Project Report On

CAB FARE PREDICTION

Submitted by

NIKET RAMESH PATIL

Contents

CHAPTER 1

INRODUCTION

1.1 Problem Statement

You are a cab rental start-up company. You have successfully run the pilot project & 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.

1.3 Data Understanding

Attribute Information:

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

1.4 Exploratory Data Analysis

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 [5]: cab_train.head(10)
```

Out[5]:

	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
5	12.1	2011-01-06 09:50:45 UTC	-74.000964	40.731630	-73.972892	40.758233	1.0
6	7.5	2012-11-20 20:35:00 UTC	-73.980002	40.751662	-73.973802	40.764842	1.0
7	16.5	2012-01-04 17:22:00 UTC	-73.951300	40.774138	-73.990095	40.751048	1.0
8	NaN	2012-12-03 13:10:00 UTC	-74.006462	40.726713	-73.993078	40.731628	1.0
9	8.9	2009-09-02 01:11:00 UTC	-73.980658	40.733873	-73.991540	40.758138	2.0

Fig.1.1 Training data of Taxi Fare

```
In [6]: ############################## Find The Datatypes ##############################
        cab_train.dtypes

Out[6]: fare_amount          object
        pickup_datetime      object
        pickup_longitude    float64
        pickup_latitude     float64
        dropoff_longitude   float64
        dropoff_latitude    float64
        passenger_count     float64
        dtype: object
```

Fig.1.2 Data-types of Given Variables

In the above Fig.1.2 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.

1.3.1 Assumption:

As we are talking about how independent variables will be effect on the target variable. So there will be the multiple assumptions.

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.

1.3.2 Data Understanding and Cleaning

- ***Pickup_datetime***

As we see in the train data set we have 6 independent and 1 target variables let's discusses one by one. **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, minute, second as shown below.

```
In [8]: cab_train.head()
```

Out[8]:

p_latitude	dropoff_longitude	dropoff_latitude	passenger_count	pickup_year	pickup_month	pickup_day_of_month	pickup_hour	pickup_minute	pickup_second
40.721319	-73.841610	40.712278	1.0	2009.0	6.0	15.0	17.0	26.0	21.0
40.711303	-73.979268	40.782004	1.0	2010.0	1.0	5.0	16.0	52.0	16.0
40.761270	-73.991242	40.750562	2.0	2011.0	8.0	18.0	0.0	35.0	0.0
40.733143	-73.991567	40.758092	1.0	2012.0	4.0	21.0	4.0	30.0	42.0
40.768008	-73.956655	40.783762	1.0	2010.0	3.0	9.0	7.0	51.0	0.0

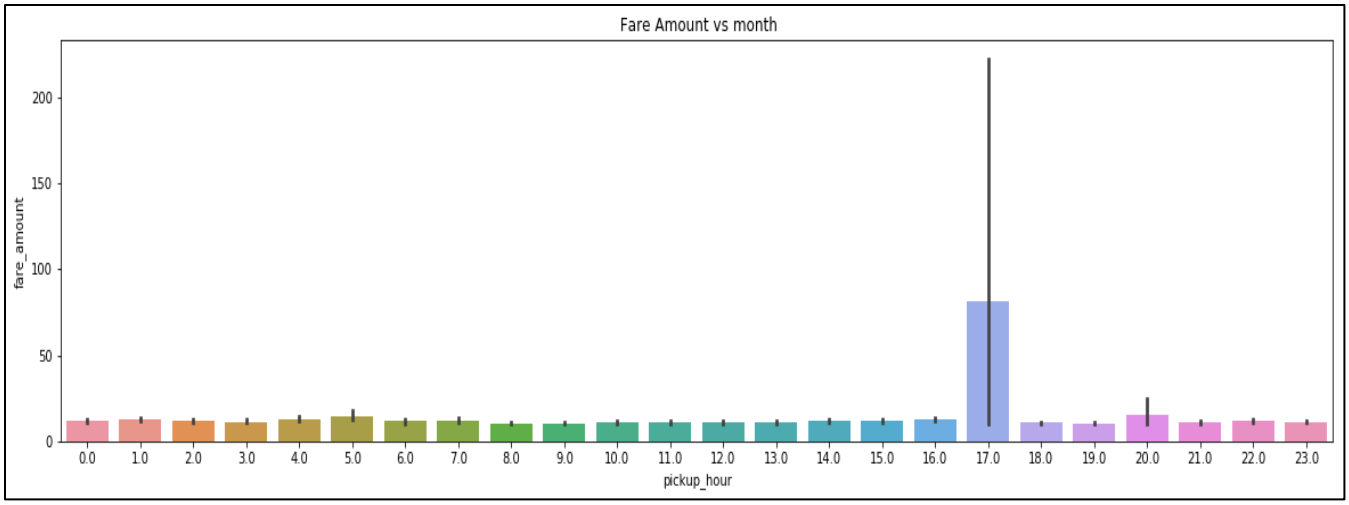

Fig.1.4 Pickup_hour vs fare_amount

As we can see hour 17 is impacting more on the target variable. In this particular time people use more cab service.

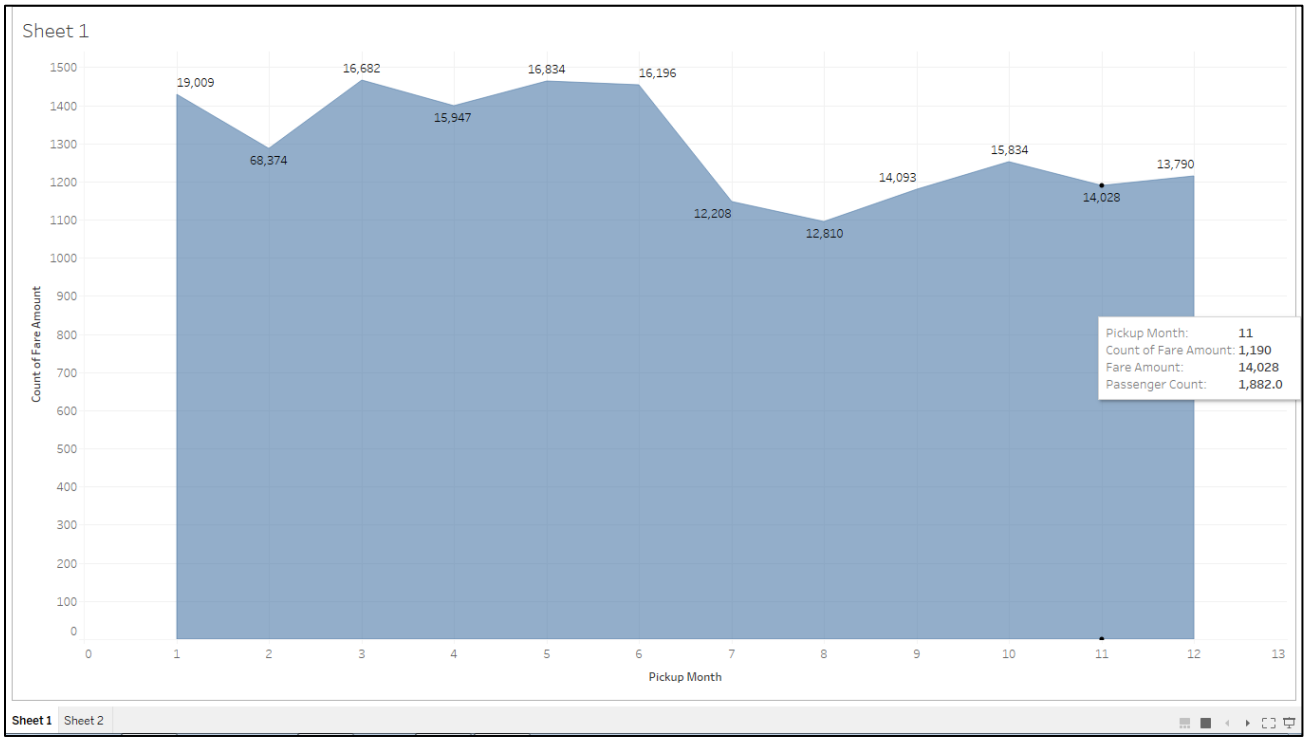

Fig.1.5 Pickup_year vs. Fare_amount on Tableau

As we can notice in the above plot 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. The pilot project data which we have received it is from 2009 to 2015 and as we can see in the below Fig.1.6 year 2015 has more fare_amount as compare to other year.

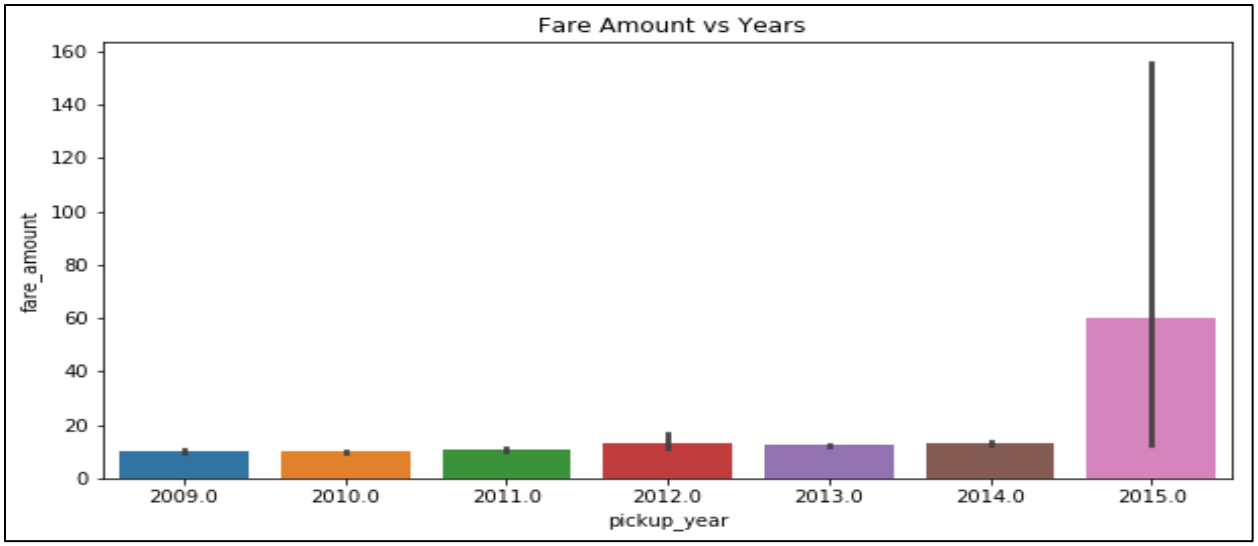

Fig.1.6 Pickup_month vs. Fare_amount

- ***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 using haversine formula. 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.

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi_1 \cdot \cos\varphi_2 \cdot \sin^2(\Delta\lambda/2)$$

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

$$d = R \cdot c$$

: Where,

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.1.7 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.

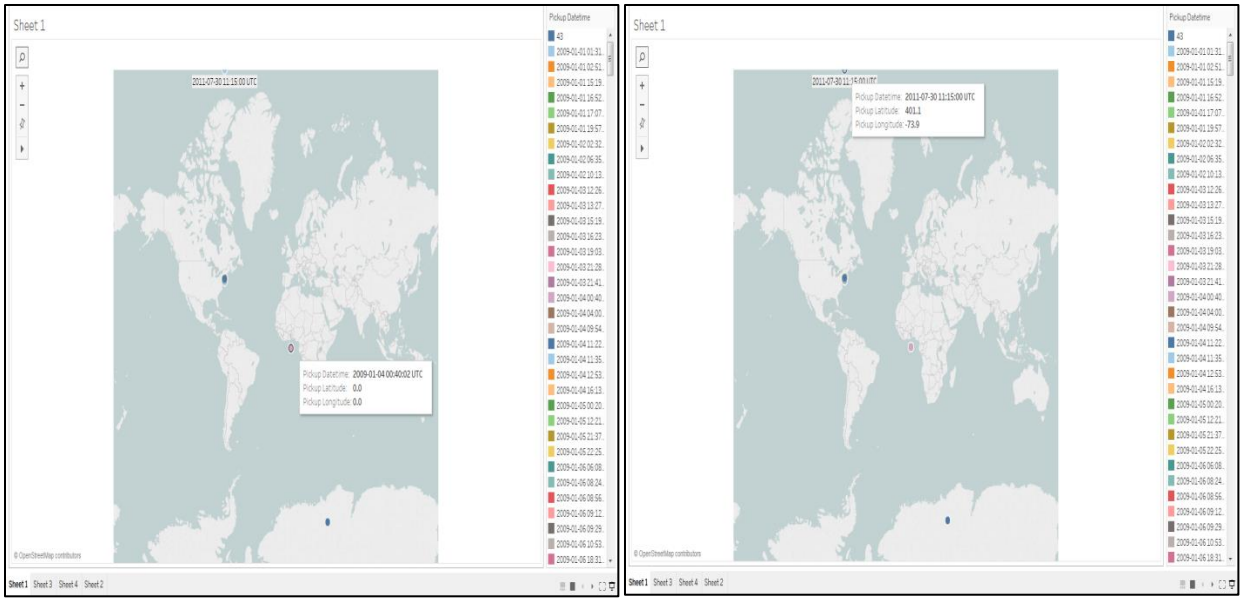

Fig.1.7 Latlong Visualization on Tableau

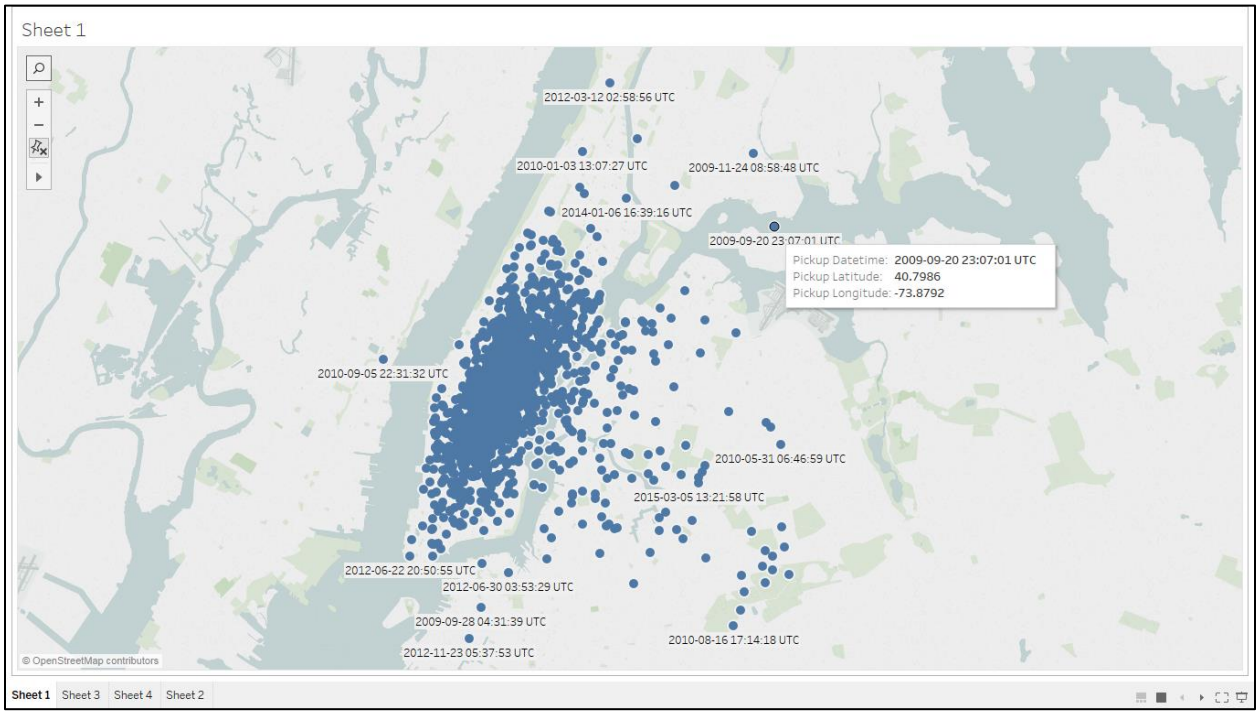

Fig.1.8 Visualization of Pickup Latlong vs. Pickup Time on Tableau

In above Fig.1.8 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. In below **Fig.1.10** we can see fare distance vs. drop-off location. When we compare train data vs. test data on the behalf of trip distance Test distance

lies within the range of 0 to 99km so here in the train data we consider the same as shown in Fig.1.9

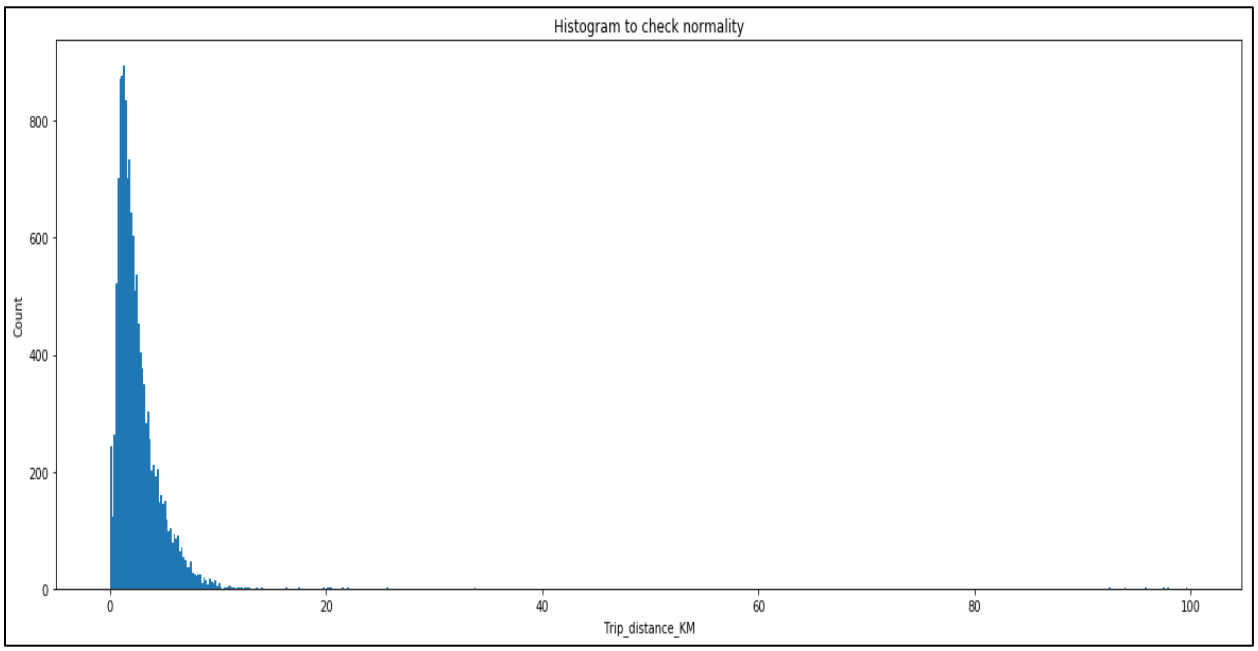

Fig.1.9 Visualization of Drop-off Lotlong vs. Trip Distance on Tableau

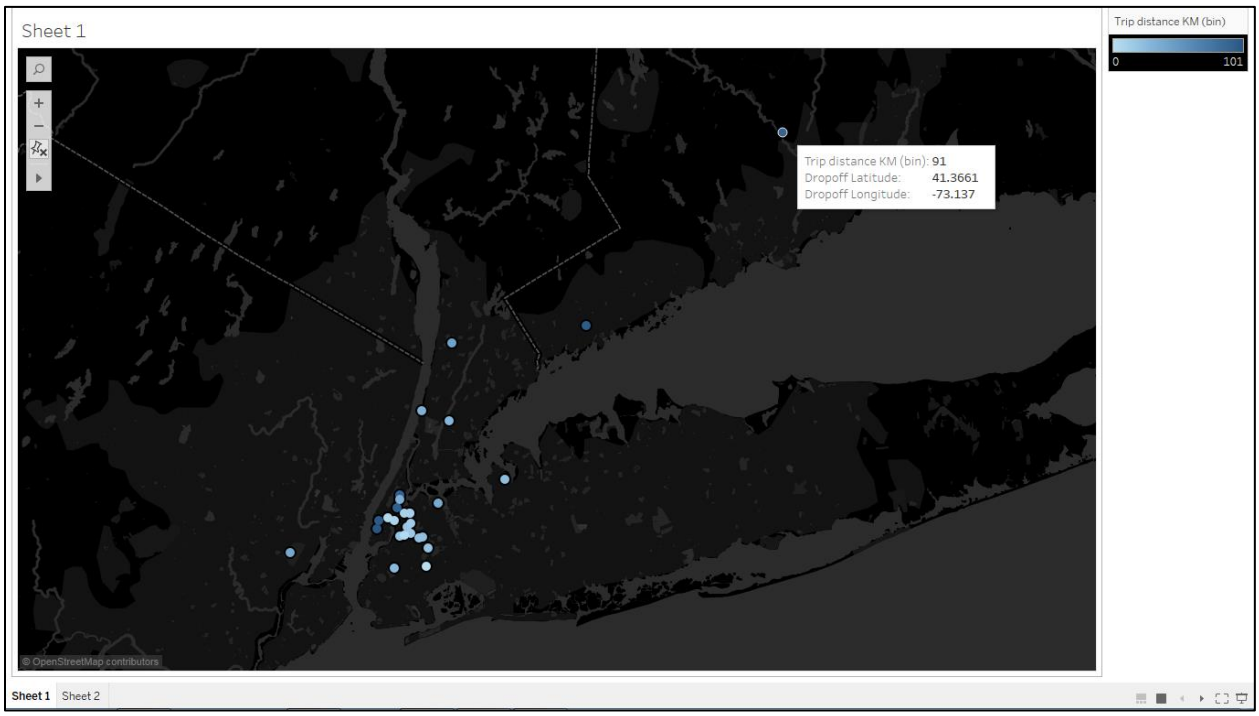

Fig.1.10 Visualization of Drop-off Lotlong vs. Trip Distance on Tableau

- ***Fare Amount***

When we plot the values of fare amount we came to know few values are negative.The min value is -3$ and max 54343$ in dollar as shown in **Fig.1.8** 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.

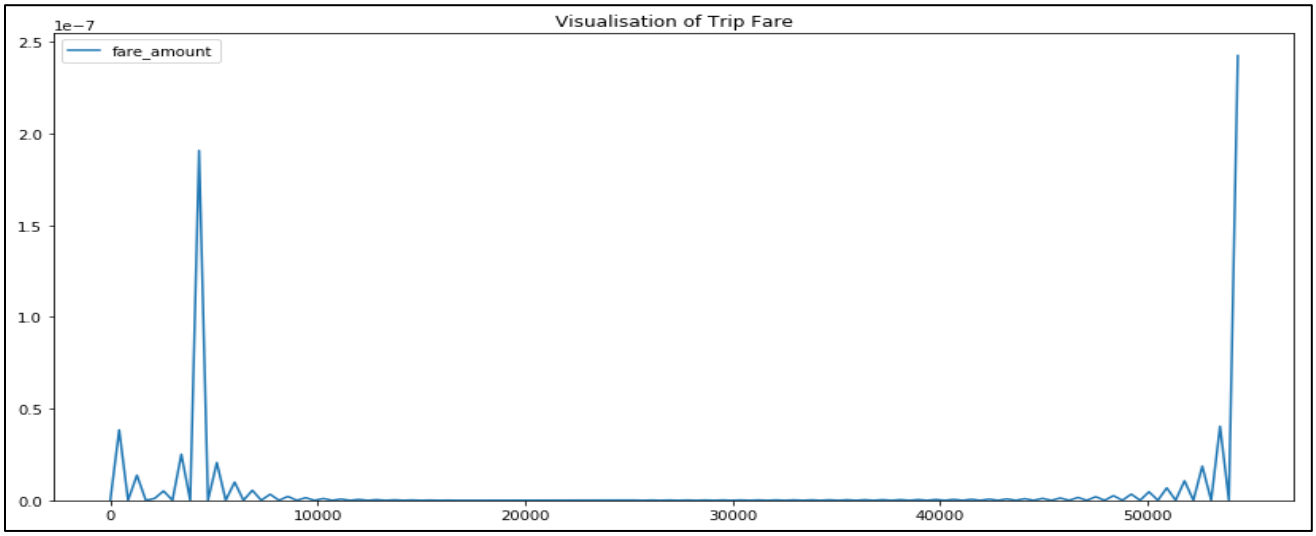

Fig.1.11 Visualization of Fare Amount

- ***passenger_count***

In train data set 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

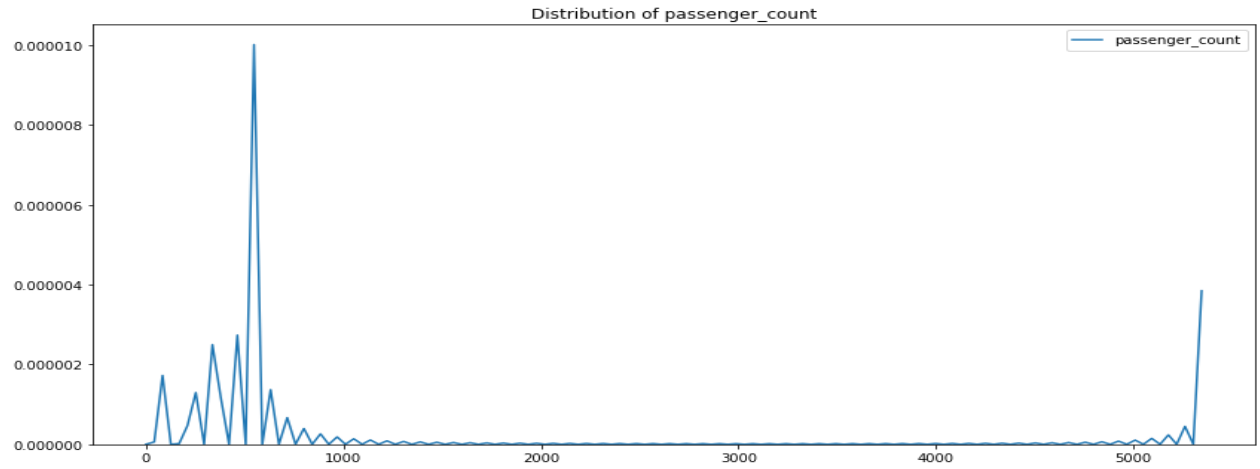

Fig.1.12 Visualization of Passenger Count

CHAPTER 2

DATA PREPROCESSING

This step involves cleaning the data, dropping unwanted attributes, conversion of data-types 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 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 **Fig. 2.1** 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 i.e. median as shown below.

```
In [14]: missing_val
```

Out[14]:

	Variables	Missing_percentage
0	pickup_year	0.006418
1	pickup_month	0.006418
2	pickup_day_of_month	0.006418
3	pickup_hour	0.006418
4	pickup_minute	0.006418
5	pickup_second	0.006418
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.2.1 Missing Value Table

2.2 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 2.2**

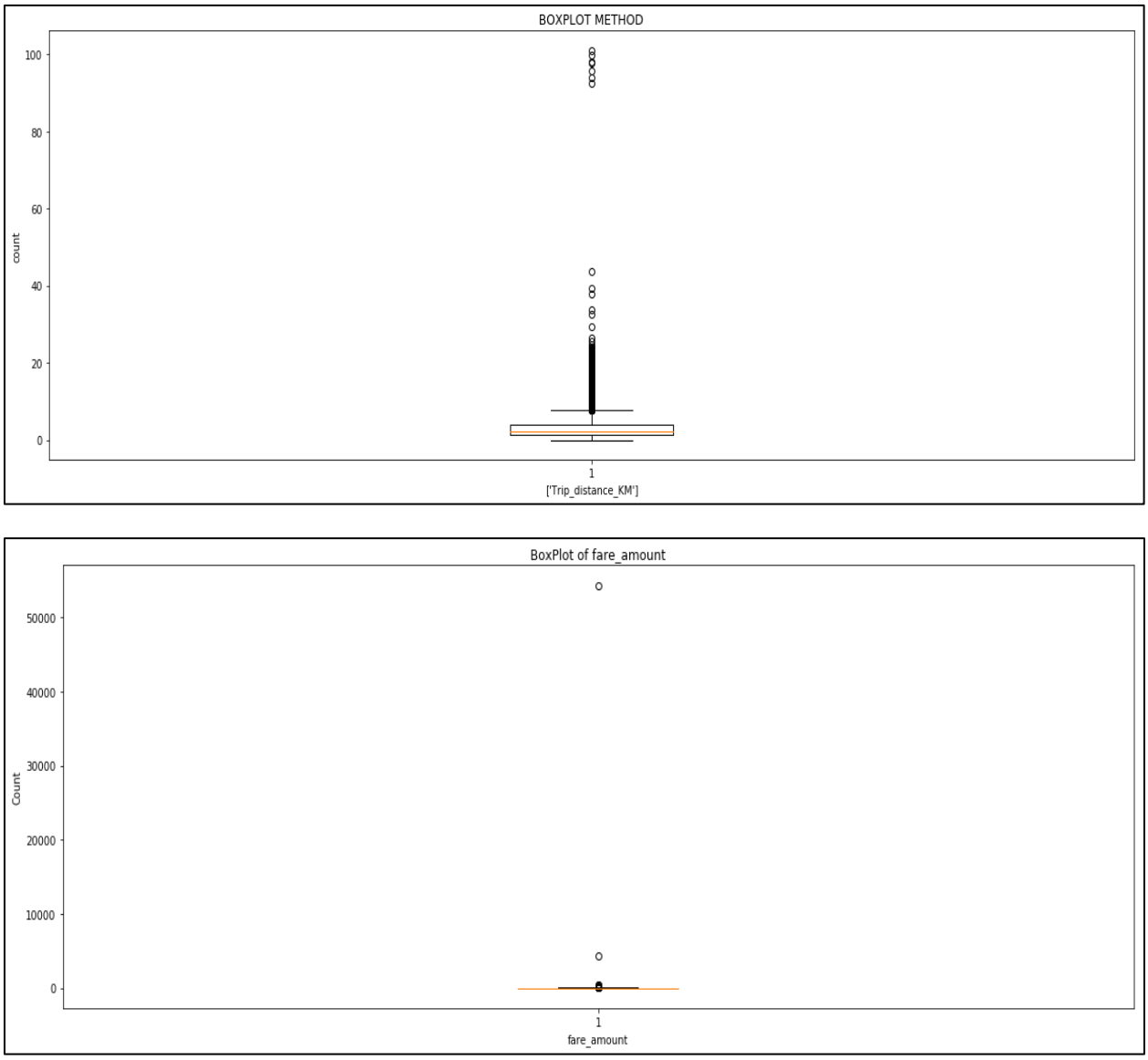

Fig.2.2 Finding Outliers Using Boxplot

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.

2.3 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-collinarity problem we need to applied 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.2.3 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 batter.**

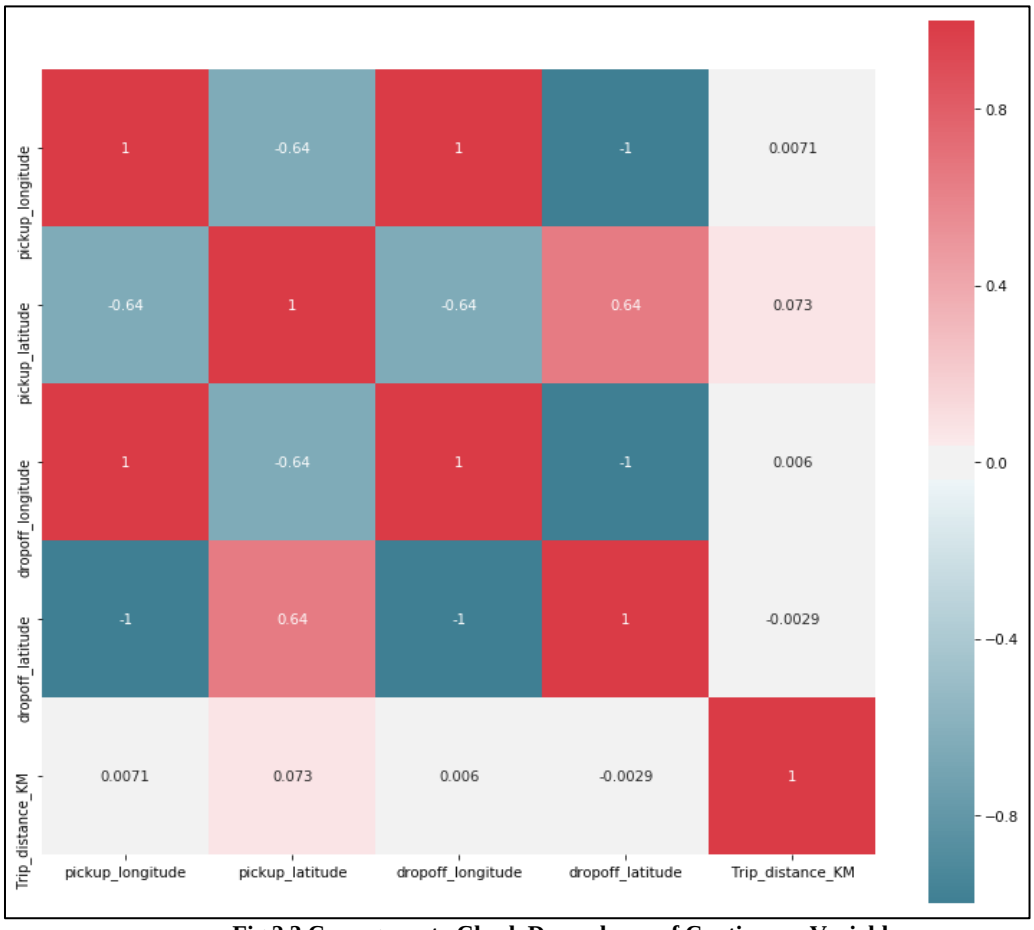

Fig.2.3 Corrogram to Check Dependency of Continuous Variable

When we applied **ANOVA** test on categorical variables the p value of all the variables is **0** it means there no any dependency, as shown in **Fig.2.4.**

```
P value for variable pickup_year is 0.0
f value for variable pickup_year is 2770443965.632115
P value for variable pickup_month is 0.0
f value for variable pickup_month is 3520.1177754289
P value for variable pickup_day_of_month is 0.0
f value for variable pickup_day_of_month is 7013.444824065899
P value for variable pickup_hour is 0.0
f value for variable pickup_hour is 5194.470021689829
P value for variable pickup_minute is 0.0
f value for variable pickup_minute is 19509.690429726423
P value for variable pickup_second is 0.0
f value for variable pickup_second is 1885.7189632049021
P value for variable passenger_count is 0.0
f value for variable passenger_count is 40121.009130591825
```

Fig.2.4 Anova Test to Check Dependency of Categorical Variable

2.4 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.2.5** our data has skewed in nature. So, here we applied **normalization on continuous variables.**

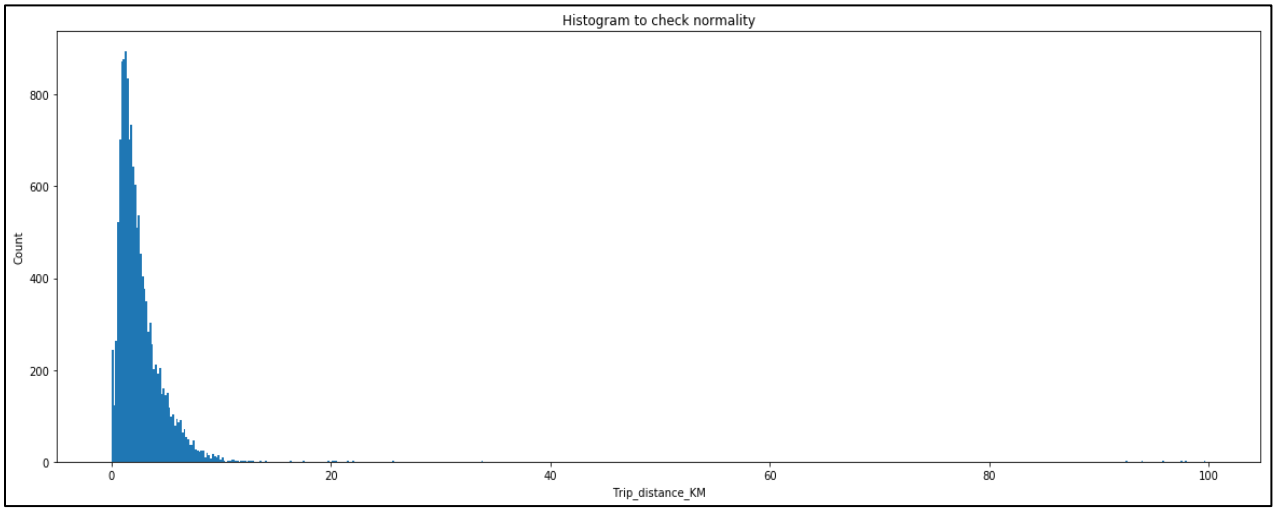

Fig.2.4 Histogram to check normality of continuous variable

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.

CHAPTER 3

Modeling

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.**

3.1 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. The RMSE and R^2mvaalue of our project are shown below.

	Python	R
RMSE Train	0.8340099802588417	0.735898067
RMSE Test	2.1992287277012164	0.1455232820
R^2 Test	0.7180091102577547	0.8597744

Table 3.1 Bias and Variance of Random Forest

The error rate of Random Forest algorithm for training data set is very low as compare to test data so there will be the problem of model **over-fitting** so, to reduce it we need to tune our model. Therefor we are moving towards another model.

3.2 Liner Regression

Linear Regression is one of the statistical methods of prediction. It is used to find a linear relationship between the target and one or more predictors. It means the target variables should be continuous in nature. 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. Below we calculated RMSE and R^2 values using liner regression.

	Python	R
RMSE Train	3.300676007712309	1.7100219162
RMSE Test	3.7581451715950474	1.711712e+02
R^2 Test	0.17654213389232376	0.1560715

Table 3.2 Bias and Variance of Liner Regression.

The error rate of linear regression i.e. RMSE is quit acceptable but the r^2 value i.e. the variance is very low so let us check another model.

3.3 XGboost

Gradient boosting is currently one of the most popular machine learning techniques for efficient modeling of tabular datasets of all sizes. It is very fast, takes quite less RAM to run, and focuses on the accuracy of the result.
It is a technique which applicable for regression and classification type of problems. In this method, multiple weak learners are ensemble to create strong learners. Gradient 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. Here we select XGBoost for model development.

	Python	R
RMSE Train	2.058141501647218	1.361333984
RMSE Test	2.107202923854742	1.419343858
R^2 Test	0.7411149340237774	0.3185938

Table 3.3 Bias and Variance of XGBoost.

The error rate of XGboost Machine learning algorithm is acceptable and the value of R^2 is also batter as compare to other model and if we are talking about speed, XGboost works faster as compare to other. So, here we select XGboost as our final model.

3.4 Applying XGboost on test data set.

Here we finalize Xgboost as our final model and applied the same to predict the cab fare of test d ata set. First we pass our test data set from all above data preprocessing techniques and finally apply XGboost to predict fare amount of test case. **Which is the final aim of our project.**

CHAPTER 4

Conclusion

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.

4.1 Model Evaluation

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.

4.2 Model Selection

As we observed on all the model performance the **XGboosting** gives us batter outcome s as compare to other model. The RMSE of **XGboost** is minimum and the value of R^2 are also acceptable. The values difference of RMSE for train and test data is vary less so there is no any p roblem of model over-fitting. So, here we select **XGboost** is our final model for our given proble m statement and we applied the same on test case data to predict fare amount.

Appendix

1. Python code is attached separately

2. R code

```
rm(list=ls())

# Lode directory

setwd("C:/Users/User/Desktop/Project 3")
getwd()

# loading Libraries

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "e1071","geosphere",
"DataCombine", "pROC", "doSNOW", "class", "readxl","ROSE","dplyr", "plyr",
"reshape","xlsx",
     "pbapply", "unbalanced", "dummies", "MASS" , "gbm" ,"Information", "rpart", "tidyr",
"miscTools")

# install.packages
#lapply(x, install.packages)

 #load libraries
lapply(x, require, character.only = TRUE)
rm(x)

#Lodeing the file
cab_train = data.frame(read.csv('train_cab.csv'))

#structure of data or data types
str(cab_train)

#Summary of data
summary(cab_train)

# Changing datatypes
cab_train$fare_amount=as.numeric(cab_train$fare_amount)

# split datetime into seprate attribute
library(lubridate)
library(dplyr)
```

```
cab_train$Date = as.Date(cab_train$pickup_datetime)
cab_train$month = month(cab_train$Date)
cab_train$year = year(cab_train$Date)
cab_train$day = day(cab_train$Date)
cab_train$hour = hour(cab_train$pickup_datetime)


#Now we can drop the column pickup_datetime and Date
cab_train = subset(cab_train, select = -c(pickup_datetime,Date))

str(cab_train)

# calulating distance between latitude/longitude using the Haversine formula

lat1 = cab_train['pickup_latitude']
lat2 = cab_train['dropoff_latitude']
long1 = cab_train['pickup_longitude']
long2 = cab_train['dropoff_longitude']

##### Function to calculate distance ######

gcd_hf = function(long1, lat1, long2, lat2) {
  R = 6371.145 # radius of earth in KM
  delta.long = (long2 - long1)
  delta.lat = (lat2 - lat1)
  a = sin(delta.lat/2)^2 + cos(lat1) * cos(lat2) * sin(delta.long/2)^2
  c = 2 * atan2(sqrt(a),sqrt(1-a))
  d = R * c
  return(d) # Distance in km
}

#################### Let's apply to all variables ######################

for (i in 1:nrow(cab_train))
{
  cab_train$Trip_distance_KM[i]= gcd_hf(cab_train$pickup_longitude[i],
cab_train$pickup_latitude[i], cab_train$dropoff_longitude[i],
                          cab_train$dropoff_latitude[i])
}

############ Lets drop the columns of latitude/longitude.#####################

cab_train = subset(cab_train, select = -
c(pickup_latitude,dropoff_latitude,pickup_longitude,dropoff_longitude))
```

```
#----------------------------------Missing Values Analysis-----------------------------------------#

missing_val = data.frame(apply(cab_train,2,function(x){sum(is.na(x))}))
missing_val$Columns = row.names(missing_val)
names(missing_val)[1] =  "Missing_percentage"

#Calculating percentage missing value
missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(cab_train)) * 100

# Sorting missing_val in Descending order
missing_val = missing_val[order(-missing_val$Missing_percentage),]
row.names(missing_val) = NULL

# Reordering columns
missing_val = missing_val[,c(2,1)]

missing_val

# Missing value plot
ggplot(data = missing_val[1:100,], aes(x=reorder(Columns, -Missing_percentage),y =
Missing_percentage))+
  geom_bar(stat = "identity",fill = "red")+xlab("Variables")+
  ggtitle("Missing data percentage") + theme_bw()

# Median Method
cab_train$passenger_count[is.na(cab_train$passenger_count)] =
median(cab_train$passenger_count, na.rm = T)

sum(is.na(cab_train))

# as we know fare amount cannot be -ve
cab_train$fare_amount[cab_train$fare_amount<=0] = NA
cab_train$fare_amount[cab_train$fare_amount>500] = NA

sum(is.na(cab_train))

 #removing passengers count more than 6
cab_train$passenger_count[cab_train$passenger_count<1] = NA
cab_train$passenger_count[cab_train$passenger_count>6] = NA

sum(is.na(cab_train))

cab_train$passenger_count[is.na(cab_train$passenger_count)] =
median(cab_train$passenger_count, na.rm = T)

sum(is.na(cab_train))
```

```
summary(cab_train)

# removing outliers in distance
str(cab_train)

cab_train$Trip_distance_KM[cab_train$Trip_distance_KM <= 0] = NA
cab_train$Trip_distance_KM[cab_train$Trip_distance_KM > 200] = NA

sum(is.na(cab_train))

cab_train$Trip_distance_KM[is.na(cab_train$Trip_distance_KM)] =
median(cab_train$Trip_distance_KM, na.rm = T)

sum(is.na(cab_train))

summary(cab_train)

# Here we manually apply all preprocessing techniques i.e. (missing value, outliers, feature selection) on the top of our train data. So what we can do is we directly normalize our data

####### Feature Scaling ##########

cname = c('Trip_distance_KM')

catname = c( 'month', 'year' , 'hour' , 'passenger_count', 'day')

# data visualisation to check normality

hist(cab_train$Trip_distance_KM,col="Red",main="Histogram of Trip distance")

# Here our data is not normally distributed so here we go for Normalization

for(i in cname)
{
  print(i)
  cab_train[,i] = (cab_train[,i] - min(cab_train[,i]))/(max(cab_train[,i])-min(cab_train[,i]))
}

#Creating dummy variables for categorical variables
library(mlr)
df = dummy.data.frame(cab_train, catname)

#pf = df
```

#Clean the Environment-

```
rmExcept("df")
```

 # Divide data into train and test using stratified sampling method

```
library(caret)
set.seed(123)
train_index = createDataPartition(df$fare_amount, p = 0.8, list = FALSE)
train = df[train_index,]
test  = df[-train_index,]
```

################## Applying Supervises Machine Learning ####################

##_____ Random Forest _____##

```
set.seed(140)
library(randomForest)  # Lode library for random forest.
library(inTrees)       # Lode library for intree transformation
```

#Develop Model on training data

```
fit_RF = randomForest(fare_amount~., data = train)
```

#Lets predict for training data

```
RF_train = predict(fit_RF, train)

RF_test = predict(fit_RF,test)
```

######################### Error metrics to calculation ############################

```
print(postResample(pred = RF_train, obs = train$fare_amount)) # For Train

print(postResample(pred = RF_test, obs = test$fare_amount)) # For Test
```

############## Visualization to check the model performance ##################

```
########### TRAIN ##############
plot(train$ fare_amount ,type="l",lty=1.8,col="Green")
lines(RF_train,type="l",col="Blue")

########### TEST ###############
plot(test$ fare_amount ,type="l",lty=1.8,col="Red")
lines(RF_test,type="l",col="Blue")
```

```
#_____ Linear Regression _____#

set.seed(123)

#Develop Model on training data
fit_LR = lm(fare_amount ~ ., data = train)

#Lets predict for training data
LR_train = predict(fit_LR, train)

#Lets predict for testing data
LR_test = predict(fit_LR,test)

----------# Error metrics to calculation #----------

print(postResample(pred = LR_train, obs = train$fare_amount)) # For Train

print(postResample(pred = LR_test, obs = test$fare_amount)) # For Test

#---------- XGBoost ----------#

set.seed(123)

# develop Model on training data
fit_XGB = gbm(fare_amount~., data = train, n.trees = 500, interaction.depth = 2)

#Lets predict for training data
XGB_train = predict(fit_XGB, train, n.trees = 500)

#Lets predict for testing data
XGB_test = predict(fit_XGB,test, n.trees = 500)

----------# Error metrics to calculation #----------

print(postResample(pred = XGB_train, obs = train$fare_amount)) # For train
print(postResample(pred = XGB_test, obs = test$fare_amount)) # For Test

#---------- Visualization Part to check performance of model ----------#

########## TRAIN ##############
plot(train$fare_amount,type="l",lty=1.8,col="Red")
lines(XGB_train,type="l",col="Blue")

########## TEST ###############
plot(test$fare_amount,type="l",lty=1.8,col="Green")
lines(XGB_test,type="l",col="Blue")
```

 # WE final XGboost model as our final model and apply on test data set (1st we need to preprocess and normalize all the test data using above pre-processing steps)

```
train = pf
test = data.frame(read.csv('test_normalize.csv')) # preprocess and normalize data

#XGboost
set.seed(123)

#develop Model on training data
fit_XGB = gbm(fare_amount~., data = train, n.trees = 500, interaction.depth = 2)


#lets predict for testing data
XGB_test1 = predict(fit_XGB,test, n.trees = 500)

# predicting the fare amount for test case
test_final = data.frame(XGB_test1)
```

References

1. Data Cleaning, Model Development and Data Visualization we used.
   https://edwisor.com/career-data-scientist

2. For Visualization using seaborn.
   https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/

3. To know best time to visit New York City.
   https://santorinidave.com/best-time-to-visit-nyc

4. Haversine algorithm for distance calculation.
   https://www.movable-type.co.uk/scripts/latlong.html


FUJI TAXI GROUP
FUJI TAXI GROUP
Thank You