Cab Fare Prediction Project Report

Aditya Joshi

**1.INTRODUCTION**

**1.1 Problem Statement**

We are a cab rental start-up company. We have successfully run the pilot project and now want to launch our cab service across the country.

We have collected the historical data from our pilot project and now have a requirement to apply analytics for fare prediction.

Our objective of this Cab Fare Prediction Project is to **design a system that predicts the fare amount for a cab ride taken in the city.**

**1.2 Dataset**

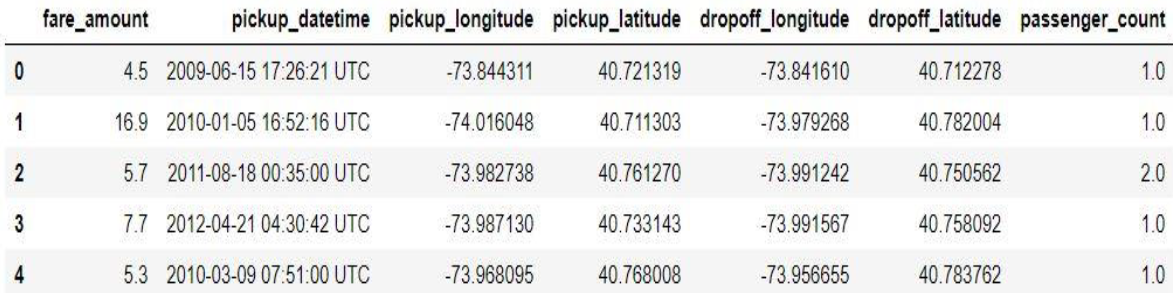
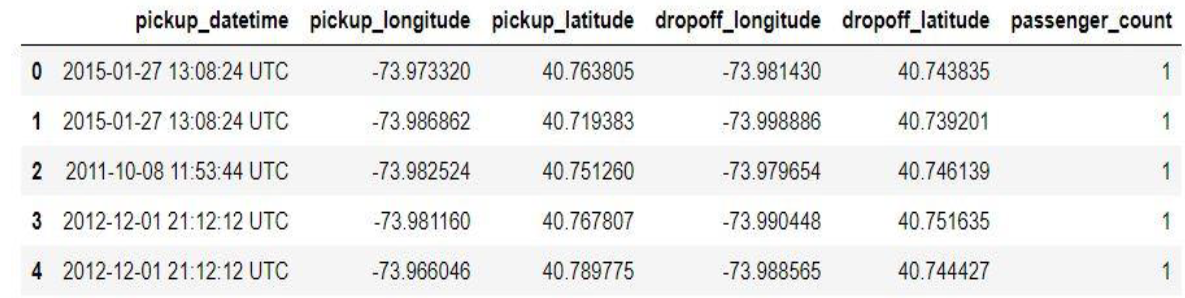


Image 1

In image 1, we can see our historical data. It has 16066 observations and 7 features, including fare\_amount which is our target variable

  
 Image 2

In image 2, we can see data, whose target variable i.e. fare\_amount values to be predicted. It has 9914 observations and 6 features, where, fare\_amount is not available.

our focus is to determine the values for our target variable fare\_amount for future test cases using the above shown dataset in image 1.

**1.3 Exploratory Data Analysis**

In Exploratory Data Analysis, we use different strategies like, like

• **Brainstorming** – Here, we will draw a rough sketch which talks about – the steps you are going to follow to achieve your given objective.

• **Defining Problem Statement** – In this project we are given with problem statement to work with, however, when dealing with real time scenarios, we are sometimes only given with limited information,

In that case, we are responsible to define the problem statement by ourselves. We have to understand the complete in and out operations of the client’s business and take decisions accordingly.

• Knowing the types of variables in the given dataset, whether they are factor, character or numeric. In our project pickup\_datetime variable is a time stamp variable, the fare\_amount is a float variable, the pickup\_latitude, pickup\_longitude, dropoff\_latitude and dropoff\_longitude are float variable or numerical variable.

• Changing and removing datatypes if required. In our project, we removed four variables i.e. pickup\_latitude, pickup\_longitude, dropoff\_latitude and dropoff\_longitude after we obtained distance from them.

• We also split the pickup\_datetime variable into year, month, date, day and hour,

**2. Methodology**

Now, we have the dataset and also, we discussed about Exploratory Data Analysis, let’s talk about the **methodology** we are going to follow to achieve  
our goal.

We will be going through:

• Pre-processing which includes missing value analysis, outlier analysis, feature selection and feature scaling.

• Model development, where we will choose what machine learning

algorithms to apply.

**2.1 Pre-processing**

In pre-processing, we actually apply few techniques like missing value analysis, outlier analysis, feature selection, feature scaling.

We never get a structured data to work with. Always messy data is handed to us, and we need to clean that data.

The data may have many observations (rows in dataset), where values in few fields will be absent. We can also say, there may be some inconsistent values in a variable (column in dataset), when compared with other values.

When we go for model development, we should have a structured dataset. We can’t go forward for model development, if we don’t apply pre-processing techniques on data and convert it into structured format.

**2.1.1 Missing Value Analysis**

Missing value analysis, as the name suggests, we face with situations, where we are given with dataset, and we have missing values in the observations.

The reason why we have missing values may be plenty, like human error, the user didn’t want to share his complete information, the one who was supposed to fill the data may didn’t work properly.

But, as discussed earlier, we have to give structured data to the model. In order to achieve that, we perform missing value analysis on top of the data to clean the data, to transform the data from unstructured format to structured format.

We give a line of code and it gives us the total number of missing values in each column. Later, we impute those missing values using mean, median or KNN method for numerical variables or using mode for categorical variables. In some cases, we may delete the observations with missing values, only when we have a case where we got few observations with missing values.

After imputing the missing values, we proceed further with outlier analysis. In our project, we had missing values in fare\_amount i.e. 25 missing values and passenger\_count i.e. 55 missing values. We dropped these observations from the dataset, as mentioned above, we may go for deletion of observations with missing values when we have a low number of it.

As discussed earlier, we have to give structured data to the model. In order to achieve that, we perform missing value analysis on top of the data to clean the data, to transform the data from unstructured format to structured format.

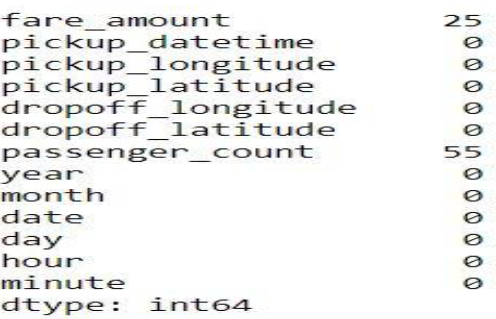


Image 3

We apply missing value techniques only on numerical variables.

**2.1.2 Outlier Analysis**

Outliers may be defined as the inconsistent values in a variable. For example, a = 1,2,3,4, 20. In object a, 20 is inconsistent, in terms of mean. Another important thing is, we apply outlier analysis only on numerical variables.

Outliers are used for fraud detection. Let’s say, in one bank account, consistently, we observe an amount which ranges from 50,000 to 1,00,000, but in one case 10 lakhs gets deposited. In that case, simply using outlier technique, we can get the inconsistent values.

In our project, we found outliers in fare\_amount, pickup\_datetime i.e. 43 (refer to image attached below), pickup\_latitude, and passenger\_count variables. In other variables the values are consistent with each other.

We deleted the outliers in these variables as the number of outliers were minimum. We went for ascending order and knew the inliers and went for descending order and learnt about outliers and finally sometimes we used to describe function to know minimum and maximum.

For example, in pickup\_latitude, we came to know about an outlier present in that simply using describe function.

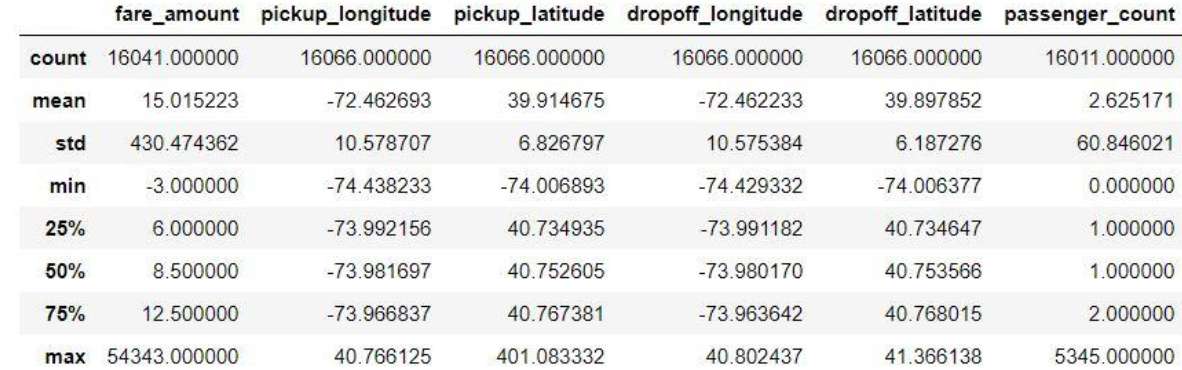


Image 4

You can check about that in third column, at the end of pickup\_latitude variable. We understood that 401.083332 is an outlier present after learning that latitudes value lies in between from -90 to +90.

**2.1.3 Data Understanding**

Understanding visually is easier sometimes and for that purpose we have few libraries in python which allows us to plot awesome visualizations.

We get data, and in order to talk about how few attributes relationship with each other, we can use these visualizations.

You can give two variables, to know how they are related with each other. You can also give three variables to understand the relationship between three variables. You got infinite possibilities to plot.

Let’s have a view on few of the plots we used in our project.

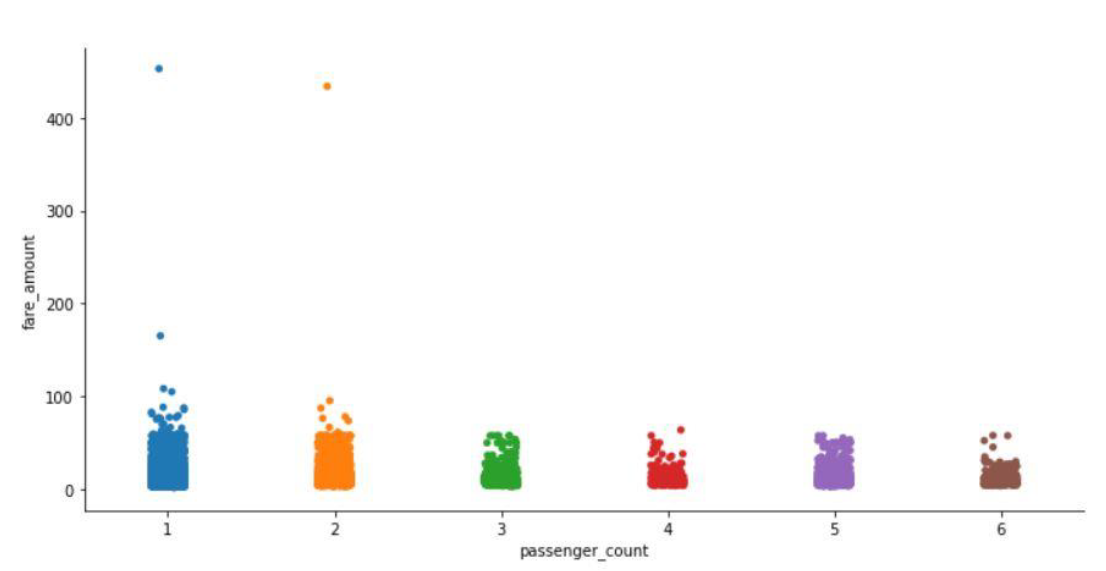


IMAGE 5

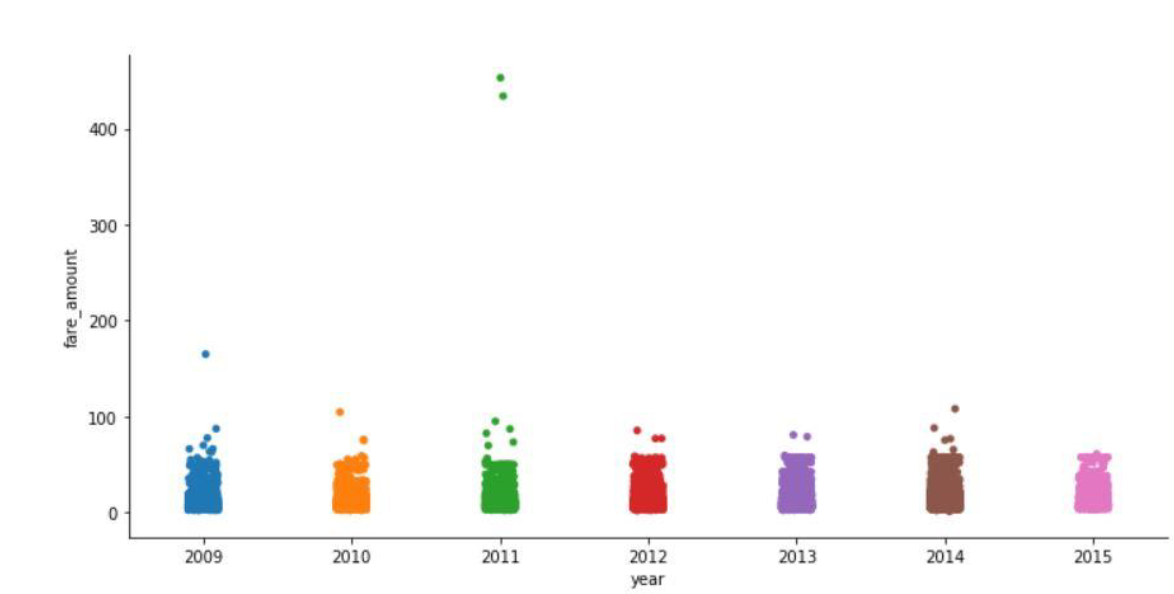


IMAGE 6

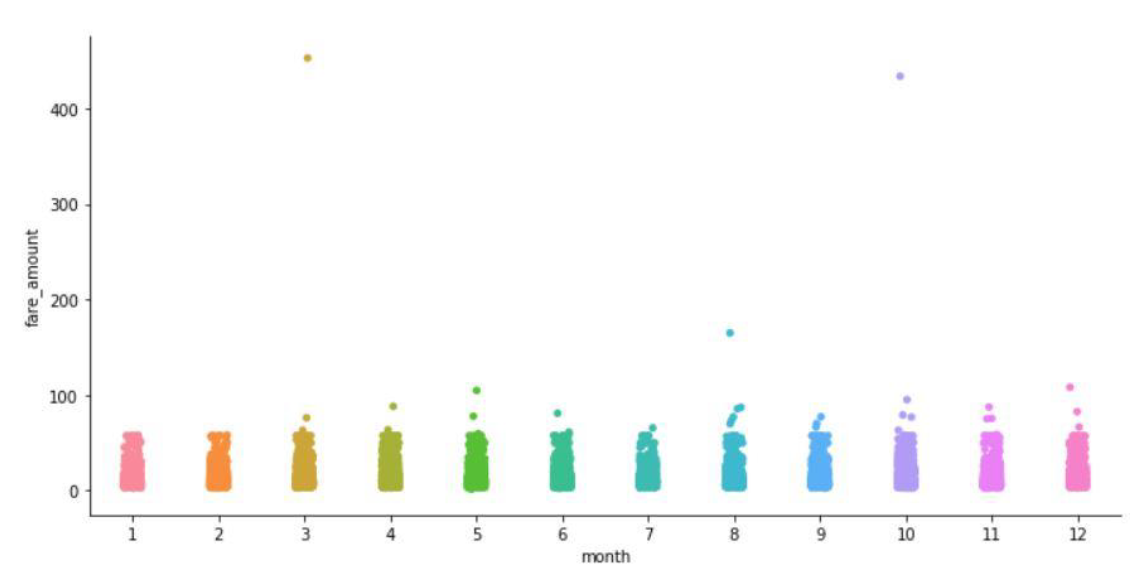


IMAGE 7

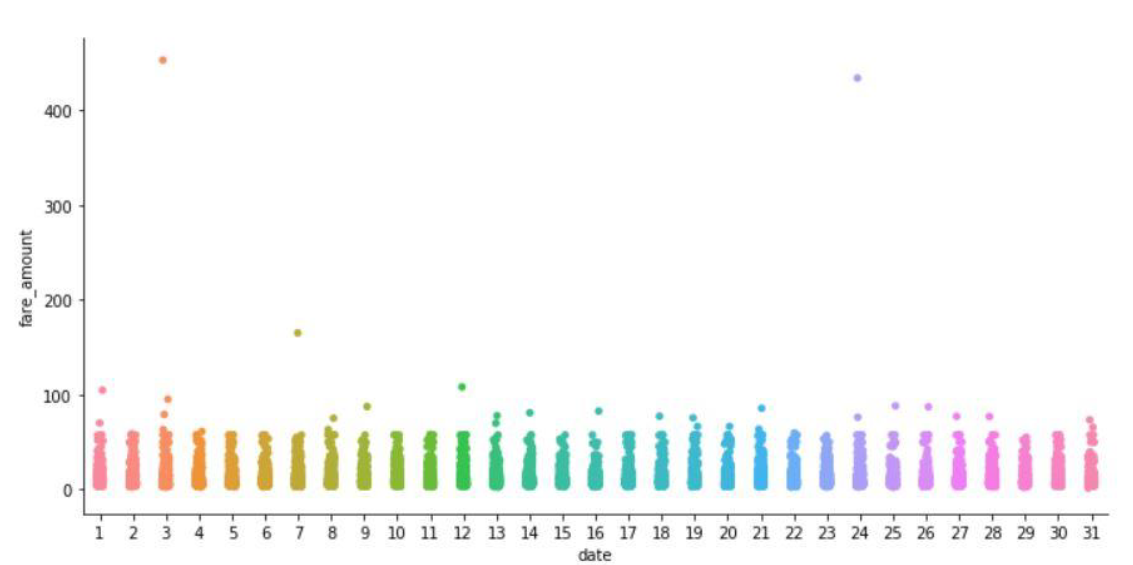


IMAGE 8

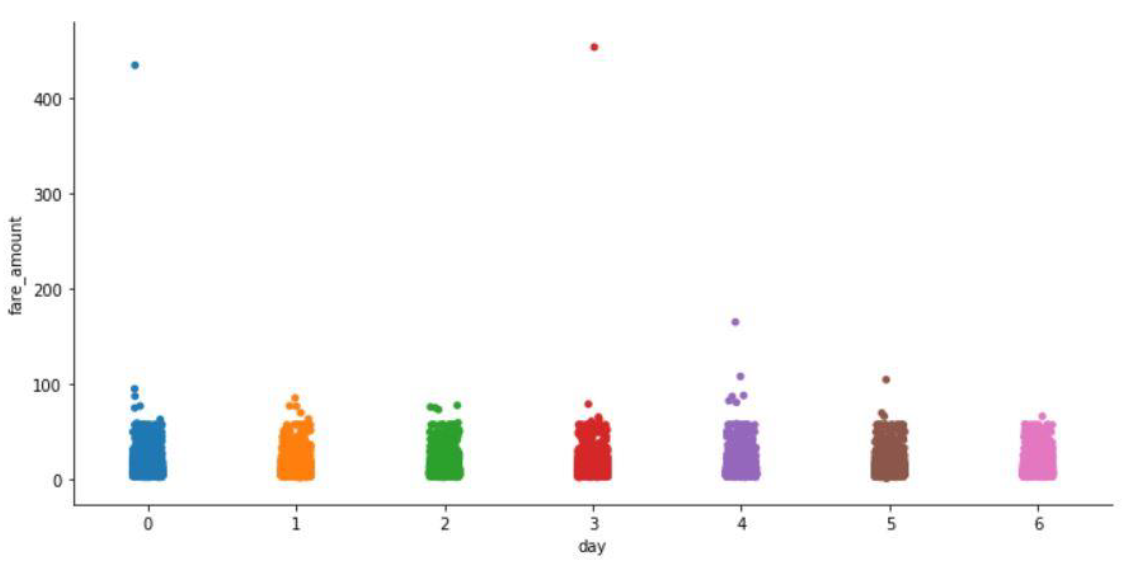


IMAGE 9

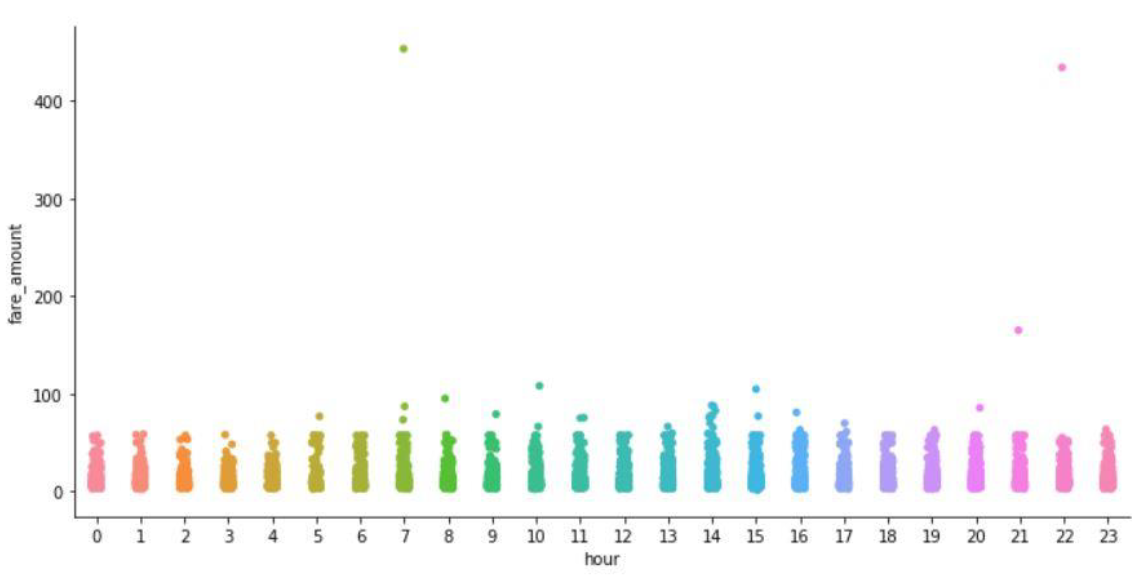


IMAGE 10

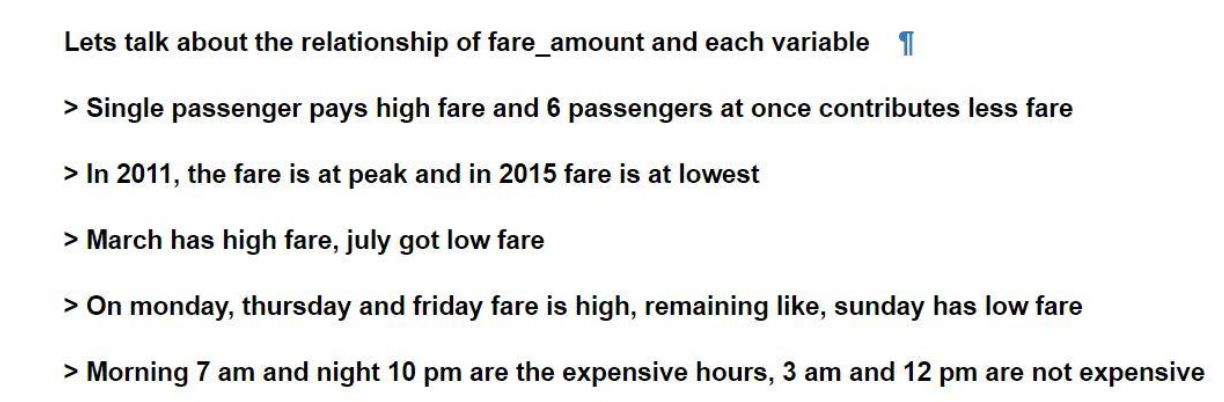


IMAGE 11

**2.1.4 Feature Selection**

In Feature Selection, we use few techniques to know which variables are important to us, it’s all about selecting subset of variables from all variables. Actually, when we are given with a dataset, we perform exploratory data analysis, later data pre-processing to clean the data and transform it from unstructured to structured data to feed into model.

However, after this, we may face a situation where we may have variables which have same information with them about the target variable. Let’s talk about an example, in a situation where we are sending five people on a mission.

Later, you came to know that, two individuals have the same exact information with them about the mission. Definitely you would drop one, in order to reduce infrastructure and complexity.

The same way, we also drop few variables if they have same information. We always aim that, there should be no independent variable which talks the same as other independent variables but, we appreciate those variables which talks more about the target variable.

In our project, we did correlation analysis between target variable (continuous variable) and other continuous variable i.e. distance. Let’s check it:

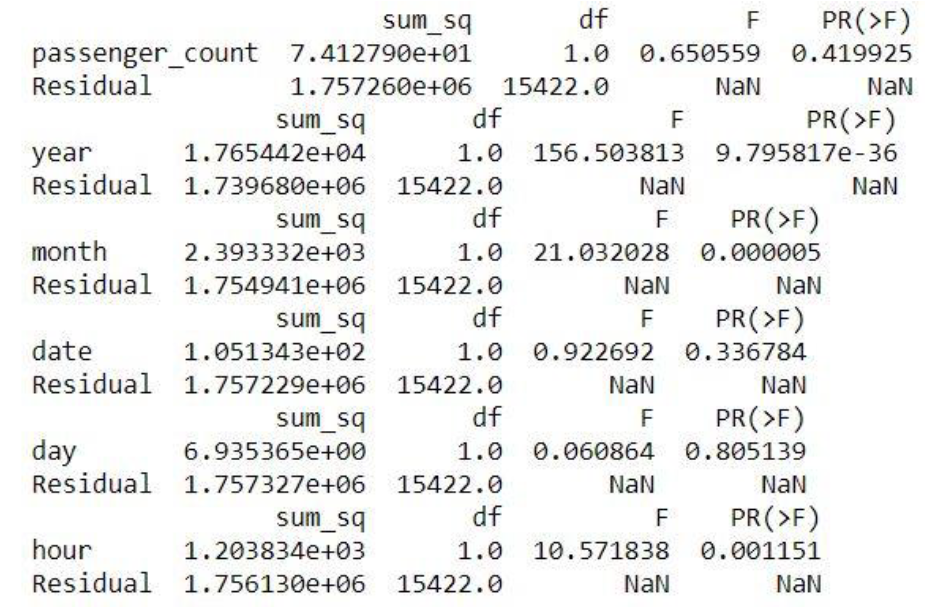


IMAGE 12

Now, we can see, distance got pretty good information about fare\_amount. We are not going to drop it.

Next, we did Anova test between categorical variables and target variable (continuous variable)

Let’s check that:

  
 IMAGE 13

**2.1.5 Feature Scaling**

In Feature Scaling, we try to limit the ranges of variables, so that they can be compared on the same ground.

Let’s talk about one example, consider, two variables, age and income. Age is varied from 1 to 100 but coming to income, it ranges in a large scale.

At this situation, the higher values bias the result towards themselves, in order to overcome a situation like this, we use Feature Scaling, where we limit the ranges of variables.

In our project, we did go for feature scaling for distance variable as the given data was left skewed. We also went for distribution check, so let’s have look on that.

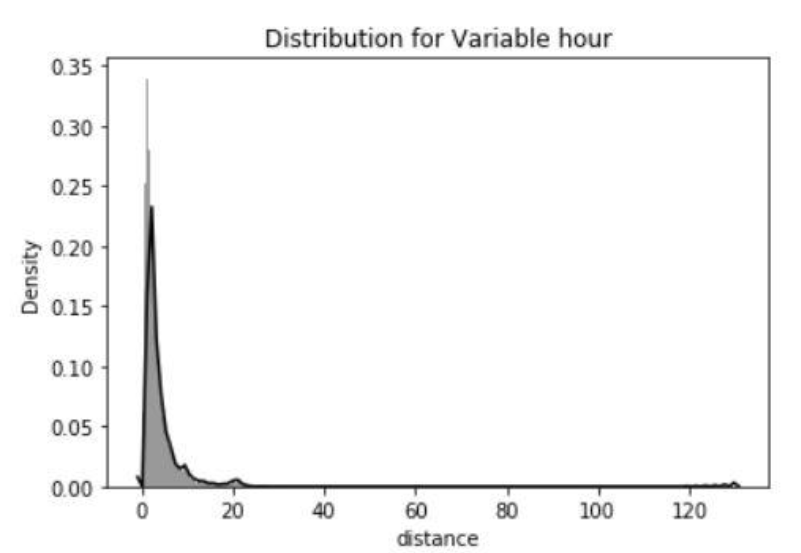
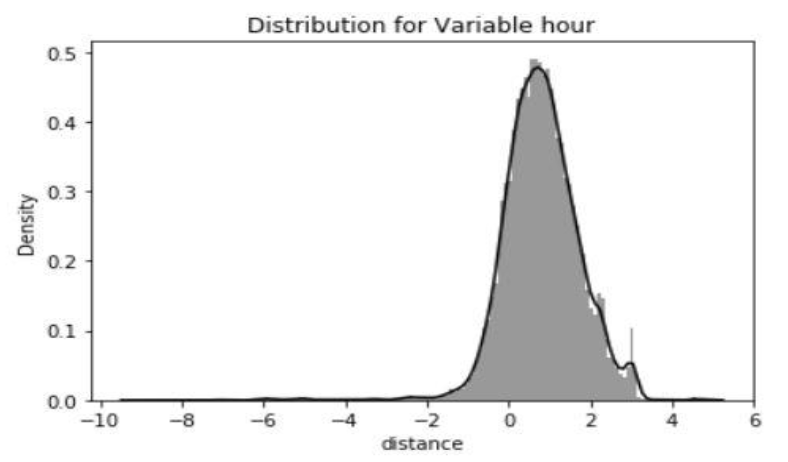


IMAGE 14

Now, as observed, the data points of distance variable are left skewed. Now, we are going to apply log to the data points and let’s check again, its distribution using histogram.



As we can see now, the distance variable got a pretty desired distribution. Finally, we are done with Feature Scaling. Now, let’s step into Model Development.

**2.2 Model Development**

Model Development, is the phase which comes after we are done with applying the exploratory data analysis, data pre-processing techniques, on the top of data.

The data, will be in structured format, which was our goal, is now ready to develop model.

After we defined our objective and received the data, we transformed it into our required form, we enter into model development, but before that, let’s discuss about model selection.

**2.2.1 Model Selection**

Model Selection particularly depends on the objective, the problem statement. We have to know at first hand, that, under which category, the problem statement falls.

We have four categories:

* Forecasting
* Classification
* Optimization
* Unsupervised Learning

Our problem statement is to **design a system that predicts the fare amount for a cab ride taken in the city.**

Our problem statement is a regression problem (target variable is a continuous variable) and it falls under Forecasting category.

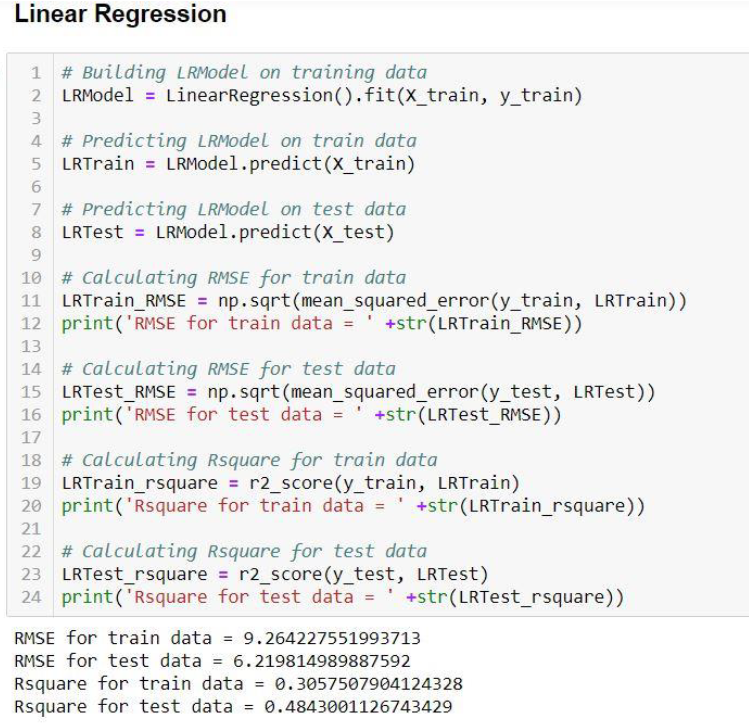
In our project we decided to go with, Linear Regression, Decision Tree, Random Forest, parameters hyper tuning of Random Forest with Randomized Search CV and Grid Search CV.

**2.2.2 Linear Regression**

Linear Regression is a supervised machine learning algorithm where the predicted output is continuous and has a constant slope. It’s used to predict values within a continuous range, (e.g. sales, price) rather than trying to classify them into categories (e.g. cat, dog).

Linear Regression, unlike other algorithms, stores information in terms of coefficients. It is a statistical model. We cannot use this for classification. It describes relationship among variables.

In our project, we get RMSE as 6.219815 and Rsquare as 0.484300. We are rejecting this model as RMSE is high and Rsquare is low when compared with all other models. Our aim is – we always want a model with low RMSE value i.e. minimum calculated errors and high R square value i.e. the independent variables should have maximum potential to explain about the target variable.



**2.2.3 Decision Tree**

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter.

The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes. And the decision nodes are where the data is split.

In our project, we get RMSE as 4.504301 and Rsquare as 0.729544. Although, Decision Tree performed better than Linear Regression, we are rejecting this model as RMSE is high and Rsquare is low when compared with Random Forest. Our aim is – we always want a model with low RMSE value i.e. minimum calculated errors and high R square value i.e. the independent variables should have maximum potential to explain about the target variable.

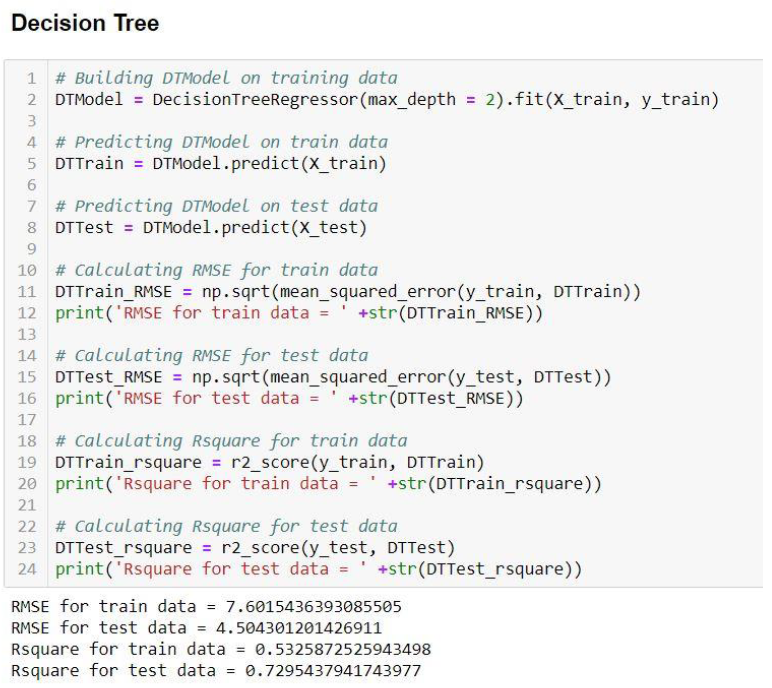


IMAGE 17

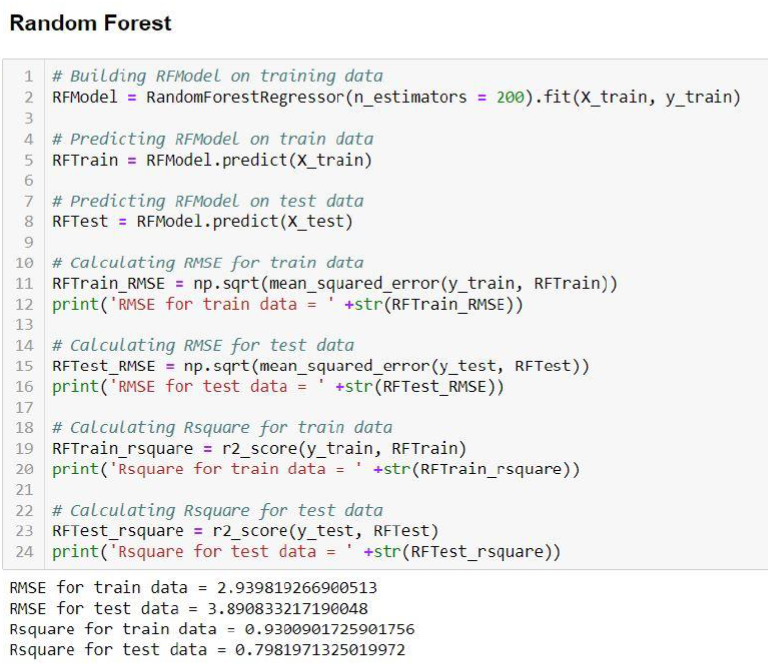
**2.2.4 Random Forest**

The Random Forest is a model made up of many decision trees. Rather than just simply averaging the prediction of trees (which we could call a “forest”), this model uses two key concepts that gives it the name *random:*

* Random sampling of training data points when building trees
* Random subsets of features considered when splitting nodes

The random forest combines hundreds or thousands of decision trees, trains each one on a slightly different set of the observations, splitting nodes in each tree considering a limited number of the features. The final predictions of the random forest are made by averaging the predictions of each individual tree.

In our project, we get RMSE as 3.870582 and Rsquare as 0.796779. Although, Random Forest performed better than Decision Tree, we are rejecting this model as RMSE is high at a small margin and Rsquare is low at a small margin when compared with Randomized Search CV. Our aim is – we always want a model with low RMSE value i.e. minimum calculated errors and high R square value i.e. the independent variables should have maximum potential to explain about the target variable

 IMAGE 18

**2.2.5 Grid Search CV**

Before talking about Grid Search CV, let me convey about Hyperparameter Tuning, which can be said as – Hyperparameter Tuning is choosing a set of optimal hyperparameters for a learning algorithm. Now, what is a hyperparameter - well, it can be defined as, it is a parameter whose value is set before the learning process begins.

There are two ways to do this – one is Grid Search and other one is Random Search. We will first discuss about Grid Search.

Grid search is a traditional way to perform hyperparameter optimization. It works by searching exhaustively through a specified subset of hyperparameters. Using sklearn’s GridSearchCV, we first define our grid of parameters to search over and then run the grid search.

The benefit of grid search is that it is guaranteed to find the optimal combination of parameters supplied. The drawback is that it can be very time consuming and computationally expensive. We can combat this with random search.

In our project, we get RMSE as 3.830760 and Rsquare as 0.804381. We will finally accept this model, as we got the best RMSE and Rsquare values when compared with all other models**.**

Our aim is – we always want a model with low RMSE value i.e. minimum calculated errors and high R square value i.e. the independent variables should have maximum potential to explain about the target variable



IMAGE 19

**2.2.6 Randomized Search CV**

Random search differs from grid search mainly in the matter that it searches the specified subset of hyperparameters randomly instead of exhaustively. The major benefit being decreased processing time.

There is a trade-off to decreased processing time. However, we aren’t guaranteed to find the optimal combination of hyperparameters.

Very similar to grid search above, we define the hyperparameters to search over before running the search.

An important additional parameter to specify here is n\_iter. This specifies the number of combinations to randomly try. Selecting too low of a number will decrease our chance of finding the best combination. Selecting too large of a number will increase our processing time.

In our project, we get RMSE as 3.887366 and Rsquare as 0.798557. It gave equal results with a little undesirable margin when compared with Random Forest, but we reject this model, as we found out that Grid Search CV is giving the best optimized values. Our aim is – we always want a model with low RMSE value i.e. minimum calculated errors and high R square value i.e. the independent variables should have maximum potential to explain about the target variable.



**Chapter 3   
Conclusion**

**3.1 Model Evaluation**

We always need a metric to evaluate the work we did. So, the same way, after we developed our models, we need a metric to validate the model we developed.

There are many metrics to evaluate, even, we have different metrics for classification problem and different metrics for regression problems

For classification problems, we have metrics like:

* Confusion matrix
* Accuracy
* Recall
* Specificity

For regression problems, we have metrics like:

* MSE
* RMSE
* MAPE
* Rsquare

We are choosing RMSE and Rsquare for our project. **Why RMSE over MAPE?**

Because, in RMSE, as the errors are squared before they are averaged, the RMSE gives a relatively high weightage to large errors, another reason is, RMSE penalizes large errors. For the above-mentioned reasons, we choose RMSE over MAPE.

**3.1.1 Root Mean Squared Error (RMSE) & Rsquare**

We are going to use RMSE and Rsquare as our error metrics to evaluate our models.

RMSE – Simply said, it is the sum of calculated errors.

Rsquare – Simply defined, correlation of original and predicted values.

**3.2 Model Selection**

Finally, it’s our Model selection time. We developed five models. Linear Regression, Decision Tree, Random Forest, GridSearchCV on Random Forest and RandomizedSearchCV on Random Forest.

We are going to freeze GridSearchCV with RMSE as 3.830760 and Rsquare as 0.804381, even though RandomizedSearchCV on Random Forest has low RMSE value than Random Forest, because Rsquare is our priority, it’s because, high Rsquare is equivalent to minimizing sum of squared errors, on the other hand, minimizing RMSE may yield biased point predictions.

PYTHON OUTPUT

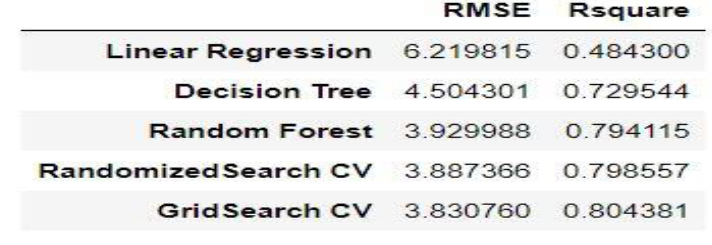
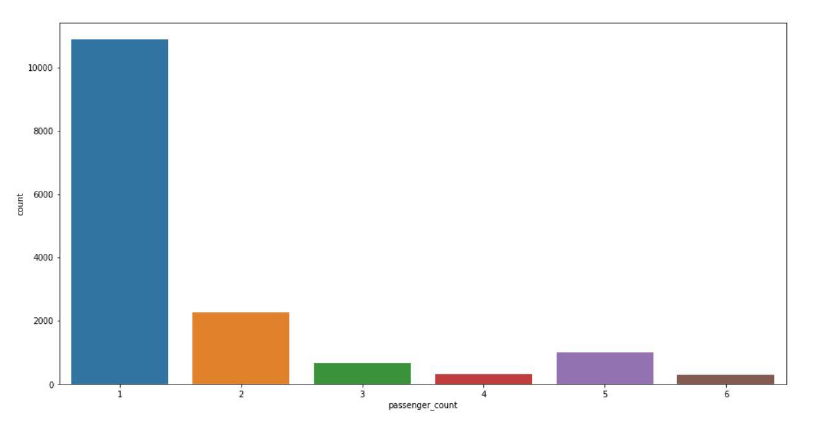
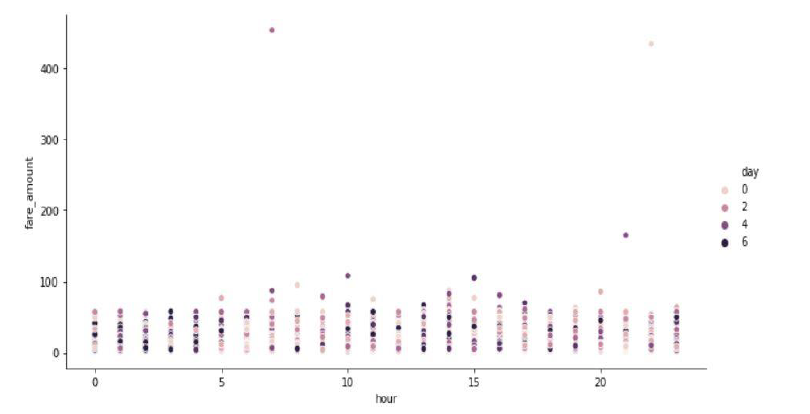


Image 21

R OUTPUT  


**Appendix A – Extra Figures**

  
single passengers are the most frequent travellers



The fare is highest when hour is 7am and 10pm, and where days are Monday, Thursday and Friday.

**Appendix B – R Script**

#cleaning the environment

rm(list = ls())

# Setting the working directory

setwd("C:\\Users\\aditya joshi\\Edwisor\\cab fare prediction")

#get data

getwd()

#loading the data

train\_cab = read.csv("train\_cab.csv")

##Exploratory Data Analysis

class(train\_cab)

#dataframe

head(train\_cab)

#shows first 6 observations

str(train\_cab)

#the strucutre of the dataframe

colnames(train\_cab)

#all the column names in the dataframe

class(train\_cab$fare\_amount)

#the variable is a factor variable however we need to change it to numerical variable

#converting factor into numeric

train\_cab$fare\_amount = as.numeric(train\_cab$fare\_amount)

#class of fare\_amount

class(train\_cab$fare\_amount)

#the variable is now numeric

# Fare\_amount has some missing values around 25, but they got replaced with 1, so we are removing those observations with 1

train\_cab = subset(train\_cab, fare\_amount != 1)

View(train\_cab)

####################################MISSING VALUE ANALYSIS##############################################

sum(is.na(train\_cab))

# Line of code to know missing values in respective columns

apply(train\_cab, 2, function(x) {sum(is.na(x))})

# Let's create a Missing Values dataframe for more clarity

missing\_values = data.frame(apply(train\_cab, 2, function(x){sum(is.na(x))}))

missing\_values$Columns = row.names(missing\_values)

names(missing\_values)[1] = "Missing\_values\_percentage"

View(missing\_values)

# Now, moving to calculate % of missing values

missing\_values$Missing\_values\_percentage = (missing\_values$Missing\_values\_percentage/nrow(train\_cab)) \* 100

# Have a glance at descending order of missing values percentage

missing\_values = missing\_values[order(-missing\_values$Missing\_values\_percentage),]

row.names(missing\_values) = NULL

# Re-ordering columns for better understanding

missing\_values = missing\_values[,c(2,1)]

# We got idea that the missing values percentage is negligible, so we are dropping the observations with missing values

train\_cab = na.omit(train\_cab)

sum(is.na(train\_cab))

#Total number of missing values are 0

##OUTLIER ANALYSIS

#FIRST WE TRY TO MANUALLY PLOT THE OUTLIERS

#fair\_amount variable

# Descending order to know outliers

train\_cab = train\_cab[order(-train\_cab$fare\_amount),]

# Ascending order to know inliers

train\_cab = train\_cab[order(train\_cab$fare\_amount),]

#no outliers in fare\_amount

#longitude and latitude

#as per the knowledge latitudes must range from -90 to +90 and longitudes must range from -180 to +180

#so we will just use summary function to check outliers

summary(train\_cab)

#In the console we can see that only pickup\_longitude has one value out of range i.e Max.:401.08

#removing outlier

train\_cab = subset(train\_cab, (pickup\_latitude > -90 & pickup\_latitude < +90))

summary(train\_cab)

#the outlier has been removed

# There are no outliers in pickup\_longitude, pickup\_latitude,dropoff\_longitude, dropoff\_latitude

#passenger\_count

#practically we can have minimum of 1 and maximum of 6 passengers.

# Descending order to know outliers

train\_cab = train\_cab[order(-train\_cab$passenger\_count),]

#Ascending order to know inliers

train\_cab = train\_cab[order(train\_cab$passenger\_count),]

#outliers detected

#removing outliers from passenger\_count

train\_cab = subset(train\_cab, passenger\_count > 0 & passenger\_count < 7) # Removed the outliers and inliers

train\_cab = subset(train\_cab, passenger\_count != 0.12) # Removed observation with passenger\_count value as 0.12

View(train\_cab)

# as longitude and latitude are still there With the concept of feature engineering,we decided to create distance variable

#Using Haversine Formula,we can create distance variable.

gcd.hf = function(long1, lat1, long2, lat2) {

R = 6371 # Earth mean radius [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 \* asin(min(1,sqrt(a)))

d = R \* c

return(d)

}

#creating new variable distance while storing the values of all 4 variables

for (i in 1:nrow(train\_cab)){

train\_cab$distance[i] = gcd.hf(train\_cab$pickup\_longitude[i], train\_cab$pickup\_latitude[i],

train\_cab$dropoff\_longitude[i], train\_cab$dropoff\_latitude[i])

}

#deleting the 4 variables which has been converted into distance

train\_cab = subset(train\_cab, select = -c(pickup\_latitude, pickup\_longitude, dropoff\_latitude, dropoff\_longitude))

#checking outliers in variable distance

train\_cab = train\_cab[order(-train\_cab$distance),]

train\_cab = train\_cab[order(train\_cab$distance),]

# Removing the outliers from the distance variable

train\_cab = subset(train\_cab, distance > 0 & distance <500)

#replacing UTC in pickup\_datetime variable with ''

train\_cab$pickup\_datetime = gsub('\\ UTC', '', train\_cab$pickup\_datetime)

train\_cab$Date = as.Date(train\_cab$pickup\_datetime)

train\_cab$year = substr(as.character(train\_cab$Date),1,4)

train\_cab$month = substr(as.character(train\_cab$Date),6,7)

train\_cab$weekday = weekdays(as.POSIXct(train\_cab$Date), abbreviate = F)

train\_cab$Date = substr(as.character(train\_cab$Date),9,10)

train\_cab$hour <- substr(as.factor(train\_cab$pickup\_datetime),12,13)

View(train\_cab)

train\_cab$time = NULL

train\_cab$pickup\_datetime = NULL

###########################VISUALIZATION##################################

library(ggplot2)

library(scales)

library(psych)

library(gplots)

# Visualization between fare\_amount and weekday.

ggplot(data = train\_cab, aes(x = reorder(weekday,-fare\_amount), y = fare\_amount))+

geom\_bar(stat = "identity")+

labs(title = "Fare Amount Vs. days", x = "Days of the week", y = "Fare")+

theme(plot.title = element\_text(hjust = 0.5, face = "bold"))+

theme(axis.text.x = element\_text( color="black", size=6, angle=45))

# We can understand that - Thursday and Saturday rides obtains the highest fare\_amount

# Visualization between fare\_amount and months.

ggplot(train\_cab, aes(x = reorder(month,-fare\_amount), y = fare\_amount))+

geom\_bar(stat = "identity")+

labs(title = "Fare Amount Vs. Month", x = "Month", y = "Fare")+

theme(axis.text.x = element\_text( color="navy blue", size=8))

#--> We can observe that - specific months liks January, March, June collects the highest fare\_amount

# Visualization between fare\_amount and years.

ggplot(data = train\_cab, aes(x = reorder(year,-fare\_amount), y = fare\_amount))+

geom\_bar(stat = "identity")+

labs(title = "Fare Amount Vs. days", x = "Days of the week", y = "Fare")+

theme(plot.title = element\_text(hjust = 0.5, face = "bold"))+

theme(axis.text.x = element\_text( color="black", size=6, angle=45))

#--> We can say, in year 2009 and 2010 there were rides which got high fare\_amount

# Visualization between fare\_amount and hour.

ggplot(data = train\_cab, aes(x = hour, y = fare\_amount))+

geom\_bar(stat = "identity")+

labs(title = "Fare Amount Vs. days", x = "Days of the week", y = "Fare")+

theme(plot.title = element\_text(hjust = 0.5, face = "bold"))+

theme(axis.text.x = element\_text( color="black", size=6, angle=45))

#--> Rides taken during 6 pm to 8 pm gives highest fare\_amount

# Visualization between fare\_amount and passenger\_count.

ggplot(data = train\_cab, aes(x = passenger\_count, y = fare\_amount))+

geom\_bar(stat = "identity")+

labs(title = "Fare Amount Vs. days", x = "Days of the week", y = "Fare")+

theme(plot.title = element\_text(hjust = 0.5, face = "bold"))+

theme(axis.text.x = element\_text( color="black", size=6, angle=45))

#--> Rides with single passenger will lead to highest fares

# Visualization between fare\_amount and distance.

ggplot(train\_cab,aes(distance,fare\_amount)) +

geom\_point(alpha=0.5) +

labs(title = "Fare amount based on distance", x = "Distance", y = "Fare Amount")+

scale\_color\_gradientn(colors=c('blue','light blue','dark blue','light green','yellow','dark orange','black')) +

theme\_bw()

#--> It's an obvious fact, fare\_amount is directly proportional to distance

#######################FEATURE SELECTION##################################

# Storing continuous variables

num\_var = c('fare\_amount', 'distance')

## Storing categorical variables

cat\_var = c('Date', 'year', 'month', 'weekday','passenger\_count','hour')

# Keeping data in another object

train\_cab2 = train\_cab

train\_cab = train\_cab2

# Correlation Analysis is performed between num\_var (continuous independent variables) & fare\_amount (continuous target variable)

library(corrgram)

corrgram(train\_cab[,num\_var],order=FALSE,upper.panel = panel.pie,

text.panel = panel.txt,

main= "Correlation Analysis between numeric variables")

for(i in cat\_var){

print(i)

Anova\_test\_result = summary(aov(formula = fare\_amount~train\_cab[,i],train\_cab))

print(Anova\_test\_result)

}

#-->> From the result, we can observe, passenger\_count, weekday and Date has p value > 0.05, by which, we accept null hypothesis.

# Deleting the below given continuous and categorical variables from day, as we found out they won't add any value to the model.

train\_cab = subset(train\_cab, select = -c(passenger\_count, weekday, Date))

####################################################FEATURE SCALING#################################################

# Checking distance variable distribution using histogram

hist(train\_cab$distance)

#--> The diagram represents, it is left skewed

#Checking summary of distance

summary(train\_cab$distance)

#--> Maximum value is 499.8405. We will go for normilisation

# We are going to define function using log

signedlog10 = function(x) {

ifelse(abs(x) <= 1, 0, sign(x)\*log10(abs(x)))

}

# Applying log function to distance variable

train\_cab$distance = signedlog10(train\_cab$distance)

# Checking distance distribution after applying function

hist(train\_cab$distance)

# Let's look at summary again

summary(train\_cab$distance)

# Let's manually remove the outliers in distance variable

train\_cab = subset(train\_cab, distance > 0)

###############################################MODEL EVALUATION##########################################################

# Let's clean R Environment, as it uses RAM which is limited

install.packages("DataCombine")

library(DataCombine)

rmExcept("train\_cab")

# Defining Error Metric, which are used to evaluate the model

# Defining R Square function - Correlation of original and predicted values.

Rsquare = function(y,y1){

cor(y,y1)^2

}

# Defining RMSE function - Root Mean Squared Errors (Calculated Errors)

# RMSE over MAPE, Why? - Because, RMSE gives weight to large errors during calculation and it penalizes large errors

RMSE = function(y,y1){

sqrt(mean((y - y1)^2))

}

train\_cab2 = train\_cab

train\_cab = train\_cab2

set.seed(123)

train\_index = sample(1:nrow(train\_cab),0.8\*nrow(train\_cab)) # Using Simple Random Sampling Technique

train= train\_cab[train\_index,]

test= train\_cab[-train\_index,]

####################################################LINEAR REGRESSION MODE###############################################

# Code for development of model

LRModel = lm(fare\_amount~., train)

# Let's have a view on the summary

summary(LRModel)

# Predicting model on train data

LRTrain = predict(LRModel, train[-1])

# Predicting model on test data

LRTest = predict(LRModel, test[-1])

# Calculating RMSE for Train Data

LRRMSE\_Train = RMSE(LRTrain, train[,1])

# Calculating RMSE for Test Data

LRRMSE\_Test = RMSE(LRTest, test[,1])

# Calculating Rsquare for Train Data

LRR2\_Train = Rsquare(train[,1], LRTrain)

# Calculating Rsquare for Test Data

LRR2\_Test = Rsquare(test[,1], LRTest)

#########################################DECISION TREE#######################################################

# Code to build Decision Tree

library(rpart)

DTModel = rpart(fare\_amount~., train, method = "anova")

# Predicting model on train data

DTTrain = predict(DTModel, train[-1])

# Predicting model on test data

DTTest = predict(DTModel, test[-1])

# Calculating RMSE for Train Data

DTRMSE\_Train = RMSE(DTTrain, train[,1])

# Calculating RMSE for Test Data

DTRMSE\_Test = RMSE(DTTest, test[,1])

# Calculating Rsquare for Train Data

DTR2\_Train = Rsquare(train[,1], DTTrain)

# Calculating Rsquare for Test Data

DTR2\_Test = Rsquare(test[,1], DTTest)

################################################RANDOM FOREST##########################################3

# Code for development of model

library(randomForest)

RFModel = randomForest(fare\_amount~., train, ntree = 500, method = "anova")

# Predicting model on train data

RFTrain = predict(RFModel, train[-1])

# Predicting model on test data

RFTest = predict(RFModel, test[-1])

# Calculating RMSE for Train Data

RFRMSE\_Train = RMSE(RFTrain, train[,1])

# Calculating RMSE for Test Data

RFRMSE\_Test = RMSE(RFTest, test[,1])

# Calculating Rsquare for Train Data

RFR2\_Train = Rsquare(train[,1], RFTrain)

# Calculating Rsquare for Test Data

RFR2\_Test = Rsquare(test[,1], RFTest)

#SAVING IN DATAFRAME#

Result = data.frame("Model" = c("Linear Regression", "Decision Tree", "Random Forest"),

"RMSE\_Values\_Train" = c(LRRMSE\_Train, DTRMSE\_Train, RFRMSE\_Train),

"RMSE\_Values\_Test" = c(LRRMSE\_Test, DTRMSE\_Test, RFRMSE\_Test),

"Rsquare\_Values\_Train" = c(LRR2\_Train, DTR2\_Train, RFR2\_Train),

"Rsquare\_Values\_Test" = c(LRR2\_Test, DTR2\_Test, RFR2\_Test))

#--> Random Forest gives the most optimized values. We are going to freeze Random Forest

# Now, it's time to bring the test data and perform the steps as performed on train\_cab and predict the fare\_amount

# Loading the data into our R environment

test\_cab = read.csv("test.csv")

#EXPLORITORY DATA ANALYSIS#

class(test\_cab) # Its DataFrame

head(test\_cab) # Let's have a look on first 6 observations

dim(test\_cab) # 9914 observations & 6 variables

str(test\_cab) # Have a look on the structure, pickup\_datetime is in factor

summary(test\_cab) # With a glance, we can get that, pickup\_latitude and passenger\_count don't have outliers

names(test\_cab) # In names, we can get to see as perfect naming for respective variables.

# We are going to change pickup\_datetime from factor to datetime

# But first, let's replace UTC in pickup\_datetime variable with ''

test\_cab$pickup\_datetime = gsub('\\ UTC', '', test\_cab$pickup\_datetime)

test\_cab$Date = as.Date(test\_cab$pickup\_datetime)

# We are familiar with working with variables like, hour, day, date, year but not a complete datetime variable.

# We are also not dealing with time series analysis, so, now, we are going to split the pickup\_datetime into its subsets

test\_cab$year = substr(as.character(test\_cab$Date),1,4)

test\_cab$month = substr(as.character(test\_cab$Date),6,7)

test\_cab$weekday = weekdays(as.POSIXct(test\_cab$Date), abbreviate = F)

test\_cab$Date = substr(as.character(test\_cab$Date),9,10)

test\_cab$time = substr(as.factor(test\_cab$pickup\_datetime),12,13)

# Now, we are going to delete pickup\_datetime variable, as we have already have its substitutes

test\_cab = subset(test\_cab, select = -c(pickup\_datetime))

#MISSING VALUE ANALYSIS#

# Line of code to know the sum of missing values in dataset

sum(is.na(test\_cab)) # Total number of missing values are 0

#OUTLIER ANALYSIS#

# Coming to outliers, let's check summary and check for outliers

summary(test\_cab)

#--> It clearly shows, there are no outliers.

# Now, let's create distance using Haversine Formula as did in train\_cab

# Calculates the geodesic distance between two points specified by radian latitude/longitude using the Haversine formula (hf)

gcd.hf = function(long1, lat1, long2, lat2) {

R = 6371 # Earth mean radius [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 \* asin(min(1,sqrt(a)))

d = R \* c

return(d) # Distance in km

}

# Now, we are going to apply the function, over all the rows to create a new variable - distance

for (i in 1:nrow(test\_cab)){

test\_cab$distance[i] = gcd.hf(test\_cab$pickup\_longitude[i], test\_cab$pickup\_latitude[i],

test\_cab$dropoff\_longitude[i], test\_cab$dropoff\_latitude[i])

}

# Dimension Reduction - we are going to delete latitude and longitude variables as we obtained distance from these four variables

test\_cab = subset(test\_cab, select = -c(pickup\_latitude, pickup\_longitude, dropoff\_latitude, dropoff\_longitude))

# Now, its time to check outliers in distance

test\_cab = test\_cab[order(-test\_cab$distance),]

test\_cab = test\_cab[order(test\_cab$distance),]

# Removing the outliers from the distance variable

test\_cab = subset(test\_cab, distance > 0 & distance <500)

# To match the number of features of test data to the model, we are going to delete few variables

test\_cab = subset(test\_cab, select = -c(passenger\_count, weekday, Date))

# We are going to normalize the data in distance variable

# Checking distance variable distribution using histogram

hist(test\_cab$distance)

#--> The diagram represents, it is left skewed

# Checking summary of distance

summary(test\_cab$distance)

#--> Maximum value is 499.8466. We will go for normilisation

# We are going to define function using log

signedlog10 = function(x) {

ifelse(abs(x) <= 1, 0, sign(x)\*log10(abs(x)))

}

# Applying log function to distance variable

test\_cab$distance = signedlog10(test\_cab$distance)

# Checking distance distribution after applying function

hist(test\_cab$distance)

# Let's look at summary again

summary(test\_cab$distance)

# Let's manually remove the outliers in distance variable

test\_cab = subset(test\_cab, distance > 0)

# We are near to predict our cab fare using Random Forest

# Code for development of model

RFModel = randomForest(fare\_amount~., train, ntree = 500, method = "anova")

# Predicting model on test\_cab data

RFTest\_cab = predict(RFModel, test\_cab)

# Adding our obtained predictions as Predicted Fare Amount variable to test\_cab dataset

test\_cab$Predicted\_fare\_amount = RFTest\_cab

#predicted values

head(test\_cab)

#OBJECTIVE ACHIEVED.

**REFRENCES**edwisor   
edwisor community

google  
haversine formula - <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

Grid search cv - <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html>

Random search cv - <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html>