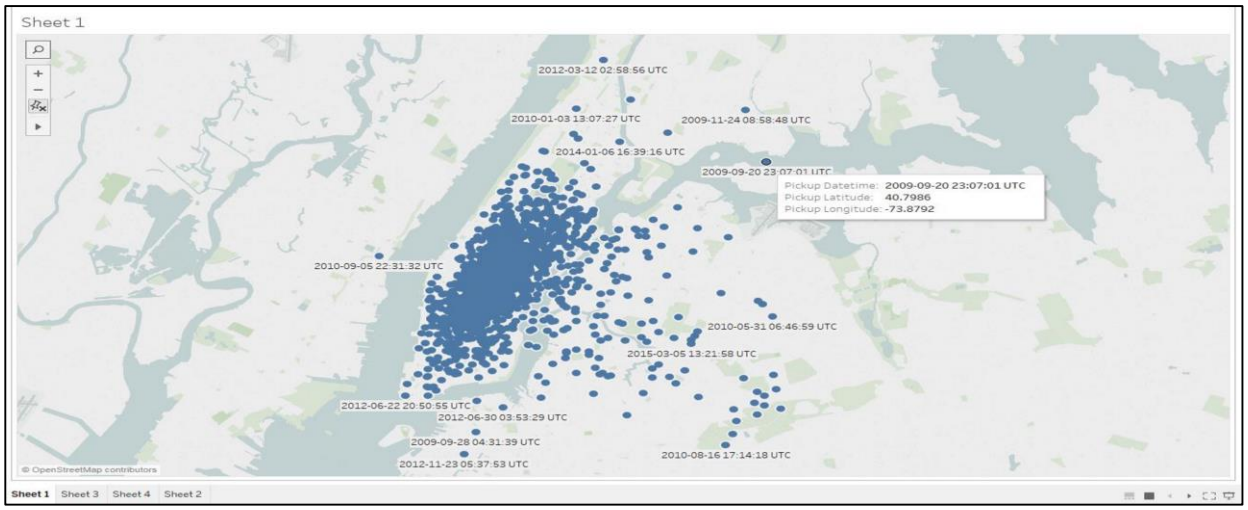


Fig 1.6 Pickup_time vs. Location

When we plot the values of **fare amount** we came to know few values are negative from fig 1.7 the min value is -3$ and max 54343$ but, as we know the cost of the journey cannot be negative so it is nothing but the impure values we must filter it out from our existing data set.

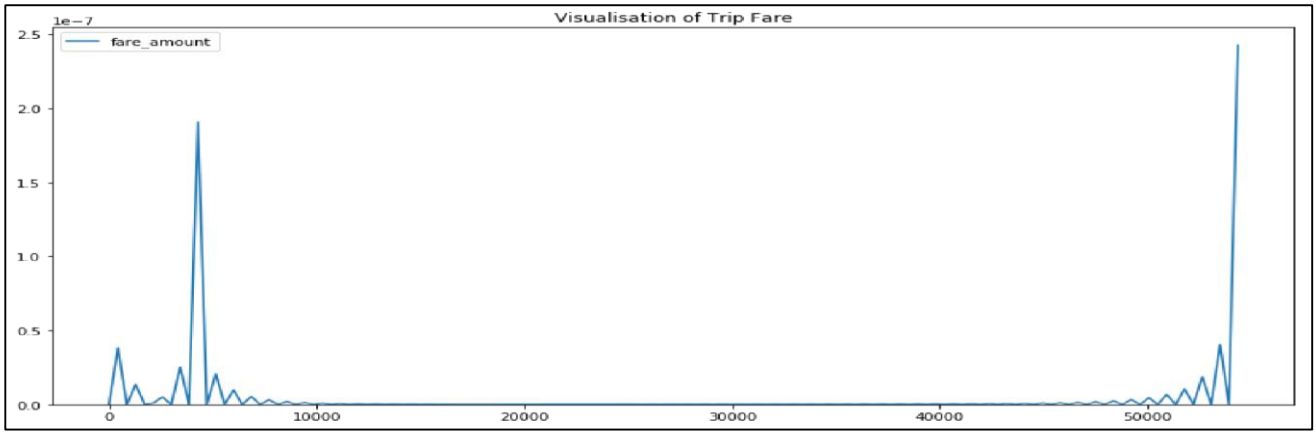


Fig 1.7 Fare Amount

In **Passenger_count** we can see the min value of passenger in single trip is 0 and maximum is 5345. Practically it is not possible because cab has desire setting capacity. Suppose we have 7 sitter cabs so max capacity is 6 passengers and 1 driver so in our data set beyond 6 we will consider as an outlier.

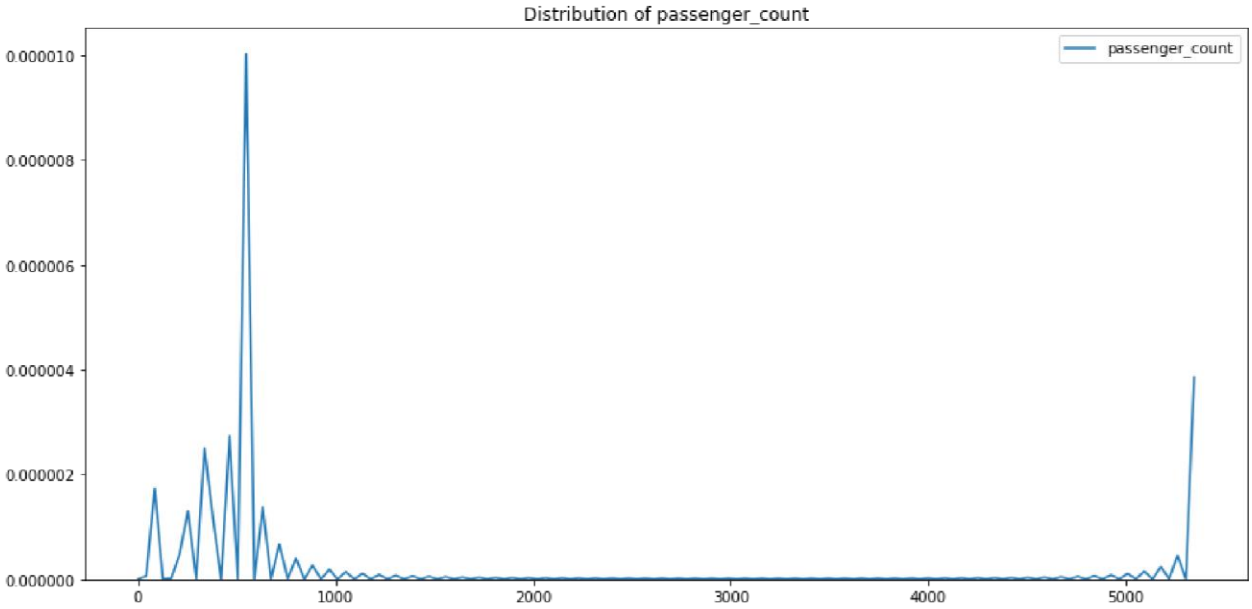


Fig 1.8 Passenger_count

1.3 Previous Assumption

As we are talking about how independent variables will be effect on the target variable. So there will be the multiple assumptions. It also helps to better understand the business process very efficiently so that we can properly come to a conclusion or to get to a final outcome very precisely.

1) Fare amount is highly depend on trip distance which we can calculate from pickup and dropout latitude and longitude.
2) Fare amount is depending on how much time it will take to travel from one place to another place. Because, in the traffic it may be take more time. So, indirectly it will effect on fare amount.
3) Pickup time is also impact on fare charges like suppose journey may be start in night time so night charges will be impact on fare amount.
4) Suppose any location from the New York City available multiple cabs so, it may be possible fare rate will be less.

CHAPTER 2

Data Pre-Processing

This step involves cleaning the data, dropping unwanted attributes, conversion of datatypes to machine-understandable format and when our unstructured data come up into structure format then finally we will split the training data into train and validation sets. Here we already removed few unstructured data from our training data set in above chapter but still few impurities are there we will figure it out using below methods.

2.1 Exploratory Data Analysis

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple preprocessing steps such as exploring the data, cleaning the data as well as visualizing the data through graph and plots, all these steps is combined under one shed which is Exploratory Data Analysis which includes following steps:

• Data exploration and Cleaning

• Missing values treatment

• Outlier Analysis

• Feature Selection

• Features Scaling

• Visualization

In the given project, we have training data set of cab fare in New York City, since from 2009 to 2015. The data which we have is unstructured in nature so, here we need to spend more time for data understanding, data cleaning, and data visualization to figure out new features that are better predictors of cab fare.

```
In [18]: #getting the train data into the python
         df = pd.read_csv("train_cab.csv")
         df #df = data frame
```

```
In [20]: #displaying first 5 data
         df.head(5)
```

Out[20]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1.0
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1.0
2	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2.0
3	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1.0
4	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1.0

Fig 2.1 loading train_data in python

It is very necessary to first know the data types of the given variables so that we can convert them to the type we want the variable to be for the further analysis. So after the data is perfectly loaded next step is to know the data types of the given variables.

```
In [22]: #knowing the data-types of given variables
         df.dtypes
```

```
Out[22]: fare_amount          object
         pickup_datetime      object
         pickup_longitude    float64
         pickup_latitude     float64
         dropoff_longitude   float64
         dropoff_latitude    float64
         passenger_count     float64
         dtype: object
```

Fig 2.2 Data-Types of Variables

In the above Fig.2.2 we can see the data-types of the given attributes. Here we need to change our data-types because pickup-date time which is indicating when the trip started it should be in timestamp. Fare_amount in dollar $ is nothing but our target variable it should be float. As we can see the target variable is not in test data set.

2.2 Missing Value Analysis

Missing value analysis plays a vital role in data preparing. There are many reasons to occur missing values. In statistics while calculating missing values, if it is more than 30% we just drop

the particular attribute because it does not carry much information to predict our target variables. As we can see in the below Fig2.3 highest percentage of missing value is **0.006418** and it is negligible but, then also we impute this missing value with the help of central statistic method.

```
missing_val
```

Out[22]:

	Var	Missing_percentage
0	pickup_year	0.006416
1	pickup_month	0.006416
2	pickup_day	0.006416
3	pickup_hour	0.006416
4	pickup_min	0.006416
5	pickup_sec	0.006416
6	fare_amount	0.000000
7	pickup_longitude	0.000000
8	pickup_latitude	0.000000
9	dropoff_longitude	0.000000
10	dropoff_latitude	0.000000
11	passenger_count	0.000000
12	Trip_distance_KM	0.000000

Fig 2.3 Missing value

2.3 Outlier Analysis

Outlier is the observation which is inconsistent related with all data set. The values of Outliers are the accurate but it is far away from the set of actual values and it heavily impact on the mean so that we consider it as an outlier. Here we have used box plot method to detect outliers as shown below Fig 2.4

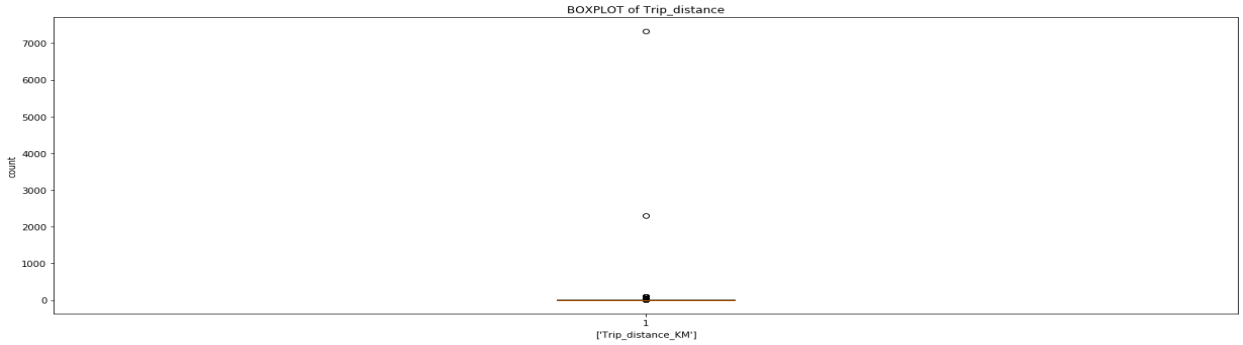


Fig 2.4 Trip_Distance_Km

Here what we did is we remove some outliers manually as discussed in chapter 1. But in some cases we find the extreme values and if we consider the same it will impact on mean. So we must have to remove it from our train data set. There are two method use to remove an outlier i.e. KNN and box plot here we used box plot method.

2.4 Feature Selection

As we know, while developing the model if we consider the independent variables which carries the same information to explain the target variables it will create the problem of multi-collinearity. So to avoid our model from the multi-collinearity problem we need to apply Feature Selection or dimensional reduction on the top of our data set. It helps us to sort out the variables which are highly correlated with each other. In our case we applied correlation analysis for numeric variables and ANOVA for the categorical variables.

In below Fig.2.4 pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude those all attributes are highly correlated with each other. It means those variables carry same information to explain the target variable actually in Chapter 1 we can drop those all variables after calculation of Trip distance but below with the help of visualization method we analyses it better.

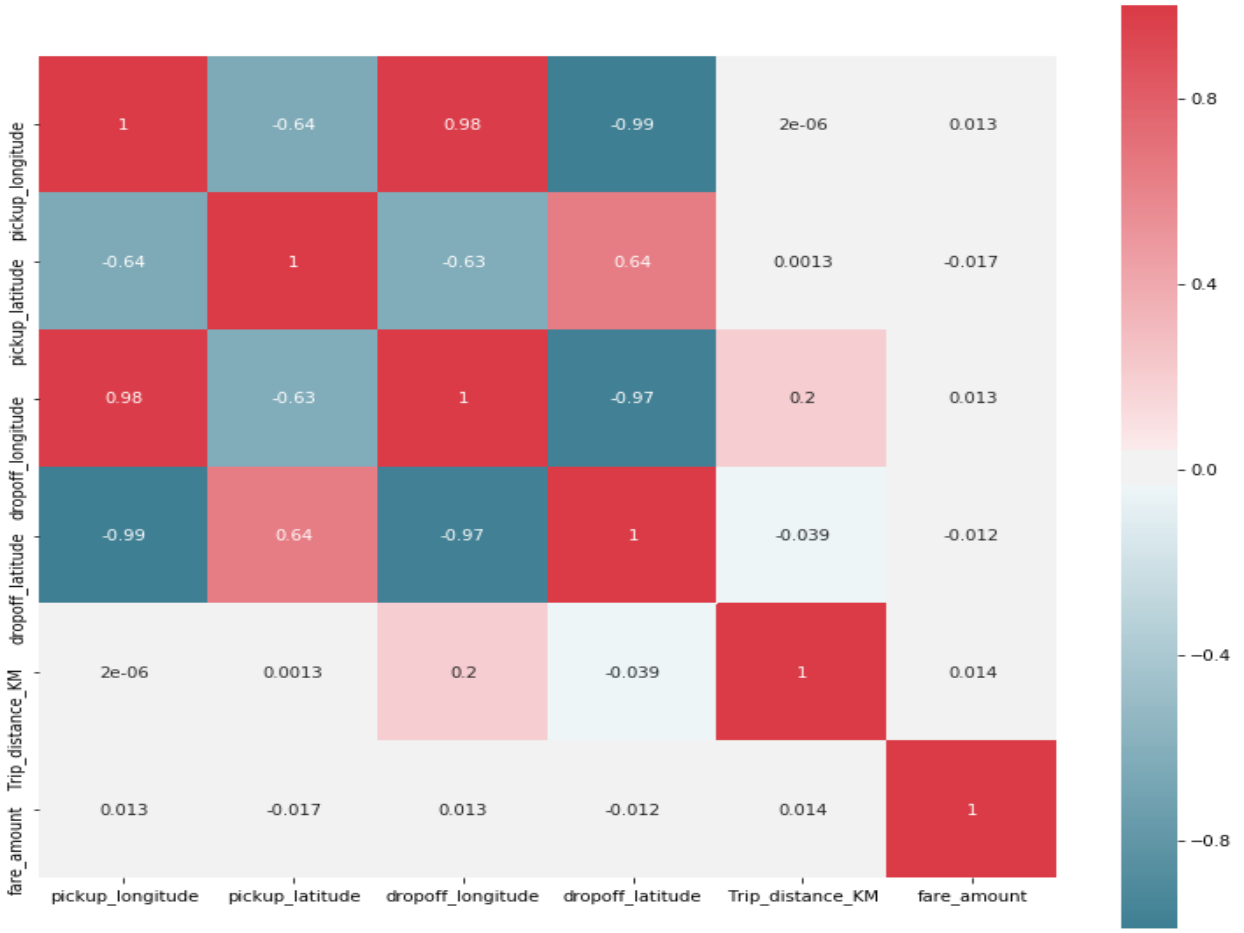


Fig 2.4 Checking Dependency

When we apply ANOVA test on categorical variables the p value of all the variables is 0 it means there no any dependency, as shown in Fig.2.5.

```
P value for variable pickup_year is 0.0
f value for variable pickup_year is 2771110369.070836
P value for variable pickup_month is 0.0
f value for variable pickup_month is 3522.2590979761903
P value for variable pickup_day is 0.0
f value for variable pickup_day is 7017.037323784967
P value for variable pickup_hour is 0.0
f value for variable pickup_hour is 5198.262636244764
P value for variable pickup_min is 0.0
f value for variable pickup_min is 19513.99833783456
P value for variable pickup_sec is 0.0
f value for variable pickup_sec is 1888.6855625584974
P value for variable passenger_count is 0.0
f value for variable passenger_count is 40125.39737083005
```

In [ ]:

Fig 2.5 Annova test

2.5 Feature Scaling

As we know before passing the data to machine learning algorithm our data should be structure in format. To arrange our data into the desire structure feature scaling technique comes into the picture. In this two methods are playing the important role i.e. normalization and standardization. If our data is normally distributed we go for normalization else for standardization. As we can see in Fig.2.6 our data has skewed in nature. So, here we applied normalization on continuous variables.

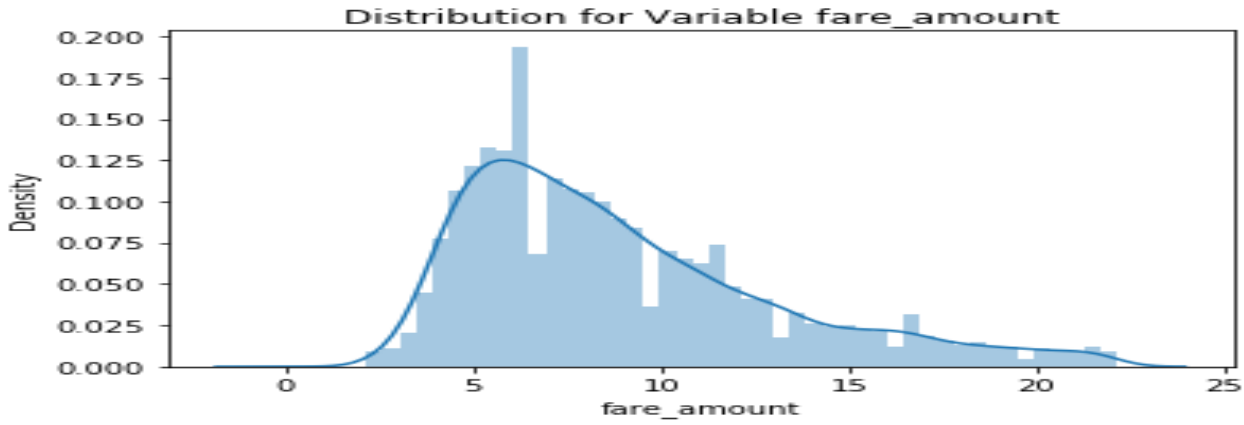


Fig 2.6 Normality of continuous variable

As we can see in our final train data set there are many variables which carry multiple classes. The variables like hour, minute, second, month, year, day and passenger. If we consider the same for model development, it will assume them as a numeric value so what we can do is we just differentiate every single class into separate variable with the help of Dummy variable method. A dummy variable is a numerical variable used in regression analysis to represent subgroups of the sample in your study. The dummy variables act like 'switches' that turn various parameters on and off in an equation.

CHAPTER 3

Modelling

After data cleaning and exploratory data analysis phase, we finally arrived at the model building phase. In this chapter we will applied multiple machine learning algorithm to predict the test case. In cab fare prediction project our target variable i.e. fare amount is numeric (predicting and forecasting type of problem) so that here we are using regression models on structure data to predict test case.

The next step is to differentiate the train data into 2 parts i.e. train and test. The splitting of train data into 2 parts is very important factor to verify the model performance and to understand the problem of over-fitting and under-fitting. Over-fitting is the term where training error is low and testing error is high and under-fitting is the term where both training and testing error is high. Those are the common problem of complex model.

In this analysis, since we are predicting fare amount which is the numeric variable. So, we come to know that, our problem statement is predicting (forecasting) type. So, what we can do is we will apply supervise machine learning algorithms to predict our target variable. As we know our target variable is continuous in nature so, here we will build regression matrix model.

Root Mean Square Error (RMSE) to measures how much error there is between two data sets. In other words, it compares a predicted value and an observed or known value. The RMSE is directly interpretable in terms of measurement units, and so is a better measure of goodness of fit. So, in our case any model we build should have lower value of an RMSE and higher value of variance i.e. R square.

Following are the models which we will built: –

- Linear Regression
- Decision Tree
- Random Forest
- Gradient Boosting

3.1 Linear Regression

Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as a predictive analysis, when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable). The main idea is to identify a line that best fits the data. To build any model we have some assumptions to put on data and model. This algorithm is not very flexible, and has a very high bias.

```
## Linear Regression Model :
# Building model on top of training dataset
fit_LR = LinearRegression().fit(X_train , y_train)
#prediction on train data
pred_train_LR = fit_LR.predict(X_train)
#prediction on test data
pred_test_LR = fit_LR.predict(X_test)
##calculating RMSE for test data
RMSE_test_LR = np.sqrt(mean_squared_error(y_test, pred_test_LR))
##calculating RMSE for train data
RMSE_train_LR= np.sqrt(mean_squared_error(y_train, pred_train_LR))
print("Root Mean Squared Error For Training data = "+str(RMSE_train_LR))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_LR))
#calculate R^2 for train data
from sklearn.metrics import r2_score
print(r2_score(y_train, pred_train_LR))
r2_score(y_test, pred_test_LR)
```

Fig 3.1 Linear Regression

3.2 Decision Tree

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

```
# Decision tree Model :
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(X_train,y_train)
#prediction on train data
pred_train_DT = fit_DT.predict(X_train)
#prediction on test data
pred_test_DT = fit_DT.predict(X_test)
##calculating RMSE for train data
RMSE_train_DT = np.sqrt(mean_squared_error(y_train, pred_train_DT))
##calculating RMSE for test data
RMSE_test_DT = np.sqrt(mean_squared_error(y_test, pred_test_DT))
print("Root Mean Squared Error For Training data = "+str(RMSE_train_DT))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_DT))
## R^2 calculation for train data
print(r2_score(y_train, pred_train_DT))
## R^2 calculation for test data
r2_score(y_test, pred_test_DT)
```

Fig 3.2 Decision Tree

3.3 Random Forest

Random forest is the collection of multiple decision trees. In Random Forest, output is the average prediction by each of these trees. For this method to work, the baseline models must have a lower bias. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. Random Forest uses bagging method for predictions. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important. To say it in simple words Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

```
# Random Forest Model :
fit_RF = RandomForestRegressor(n_estimators = 200).fit(X_train,y_train)
#prediction on train data
pred_train_RF = fit_RF.predict(X_train)
#prediction on test data
pred_test_RF = fit_RF.predict(X_test)
##calculating RMSE for train data
RMSE_train_RF = np.sqrt(mean_squared_error(y_train, pred_train_RF))
##calculating RMSE for test data
RMSE_test_RF = np.sqrt(mean_squared_error(y_test, pred_test_RF))
print("Root Mean Squared Error For Training data = "+str(RMSE_train_RF))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_RF))
## calculate R^2 for train data
print(r2_score(y_train, pred_train_RF))
#calculate R^2 for test data
r2_score(y_test, pred_test_RF)
```

Fig 3.3 Random Forest

3.4 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

```
# Gradient Boosting :
fit_GB = GradientBoostingRegressor().fit(X_train, y_train)
#prediction on train data
pred_train_GB = fit_GB.predict(X_train)
#prediction on test data
pred_test_GB = fit_GB.predict(X_test)
##calculating RMSE for train data
RMSE_train_GB = np.sqrt(mean_squared_error(y_train, pred_train_GB))
##calculating RMSE for test data
RMSE_test_GB = np.sqrt(mean_squared_error(y_test, pred_test_GB))
print("Root Mean Squared Error For Training data = "+str(RMSE_train_GB))
print("Root Mean Squared Error For Test data = "+str(RMSE_test_GB))
#calculate R^2 for test data
print(r2_score(y_test, pred_test_GB))
#calculate R^2 for train data
r2_score(y_train, pred_train_GB)
```

Fig 3.4 Gradient Boosting

So we have seen the different models that can be applied on the train data and decide which model to use on the data. Below is the table which shows the RMSE and R^2 values, considering this value we choose the model for our data.

Model Name	RMSE		R^2	
	Train	Test	Train	Test
Linear Regression	0.27	0.24	0.74	0.78
Decision tree	0.29	0.28	0.7	0.7
Random Forest	0.09	0.23	0.97	0.8
Gradient Boosting	0.22	0.22	0.81	0.82

Fig 3.5 Table of the Values of different models

Here we can clearly identify that Random Forest and Gradient Boosting values are most suited for the evaluation but to select any one we have to perform tuning on the parameters to evaluate the models more efficiently.

CHAPTER 4

Hyper- Parameters Tuning

Model hyperparameters are set by the data scientist ahead of training and control implementation aspects of the model. The weights learned during training of a linear regression model are parameters while the number of trees in a random forest is a model hyperparameter because this is set by the data scientist. Hyperparameters can be thought of as model settings. These settings need to be tuned for each problem because the best model hyperparameters for one particular dataset will not be the best across all datasets. The process of hyperparameter tuning (also called hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.

Here we have used two hyper parameters tuning techniques

➢ Random Search CV

➢ Grid Search CV

1. Random Search CV: This algorithm set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources.

2. Grid Search CV: This algorithm set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient.

We have performed the hyperparameters tuning on both the best fit models Random Forest and Gradient Boosting so that we can choose the best model for our data. For tuning the parameters, we have used Random Search CV and Grid Search CV under which we have given the range of n_estimators, depth and CV folds.

Fig no. 4.1 shows the result that have been achieved after hyperparameters tuning.

Model Name	Parameters	RMSE(Test)	R^2(Test)
Random Search CV	Random Forest	0.8	0.23
	Gradient Boosting	0.77	0.25
Grid Search CV	Random Forest	0.8	0.23
	Gradient Boosting	0.8	0.23

Fig 4.1 Hyperparameter tuning table

Above table shows the results after tuning the parameters of our two best suited models i.e. Random Forest and Gradient Boosting.

CHAPTER 5

Model Selection

In above chapters we applied multiple preprocessing to frame our data into the structural format and different machine learning algorithm to check the performance of model. In this chapter we finalize one of them. Above model help us to calculate the Root Mean Square Error (RMSE) and R-Squared Values. RMSE is the standard deviation of the prediction errors. Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. RMSE is an absolute measure of fit. R-squared is a relative measure of fit. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. Value of R-squared is between 0-1, where 0 means independent variable unable to explain the target variable and 1 means target variable is completely explained by the independent variable. So, Lower values of RMSE and higher value of R-Squared Value indicate better fit of model.

5.1 Model Evaluation

The main concept of looking at what is called residuals or difference between our predictions f(x[I,]) and actual outcomes y[i].

In general, most data scientists use two methods to evaluate the performance of the model:

I. RMSE (Root Mean Square Error): is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

II. R Squared(R^2): is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains as to how much of the variance of the target variable is explained.

III. We have shown both train and test data results, the main reason behind showing both the results is to check whether our data is overfitted or not.

5.2 Model Selection

On the basis RMSE and R Squared results a good model should have least RMSE and max R Squared value. So, from above tables we can see:

• From the observation of all RMSE Value and R-Squared Value we have concluded that,

• Both the models- Gradient Boosting Default and Random Forest perform comparatively well while comparing their RMSE and R-Squared value.

• After this, I chose Random Forest CV and Grid Search CV to apply cross validation technique and see changes brought about by that.

• After applying tunings Random forest model shows best results compared to gradient boosting.
• So finally, we can say that Random forest model is the best method to make prediction for this project with highest explained variance of the target variables and lowest error chances with parameter tuning technique Grid Search CV.

Finally, I used this method to predict the target variable for the test data file shared in the problem statement. Results that I found are attached with my submissions.

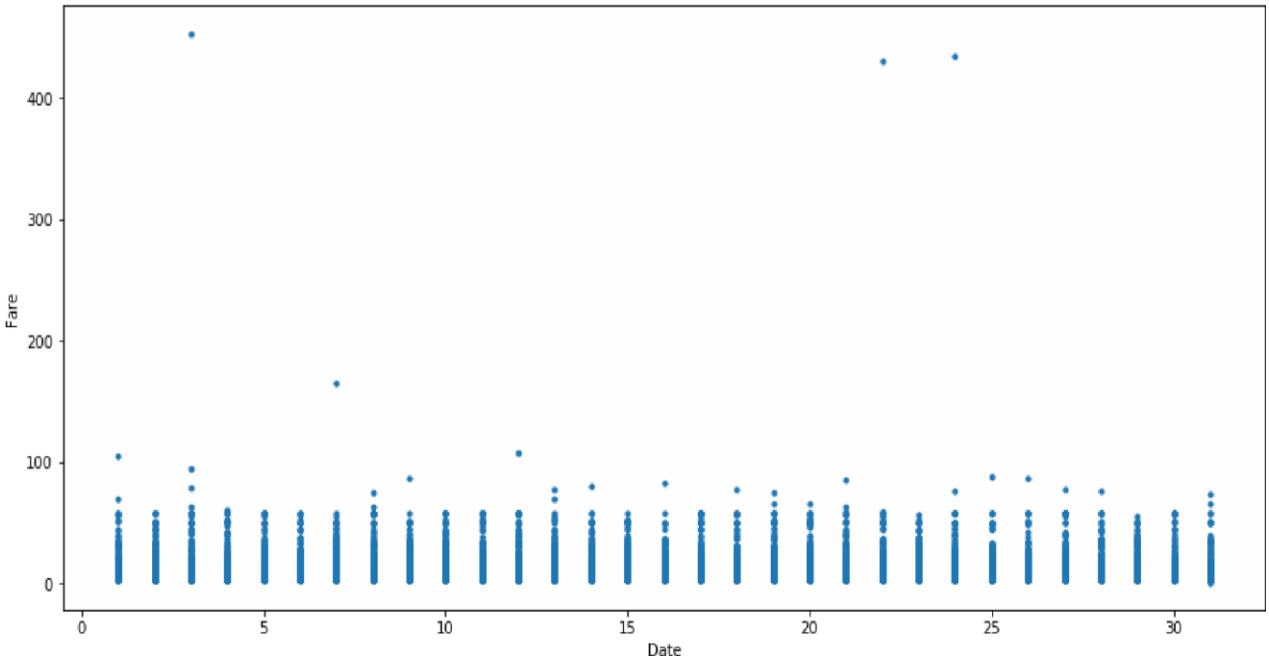
Basically this is how we can choose the model for the evaluation of the data that we are using for prediction of the Cab fare amount in the New York City.

By using the Random Forest we can predict the fare amount by using the test data set.

5.3 Important Figures

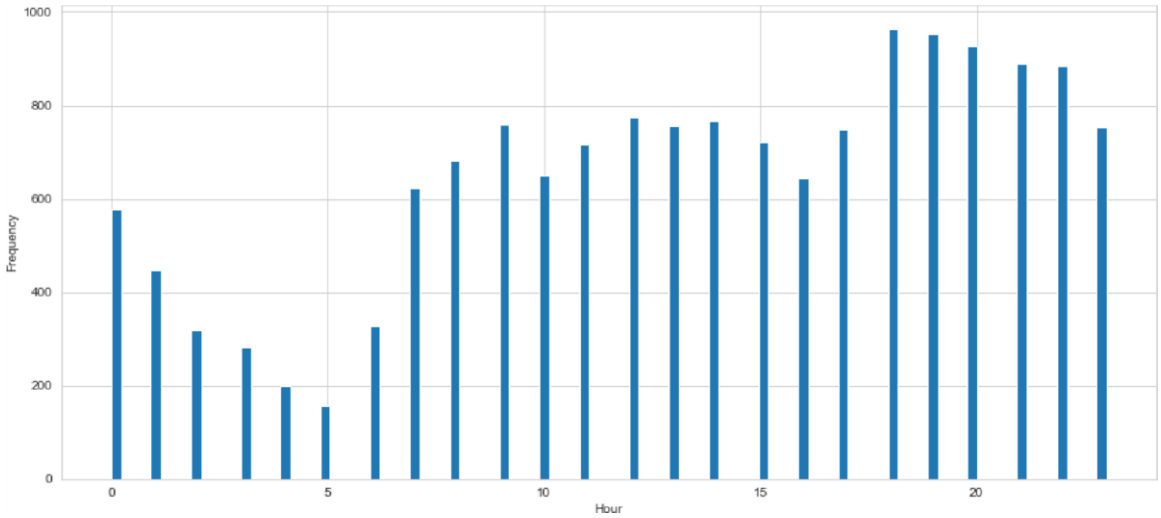
Date of month and fares

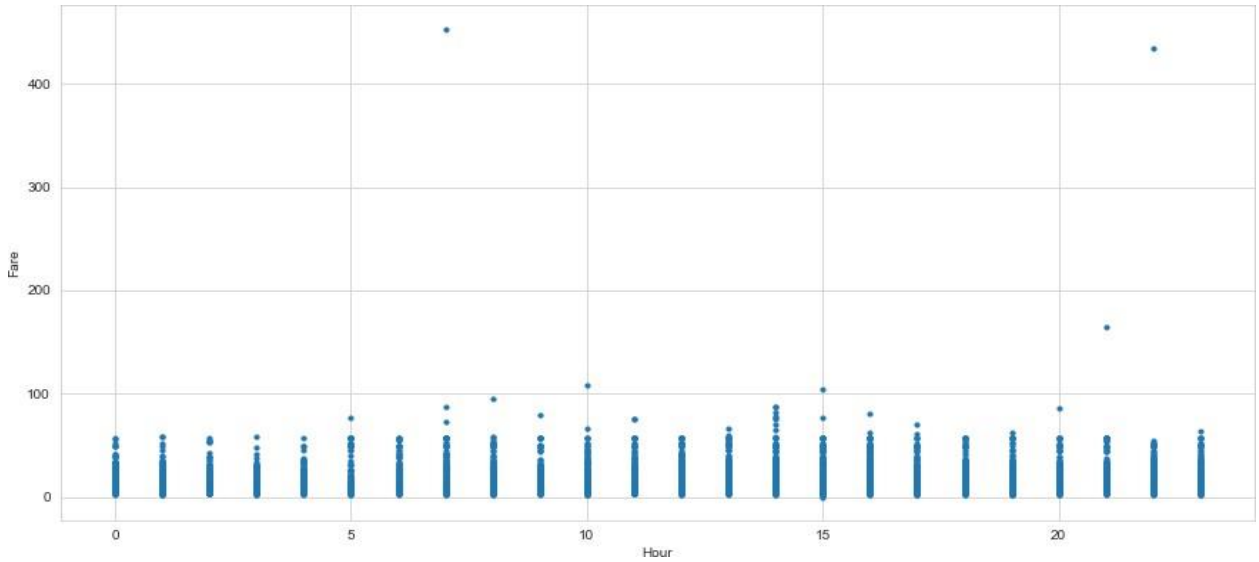
The fares throughout the month mostly seem uniform.



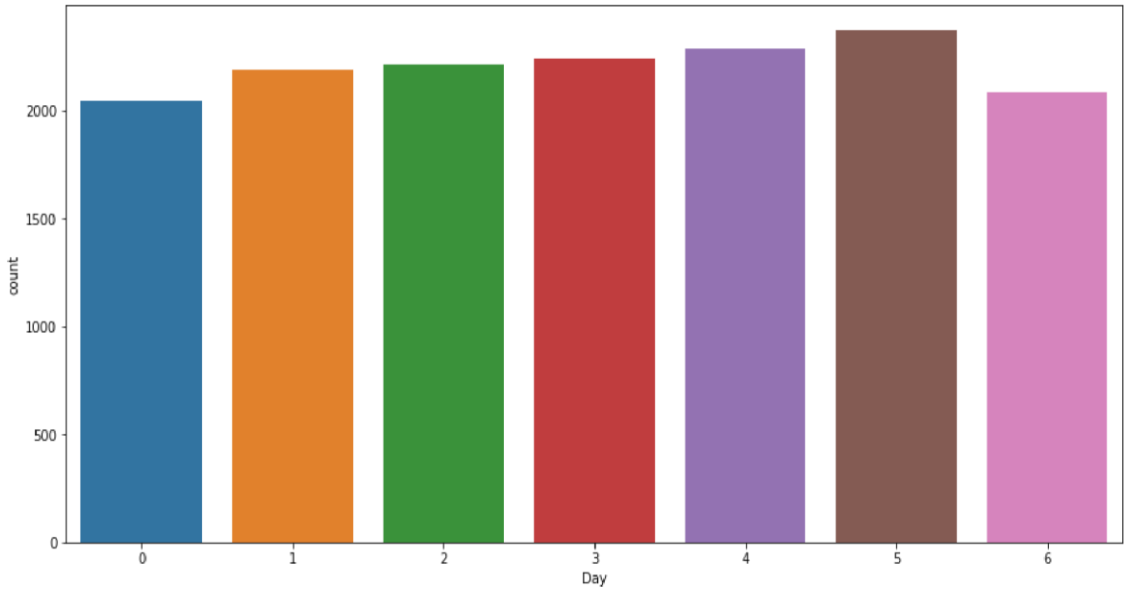
Hours and Fares

During hours 6 PM to 11PM the frequency of cab boarding is very due to peak hours. Fare prices during 2PM to 8PM is bit high compared to all other time might be due to high demands.





Impact of Day on the Number of Cab rides:



Observation: The day of the week does not seem to have much influence on the number of cabs ride.

Tools Used

- Jupyter(Python Development)
- R Studio(R code)
- Tableau(Data visualization)
- MS Word(Report Making)
- Excel(Plotting tables and reading Data)

References

1. For Data Cleaning and Model Development - https://edwisor.com/career-data-scientist
2. For other code related queries - https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
3. https://stackoverflow.com/
4. To know best time to visit New York City. https://santorinidave.com/best-time-to-visit-nyc

Python Code Attached- CAB_Fair_train.ipynb (should be run jupyter only)

R Studio Code Attached (code must be run on RStudio Only)

END OF REPORT