**PROJECT REPORT**

**ON**

**CAB FARE PREDICTION**

**By-**

**KAUSHIK MAZUMDAR**

**INDEX**

Chapter 1: Introduction

1.1 Problem Statement

1.2 Data

Chapter2: Methodology

• Pre-Processing

• Modelling

• Model Selection

Chapter 3: Pre-Processing

3.1 Visualization & Data Exploration

* Univariate Analysis
* Bivariate Analysis

3.2 Data exploration and Cleaning

3.3 Creating some new variables from the given variables

3.4 Selection of variables

3.5 Some more data exploration

• Dependent and Independent Variables

• Uniqueness of Variables

• Dividing the variables categories

3.6 Missing Value Analysis

3.7 Outlier Analysis

3.8 Feature Selection

3.9 Feature Scaling

Chapter 4: Modelling

4.1 Linear Regression

4.2 K Neighbors Regressor

4.3 Decision Tree

4.4 Random Forest

4.5 Gradient Boosting

Chapter 5: Conclusion

5.1 Model Evaluation

5.2 Model Selection

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.

1.1 Problem Statement

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

1.2 Data

Understanding of data is the very first and important step in the process of finding solution of any business problem. 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 and for better functioning.

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 |

CHAPTER 2

Methodology

* Pre-Processing

When we required to build a predictive model, we require to look and manipulate the data before we start modelling which includes multiple pre-processing 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 Analysis

• Outlier Analysis

• Feature Selection

• Features Scaling

* Modelling

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modelling. Modelling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we will try some models on our pre-processed data and post comparing the output results we will select the best suitable model for our problem. As per our data set following models need to be tested:

• Linear regression

• KNeighbors regressor

• Decision Tree

• Random forest

• Gradient Boosting

* We have also used hyper parameter tunings to check the parameters on which our model runs best. Following techniques of hyper parameter tuning we have used:

o Grid Search CV

* Model Selection

The final step of our methodology will be the selection of the model based on the different output and results shown by different models. We have multiple parameters which we will study further in our report to test whether the model is suitable for our problem statement or not.

CHAPTER 3

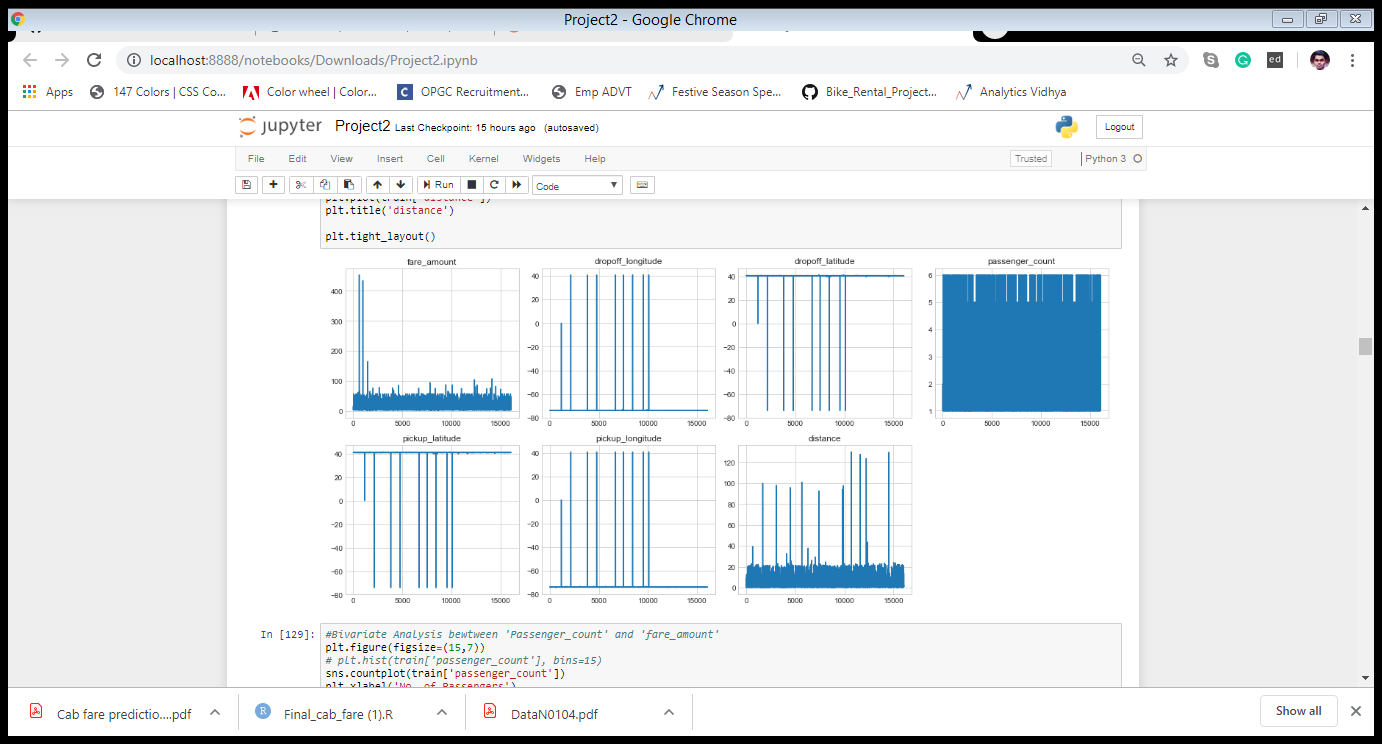
Pre-Processing

3.1 Visualization & Data Exploration

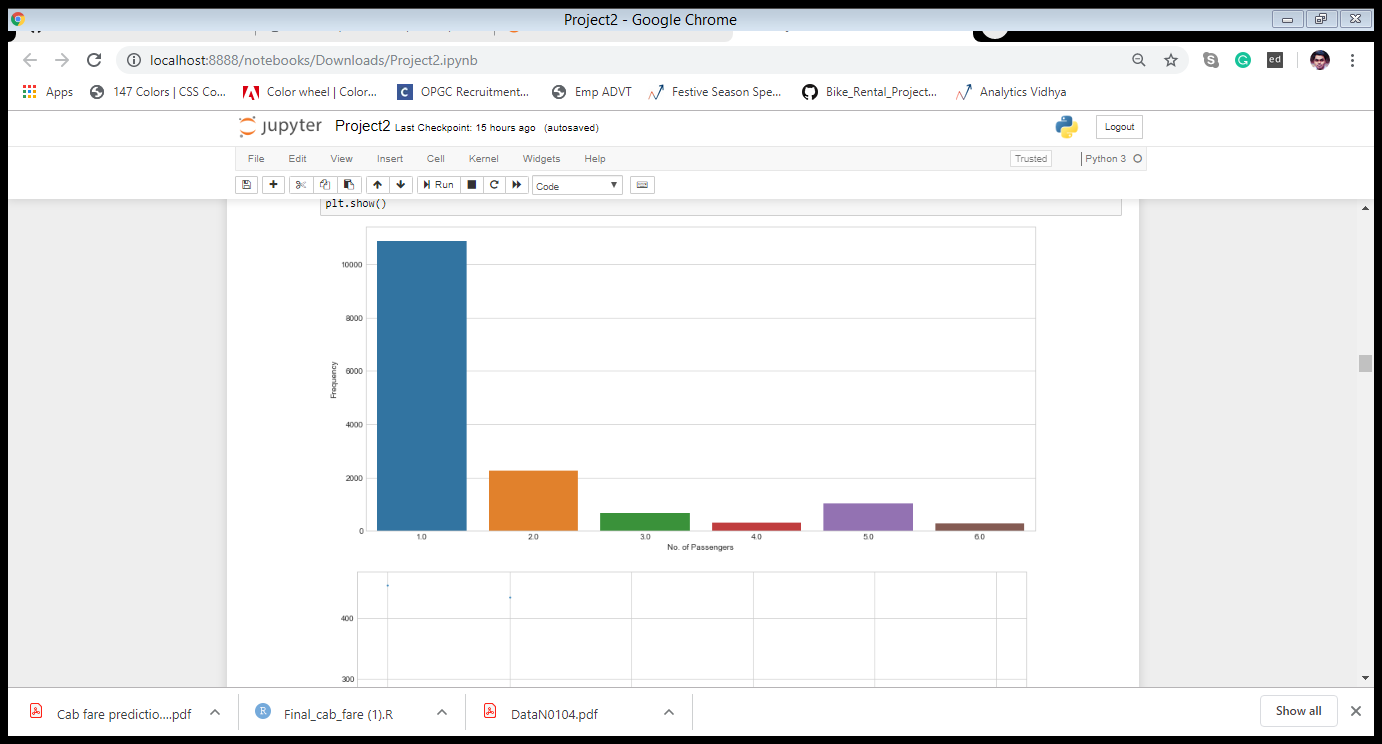
Data Exploration is one the most important steps while we try to infer any insightful information from the data. Visualization techniques make our work way easier and help us to gain a glance over the whole dataset and enables us to from an opinion about the nature of the dataset. It also enables us to compare among different variables to give us insightful information about the data.

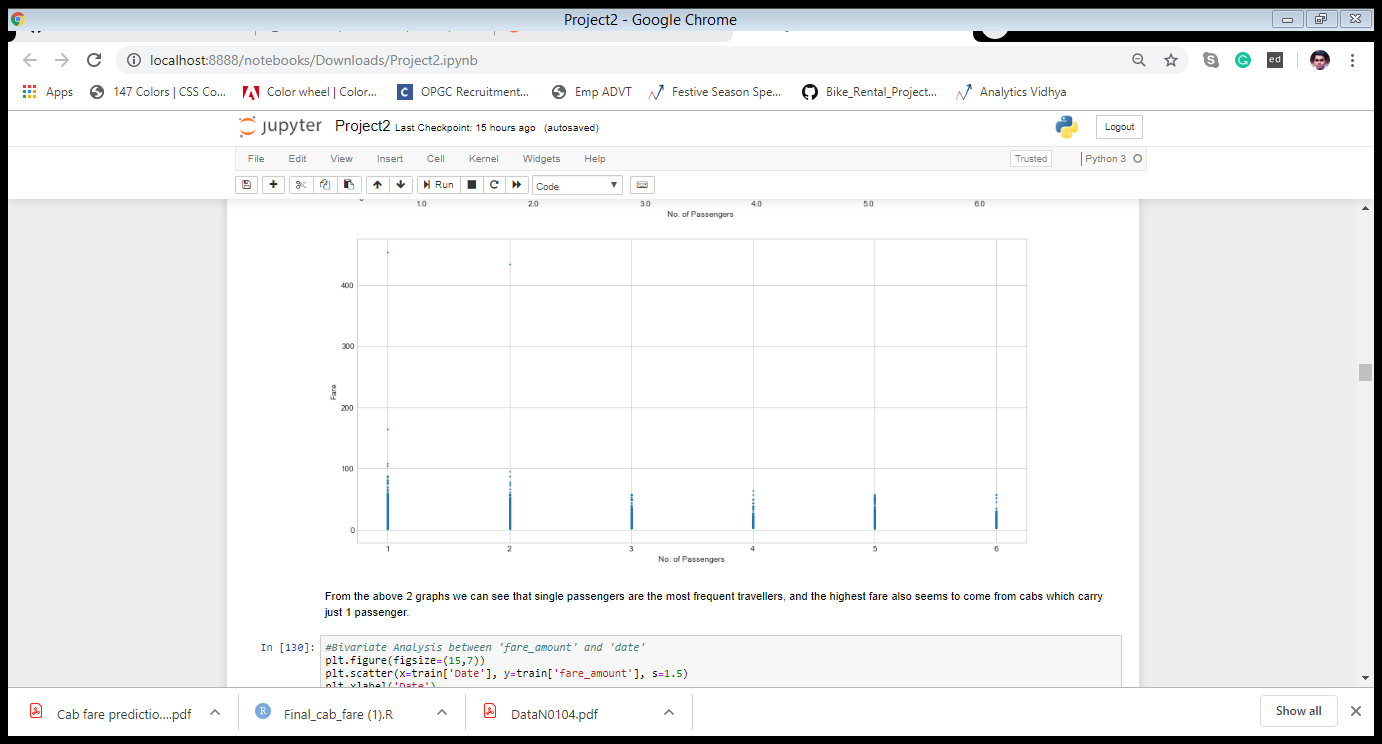
* Univariate Analysis: Well, when a single variable is analysed among itself, it is known as univariate Analysis . It is useful to gain knowledge about the nature of the dataset contained in the variable and how it changes among itself

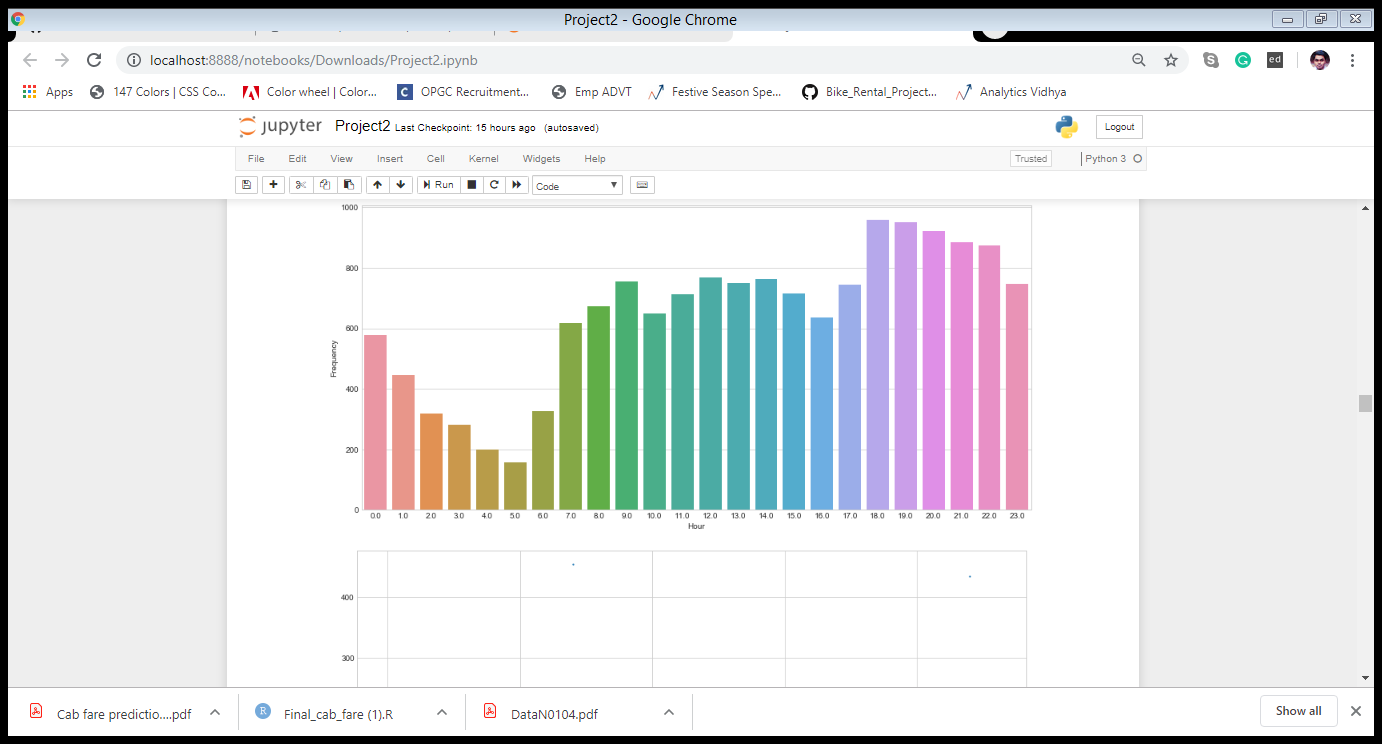
Here is a screenshot of the univariate analysis of the variables in the train dataset

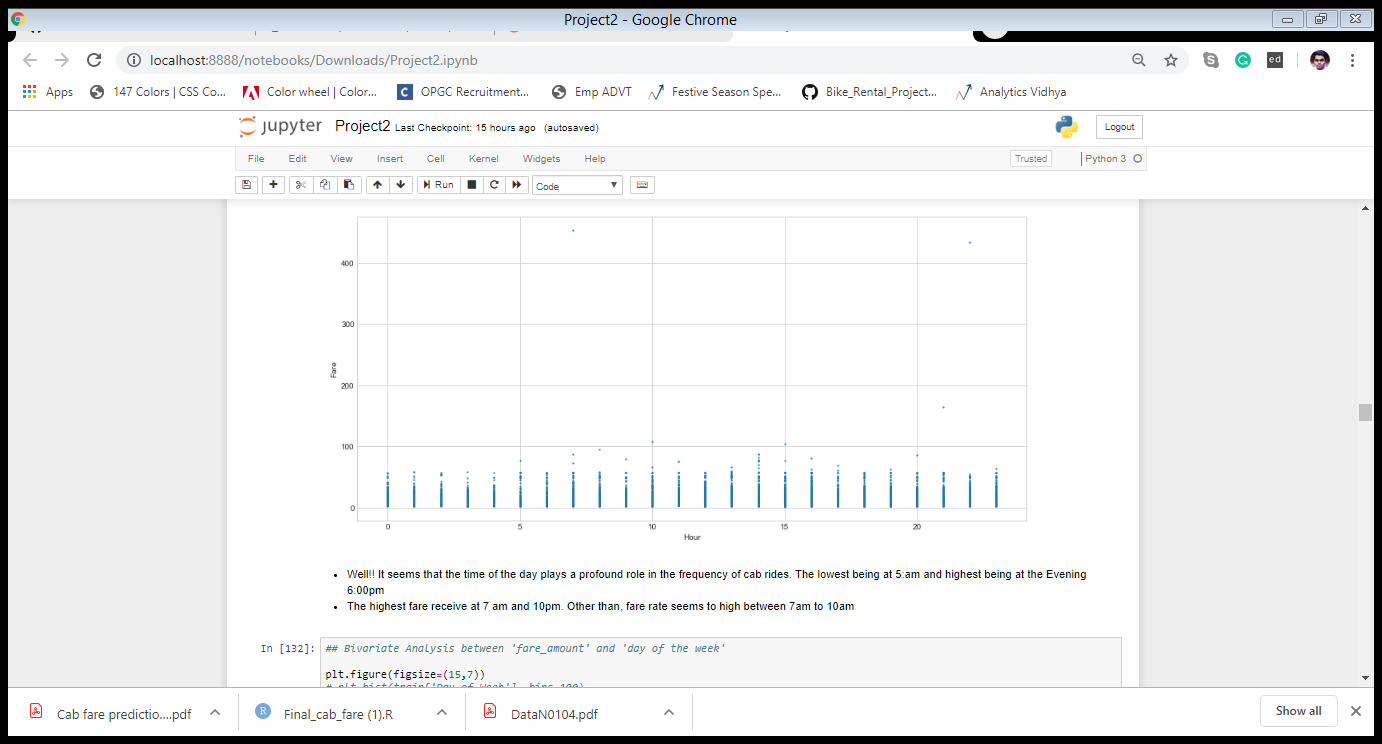


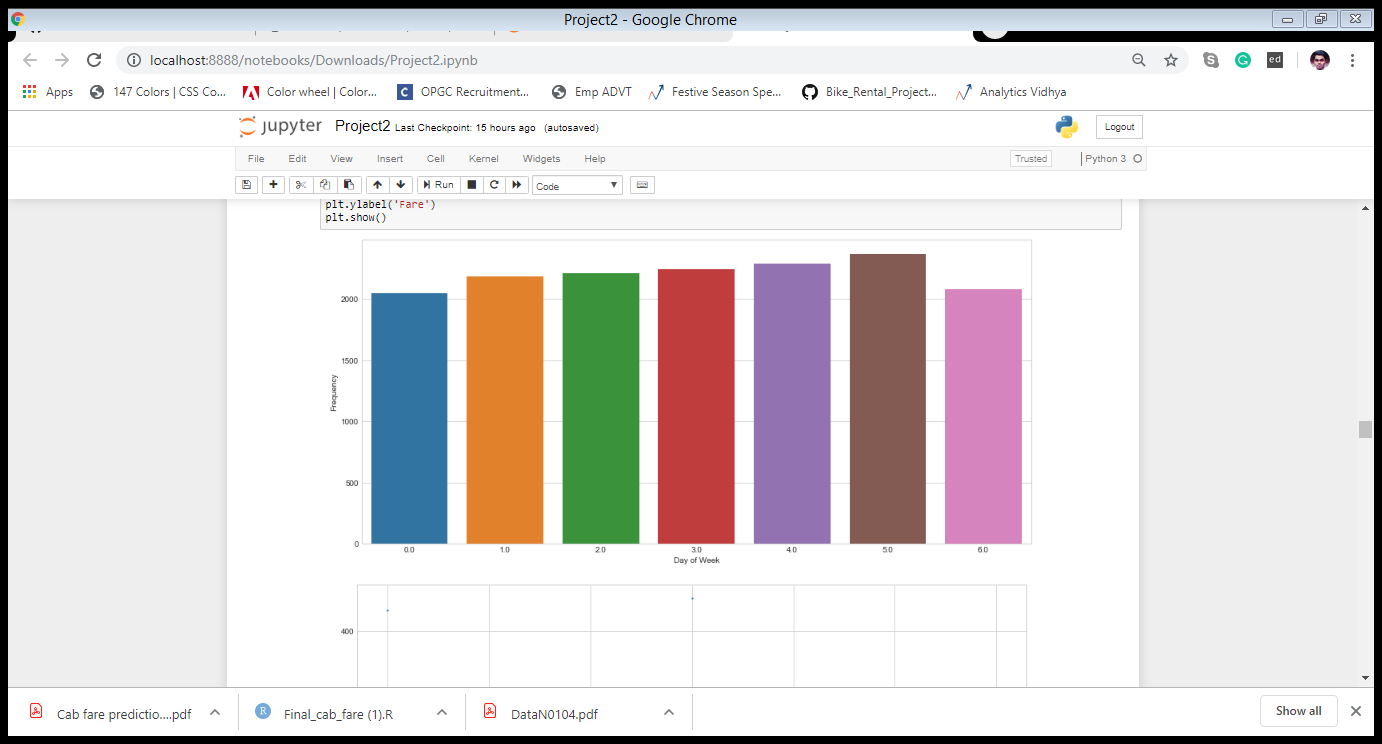
* Bivariate Analysis: It is the analysis of different variables with each other or with the target variable. It is very useful as it enables to see how one variable changes according to another variable to ascertain if it has any impact on the other variable. Variables are even compare with target variable to see how one trends with respect to Target variable

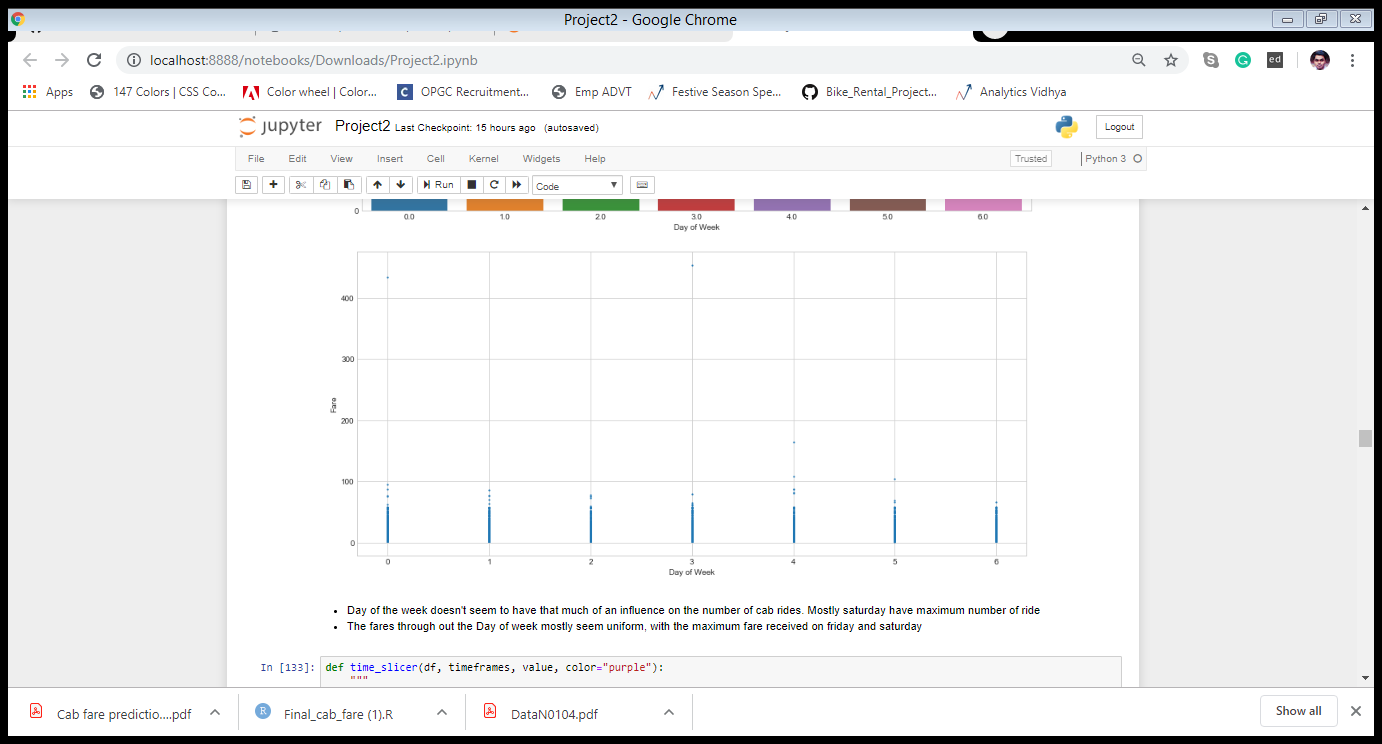
Analysis between ’Passenger\_count’ and ‘fare amount’

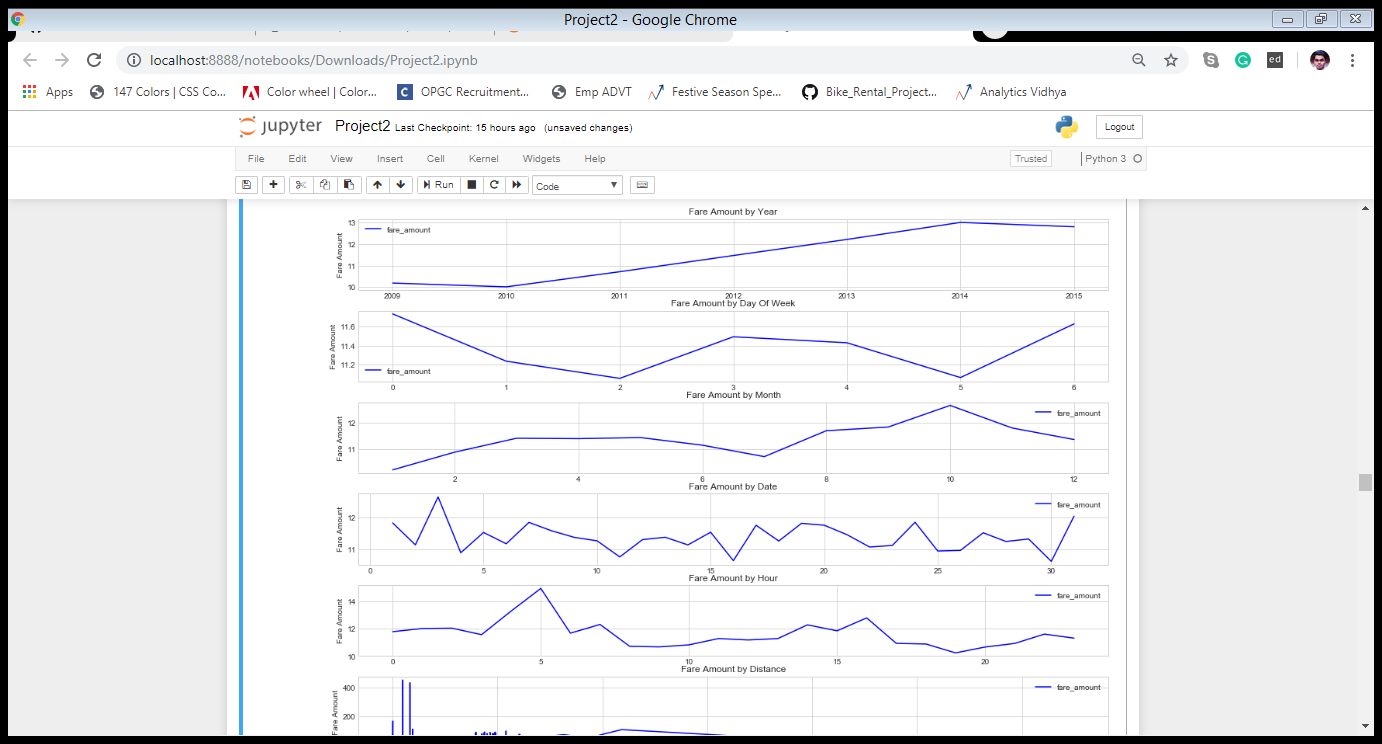


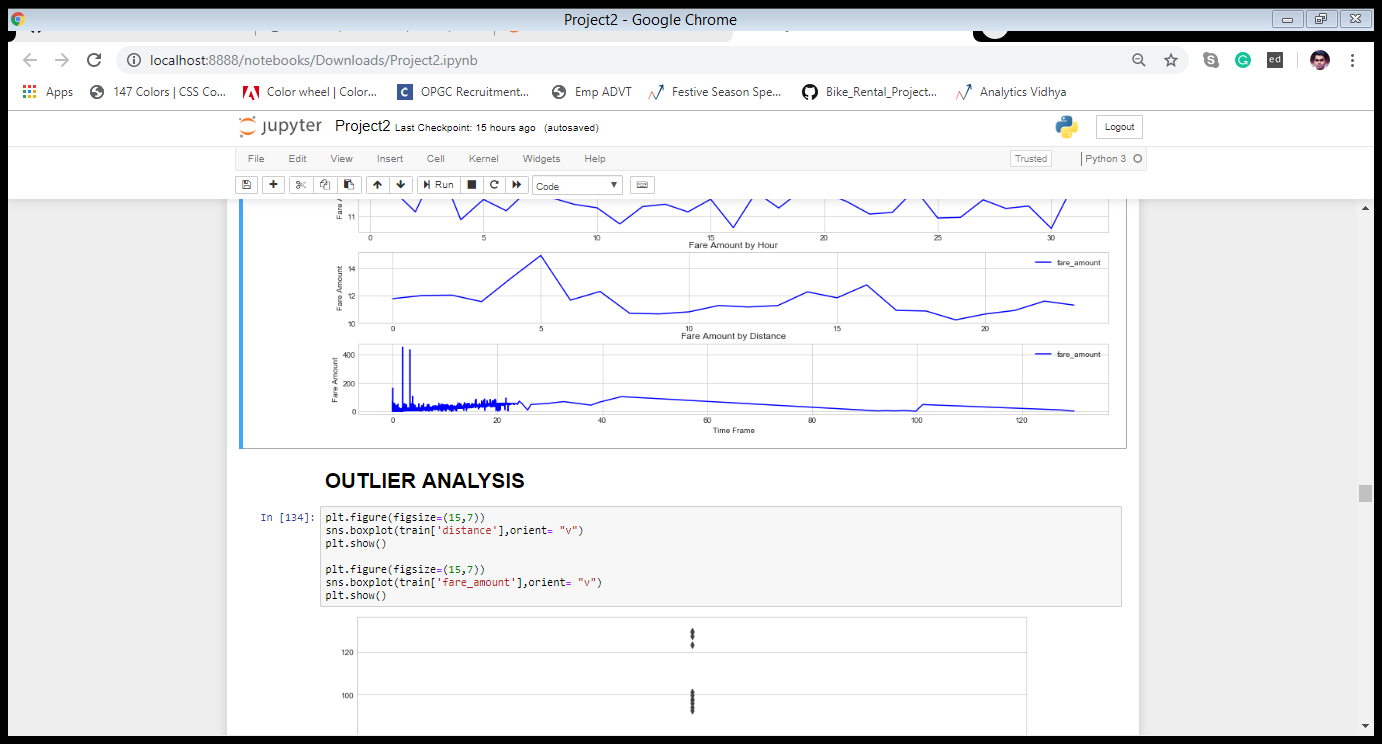
Analysis between ‘fare amount’ and ‘hour’



Analysis between ‘fare amount ‘ & ‘day of the week’



Analysis of different Variables with different lense of Time

3.2 Data exploration and Cleaning

The very first step which comes with any data science project is data exploration and cleaning which includes following points as per this project:

a. Separate the combined variables.

b. As we know we have some negative values in fare amount so we have to remove those values.

c. Passenger count would be max 6 if it is a SUV vehicle not more than that. We have to remove the rows having passengers counts more than 6 and less than 1.

d. There are some outlier figures in the fare so we need to remove those.

e. Latitudes range from -90 to 90. Longitudes range from -180 to 180. We need to remove the rows if any latitude and longitude lies beyond the ranges.

3.3 Creating some new variables from the given variables.

Here in our data set our variable name pickup\_datetime contains date and time for pickup. So we tried to extract some important variables from pickup\_datetime:

• Year

• Month

• Date

• Day of Week

• Hour

• Minute

Also, we tried to find out the distance using the haversine formula which says:

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines , that relates the sides and angles of spherical triangles

. So our new extracted variables are:

♣ fare\_amount

♣ pickup\_longitude

♣ pickup\_latitude

♣ dropoff\_longitude

♣ dropoff\_latitude

♣ passenger\_count

♣ year

♣ Month

♣ Date

♣ Day of Week

♣ Hour

♣ Minute

♣ Distance

3.4 Selection of variables

Now as we know that all above variables are of now use so we will drop the redundant variables:

♣ pickup\_datetime

♣ pickup\_longitude

♣ pickup\_latitude

♣ dropoff\_longitude

♣ dropoff\_latitude

Now only following variables we will use for further steps:

|  |  |
| --- | --- |
| Variables | Variable Data-Types |
| Fare\_amount | float64 |
| Year | object |
| Month | Object |
| Date | Object |
| Day of Week | Object |
| Hour | object |
| Passenger\_count | object |
| Distance | float64 |

3.5 Some more data exploration

In this report we are trying to predict the fare prices of a cab rental company. So here we have a data set of 16067 observations with 8 variables including one dependent variable.

3.5.1 Below are the names of Independent variables:

passenger\_count, year, Month, Date, Day of Week, Hour, distance

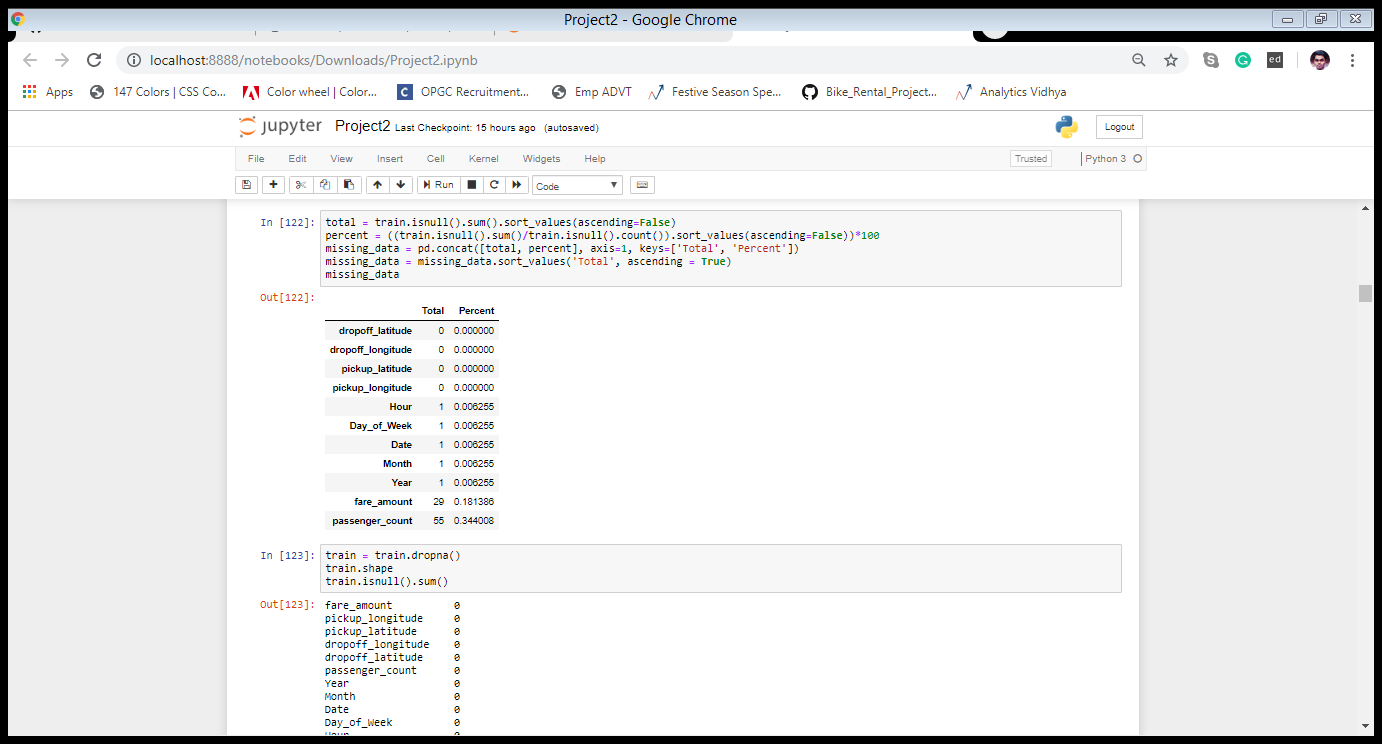
Our Dependent variable is: fare\_amount

3.5.2 Dividing the variables into two categories basis their data types:

Continuous variables - 'fare\_amount', 'distance'.

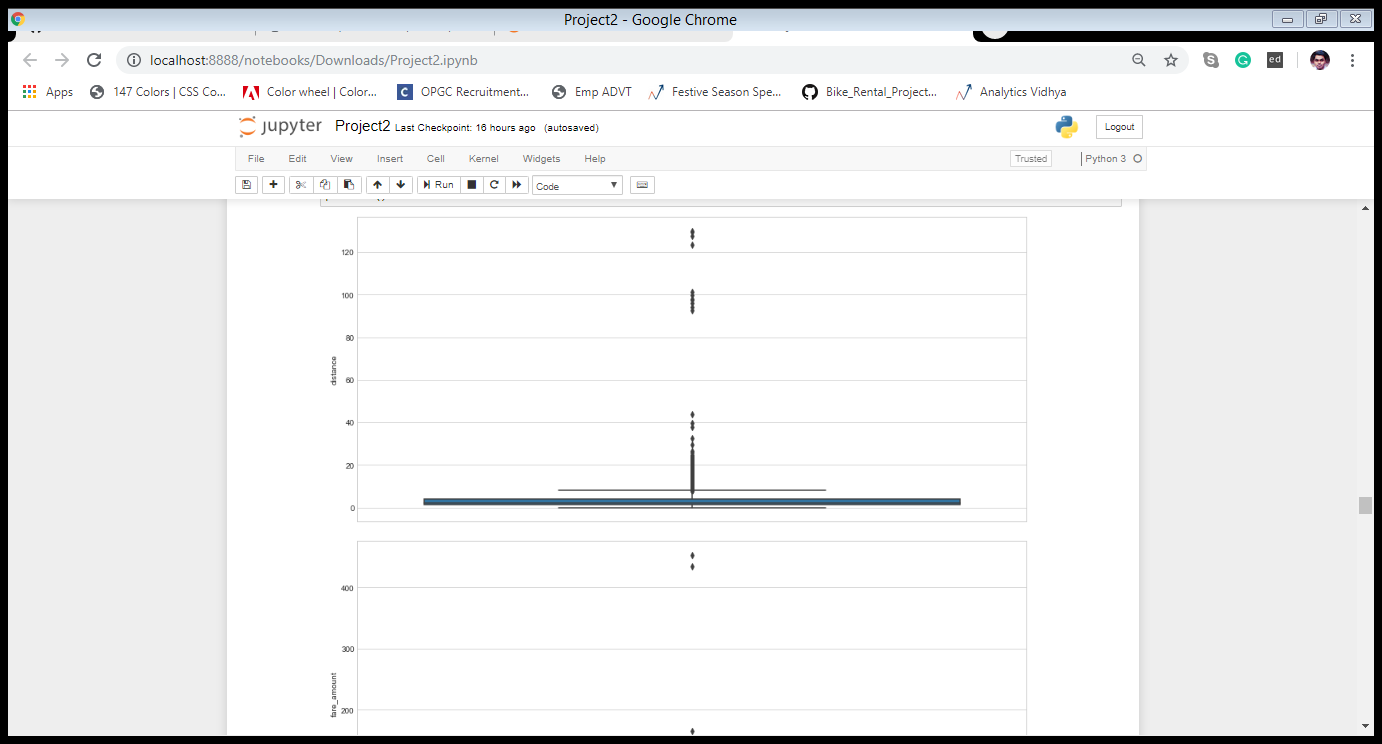
Categorical Variables - 'year', 'Month', 'Date', 'Day of Week', 'Hour', 'passenger\_count'

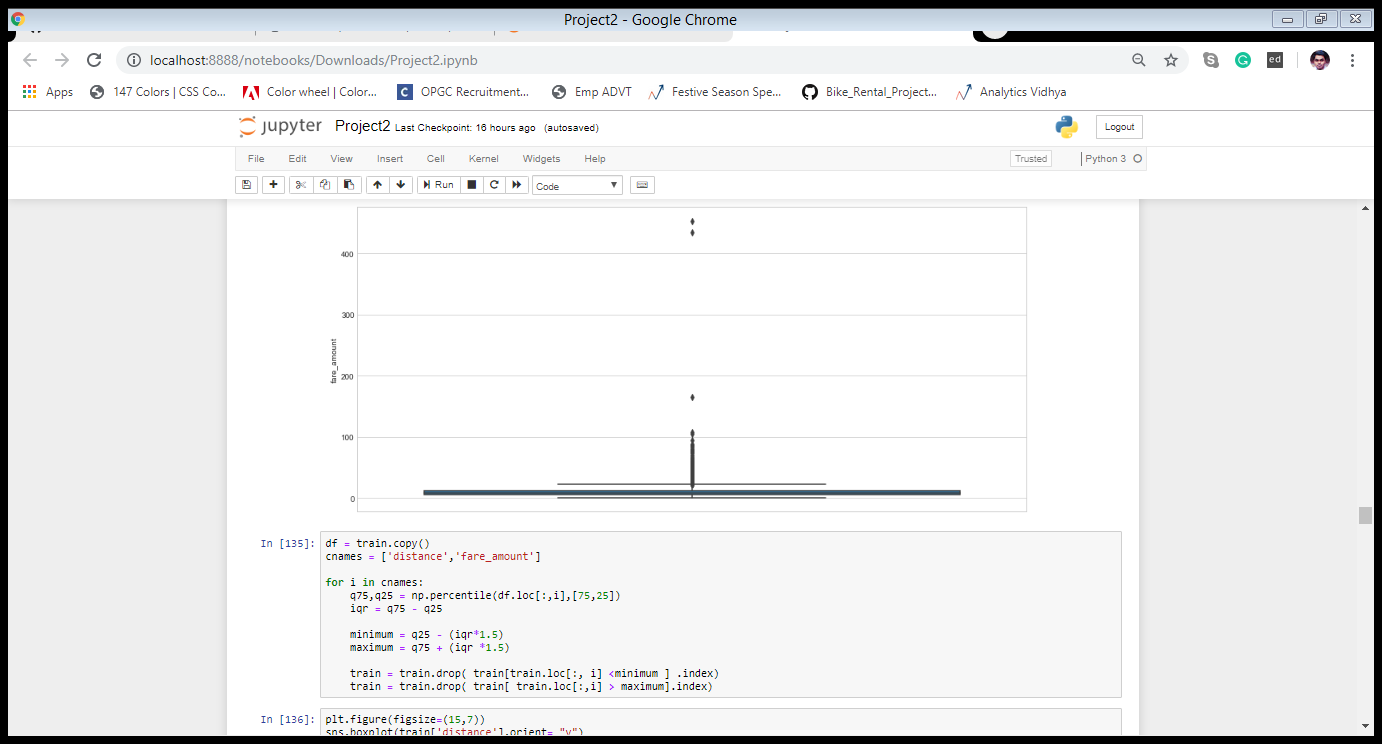
3.6 Missing Value Analysis

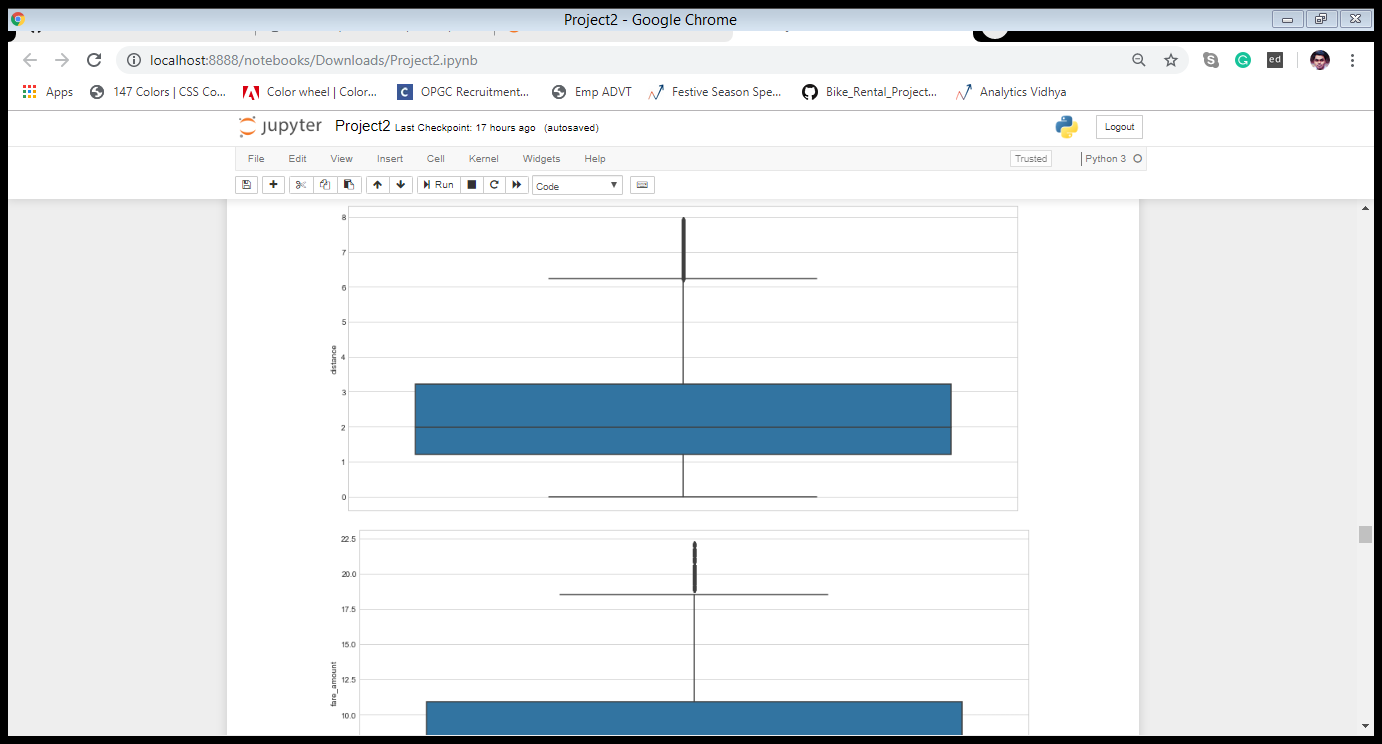
Missing values in data is a common phenomenon in real world problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models. Below table illustrate no missing value present in the data.

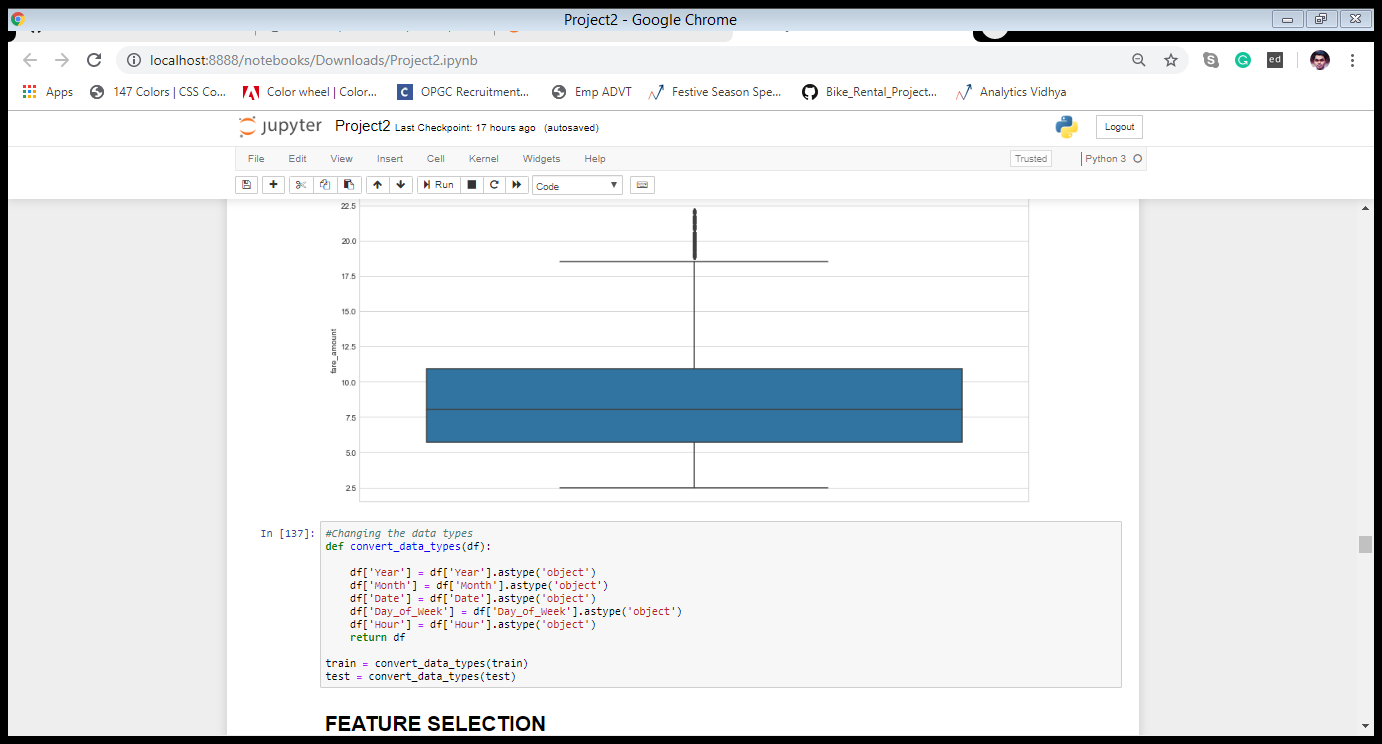
3.7 Outlier Analysis

The Other steps of Pre-processing Technique is Outliers analysis, an outlier is an observation point that is distant from other observations. Outliers in data can distort predictions and affect the accuracy, if you don’t detect and handle them appropriately especially in regression models. As we observed , there is presence of outlier in independent variable ‘distance’ and ‘fare\_amount’,

Outliers present in ‘distance’ and ‘fare\_amount’



‘distance’ & ‘fare\_amount’ after the removal of Outliers



3.8 Feature Selection

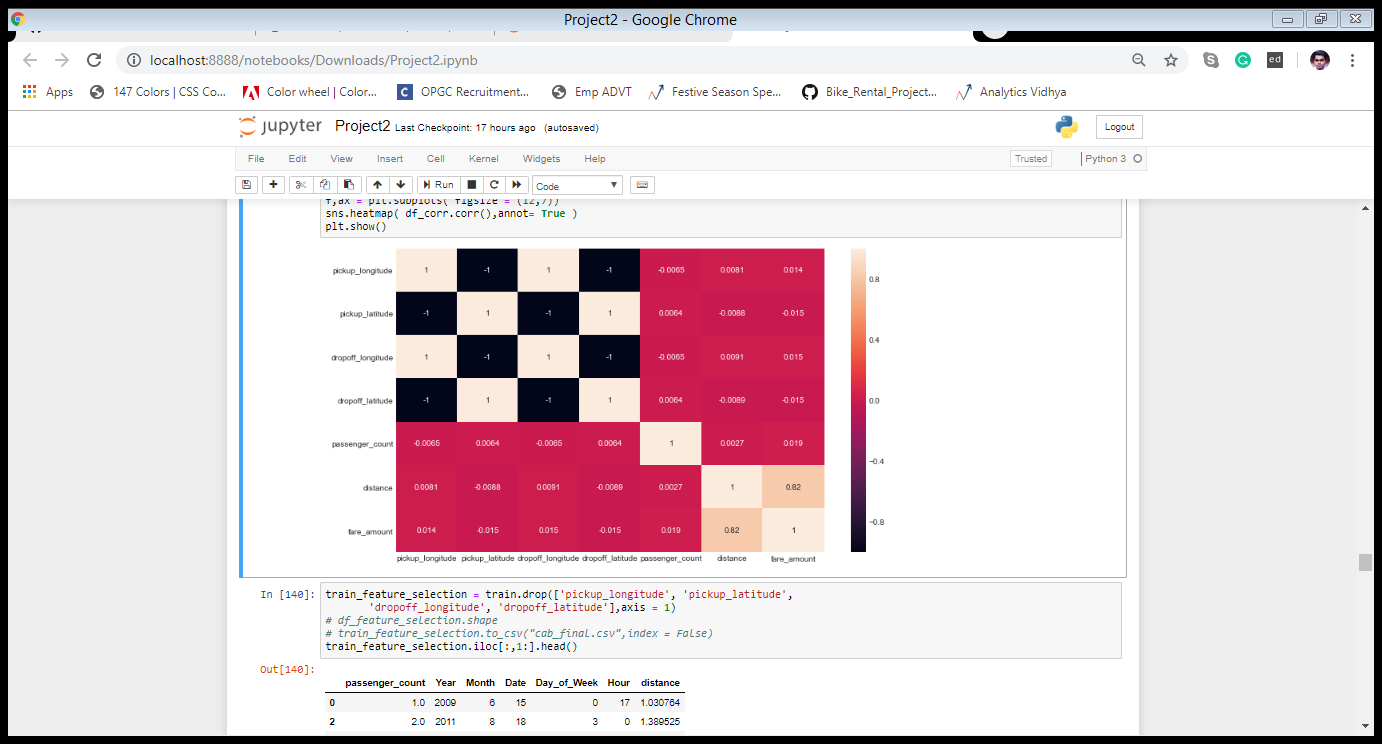
Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.

This becomes even more important when the number of features are very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. I have myself witnessed feature subsets giving better results than complete set of feature for the same algorithm or – “Sometimes, less is better!”

We should consider the selection of feature for model based on below criteria

i. The relationship between two independent variable should be less and

ii. The relationship between Independent and Target variables should be high.

 Below fig . illustrates that relationship between all numeric variables using Corrgram plot.

Correlation plot of numeric variables Color pink indicates there is strong positive relationship and if darkness is decreasing indicates relation between variables are decreasing.

Color black indicates there is strong negative relationship and if darkness is decreasing indicates relationship between variables are decreasing.

Corrgram : it help us visualize the data in correlation matrices. correlograms are implimented through the corrgram(x, order = , panel=, lower.panel=, upper.panel=, text.panel=, diag.panel=)

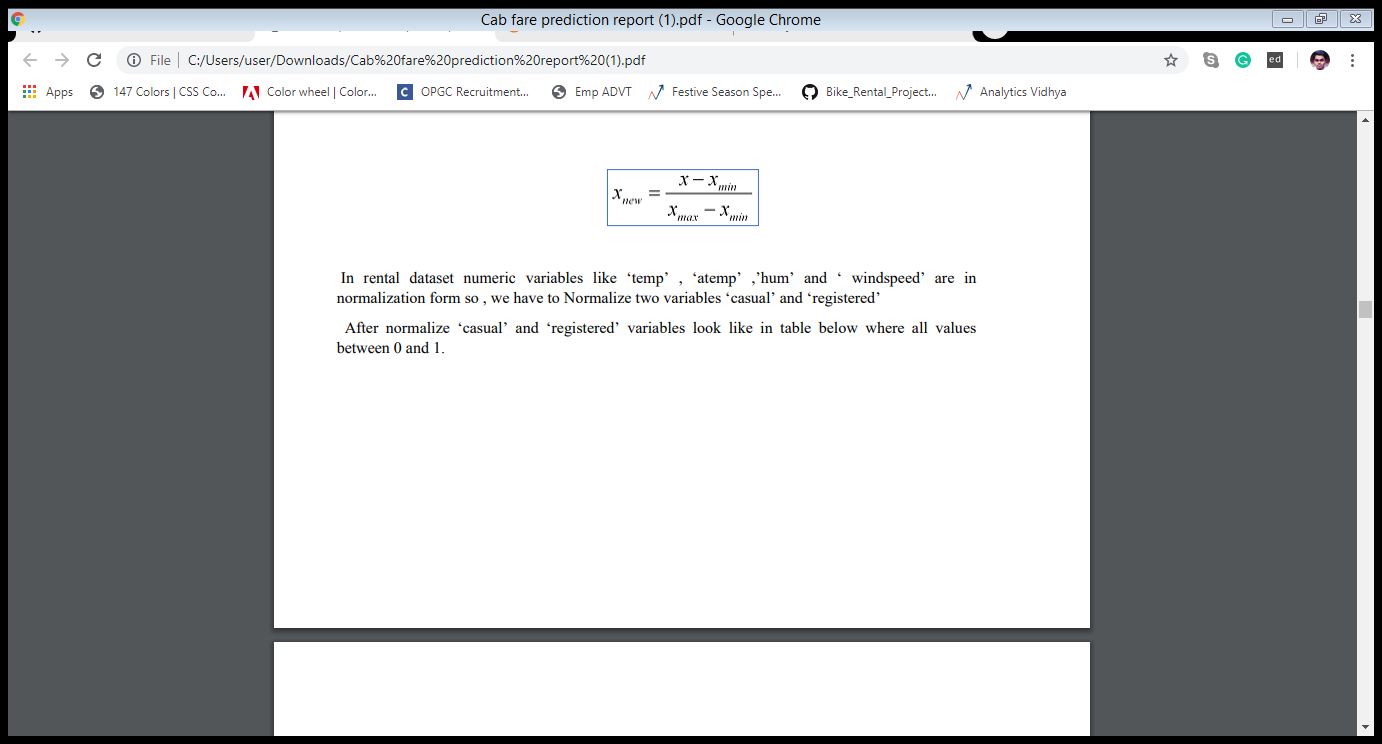
**Dimensionality Reduction for numeric variables**

Above Fig s also showing there is almost no relationship between independent variables like ‘pickup\_datetime’,’pickup\_latitude’,’pickup\_longitude’,‘dropoff\_latitude’,‘dropoff\_longitude’ and dependent variable ‘fare\_amount’.

So, these variable are not so important to predict. Subsetting independent features like ’pickup\_latitude’,’pickup\_longitude’ ‘dropoff\_latitude’ and ‘dropoff\_longitude’ from actual dataset.

3.9 Feature Scaling

The word “normalization” is used informally in statistics, and so the term normalized data can have multiple meanings. In most cases, when you normalize data you eliminate the units of measurement for data, enabling you to more easily compare data from different places. Some of the more common ways to normalize data include:

****Transforming data using a z-score or t-score. This is usually called standardization. In the vast majority of cases, if a statistics textbook is talking about normalizing data, then this is the definition of “normalization” they are probably using. Rescaling data to have values between 0 and 1. This is usually called feature scaling. One possible formula to achieve this is.

In rental dataset numeric variables like ‘temp’ , ‘atemp’ ,’hum’ and ‘ windspeed’ are in normalization form so , we have to Normalize two variables ‘casual’ and ‘registered’ After normalize ‘casual’ and ‘registered’ variables look like in table below where all values between 0 and 1.

CHAPTER 4

Modelling

After a thorough pre-processing, we will use some regression models on our processed data to predict the target variable. Following are the models which we have built –

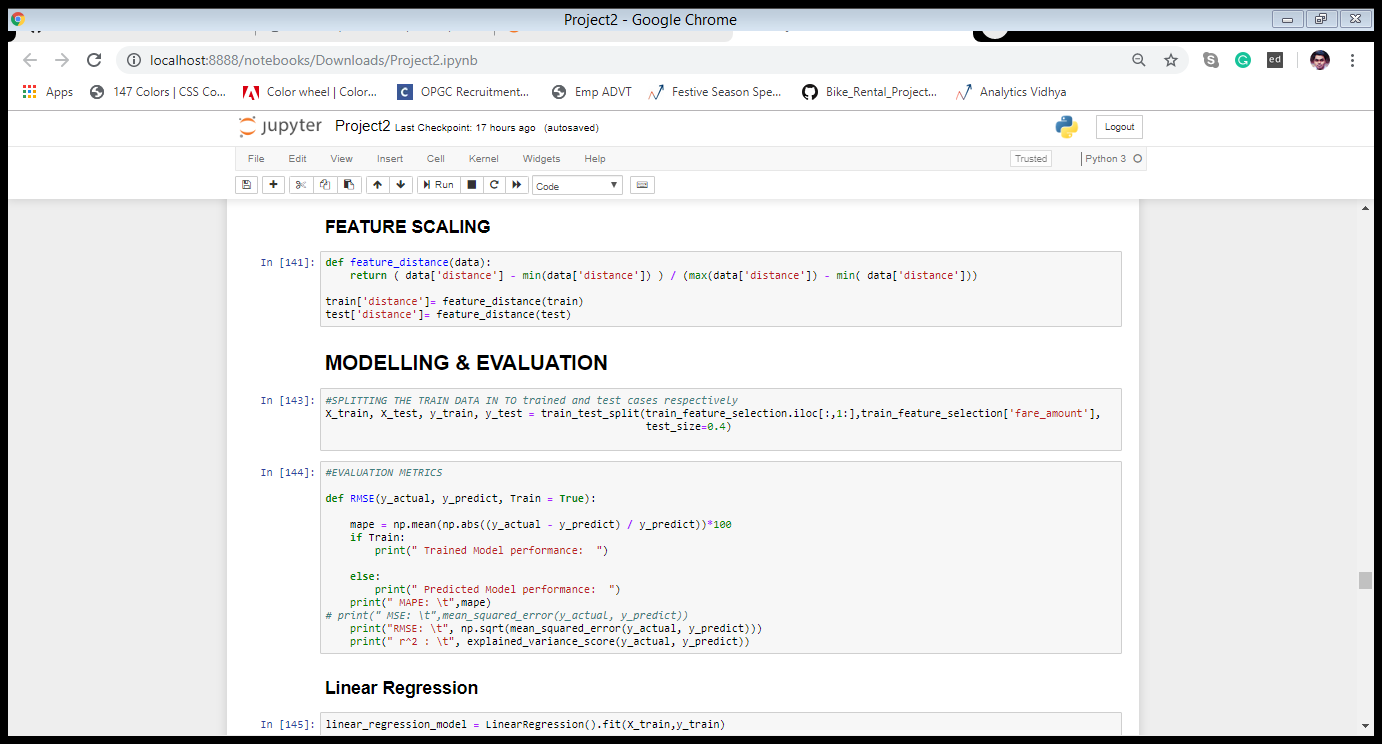
• Linear Regression

• Decision Tree

• Random Forest

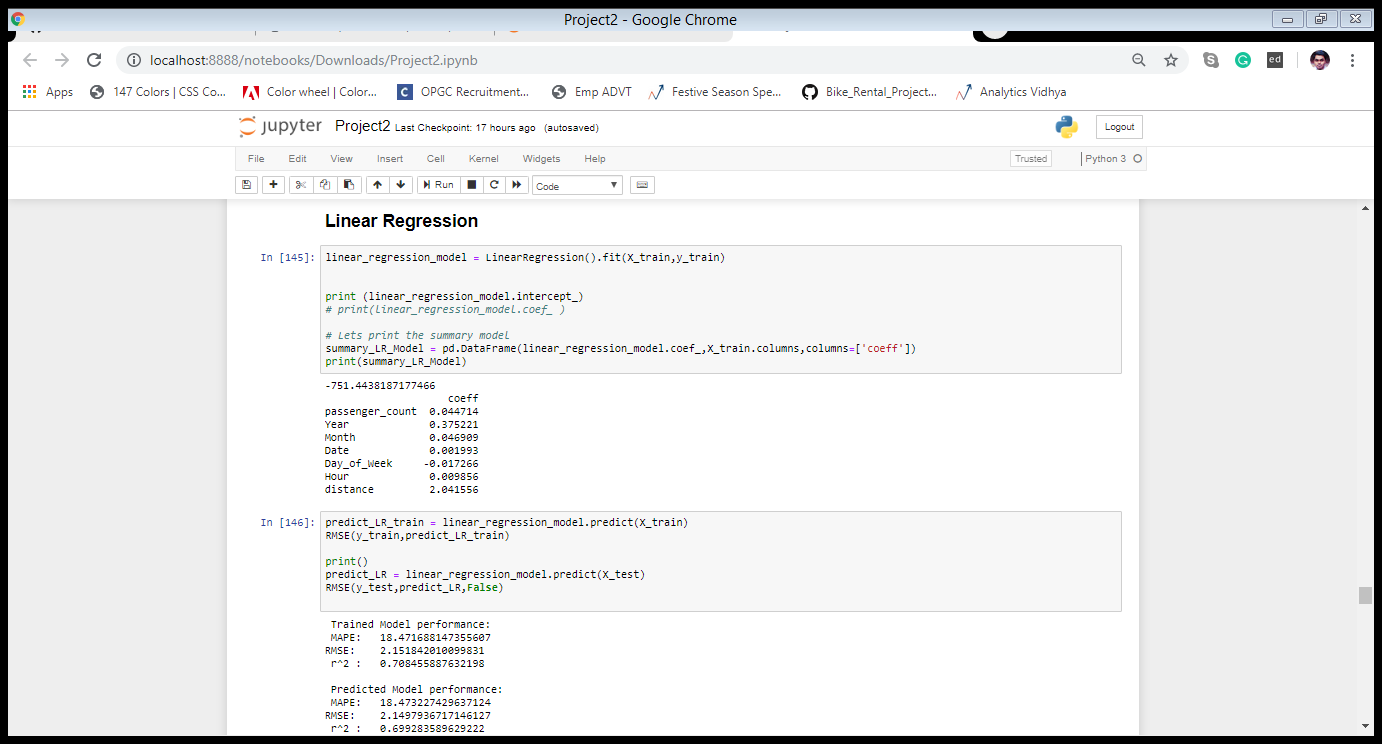
• Gradient Boosting

Before running any model, we will split our data into two parts which is train and test data. Here in our case we have taken 80% of the data as our train data. Below is the snipped image of the split of train test.



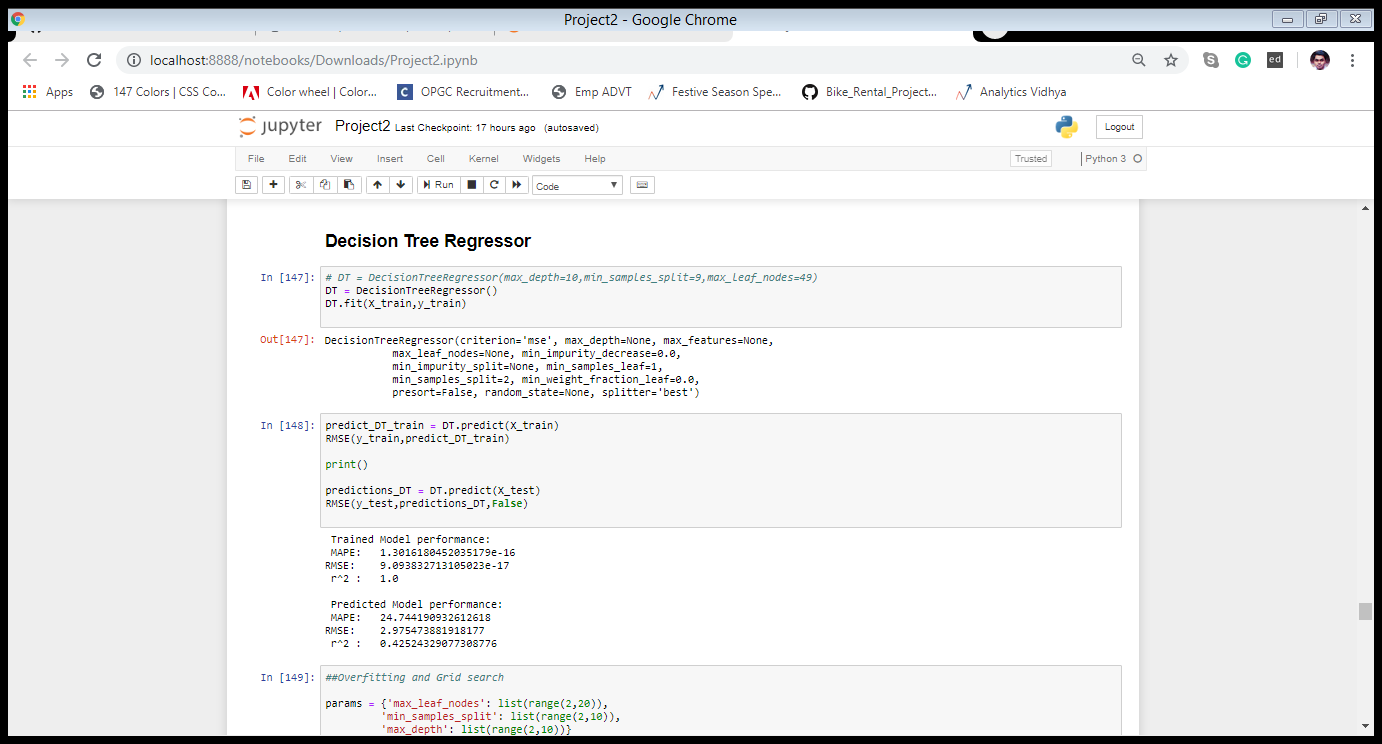
4.1 Linear Regression

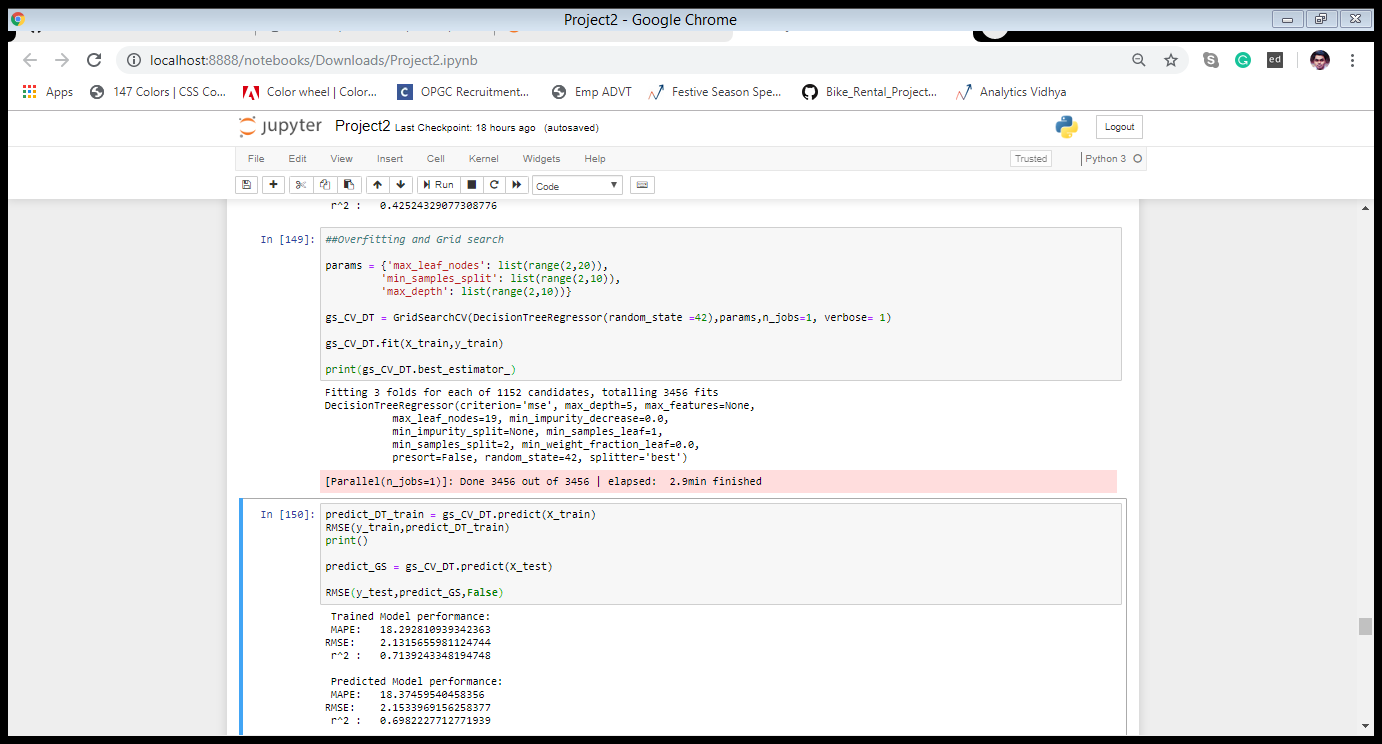
Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as a predictive analysis, when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable).

 Below is a screenshot of the model we build and its output:

4.2 Decision Tree

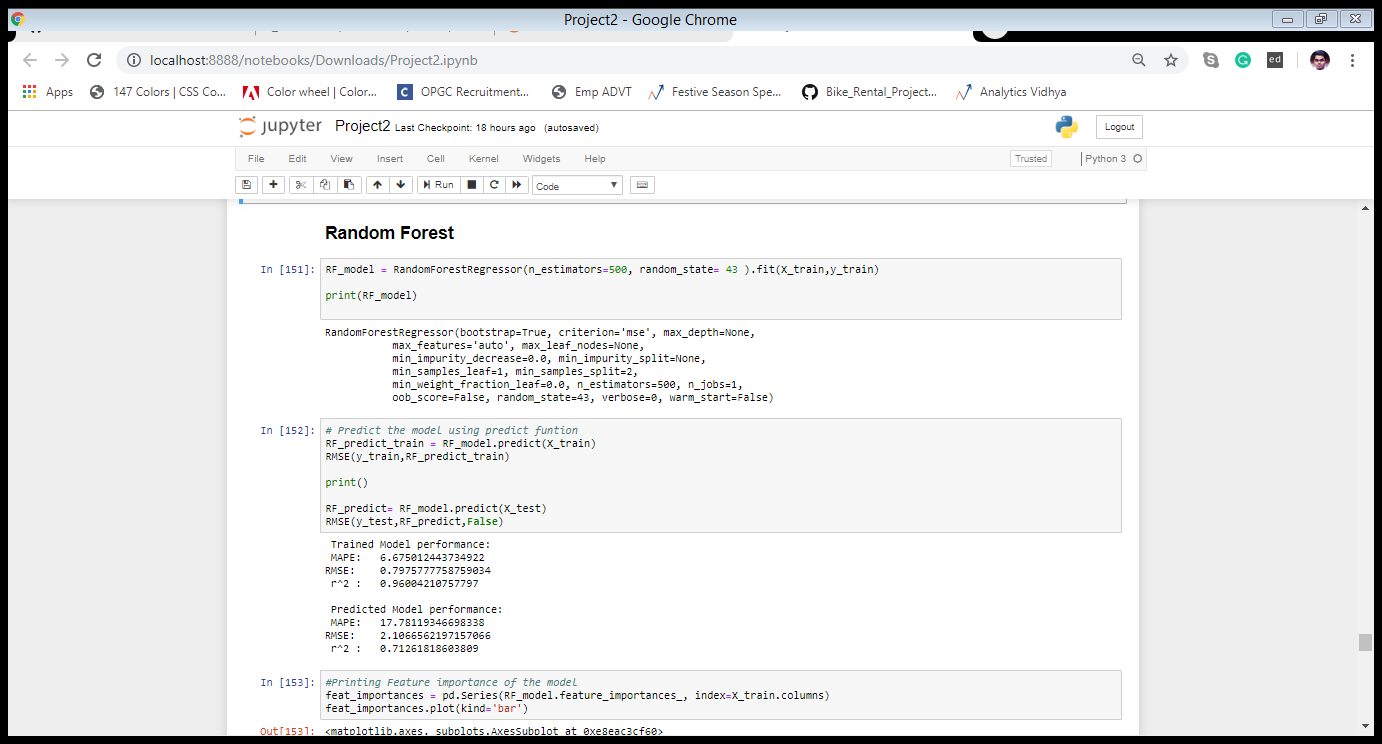
A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

Below is the screenshot of the query we executed and the result shown.

Decision Tree Algorithm Using GridSearchCV method to improve the performance of Decision Tree

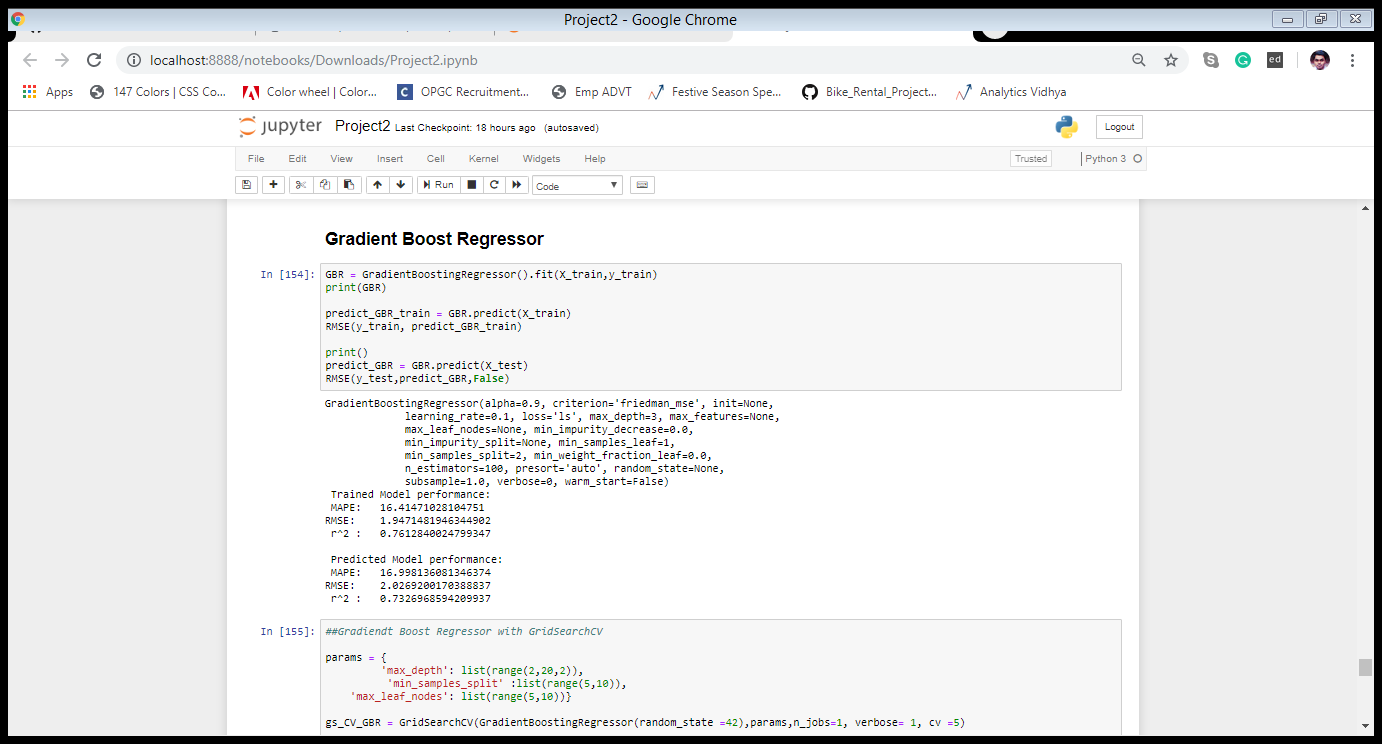
4.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other task, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Below is a screenshot of the model we build and its output:

4.4 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.

Below is a screenshot of the model we build and its output:

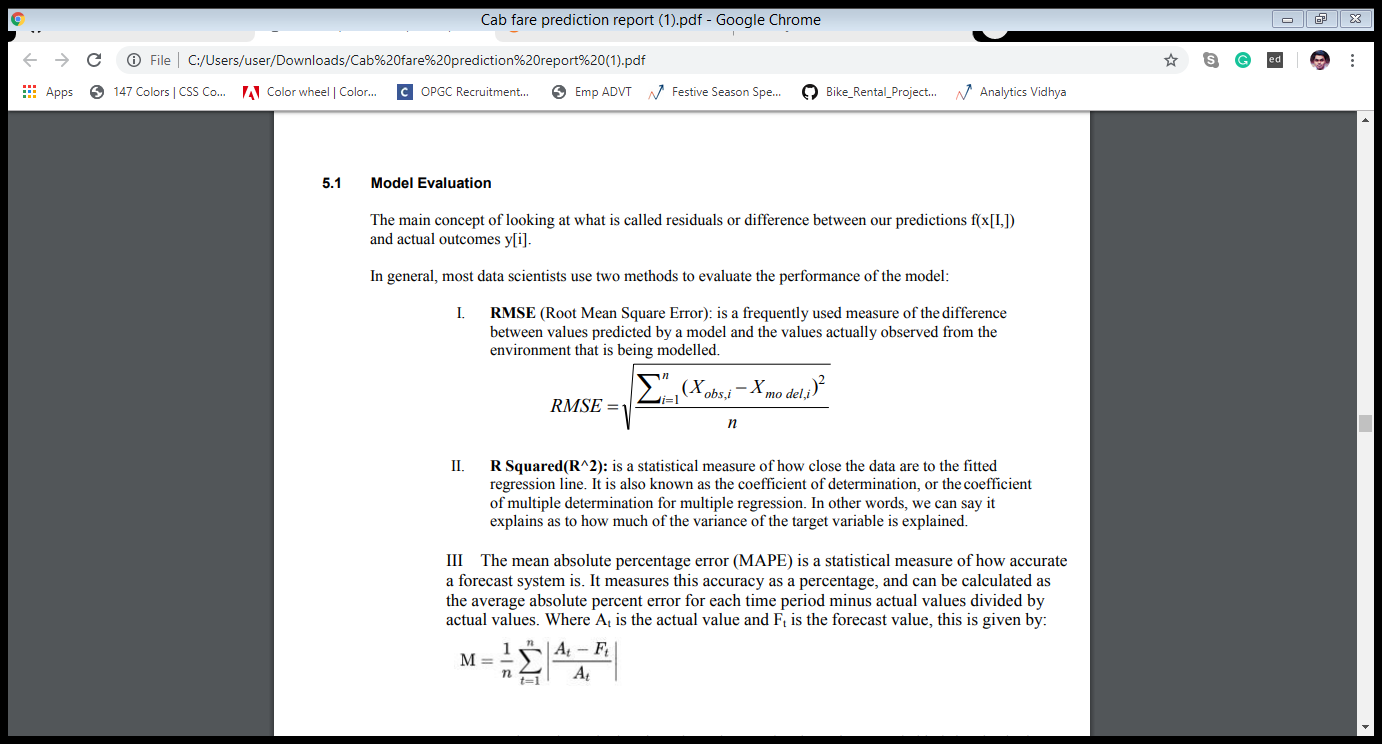
Chapter 5

Conclusion

5.1 Model Evaluation

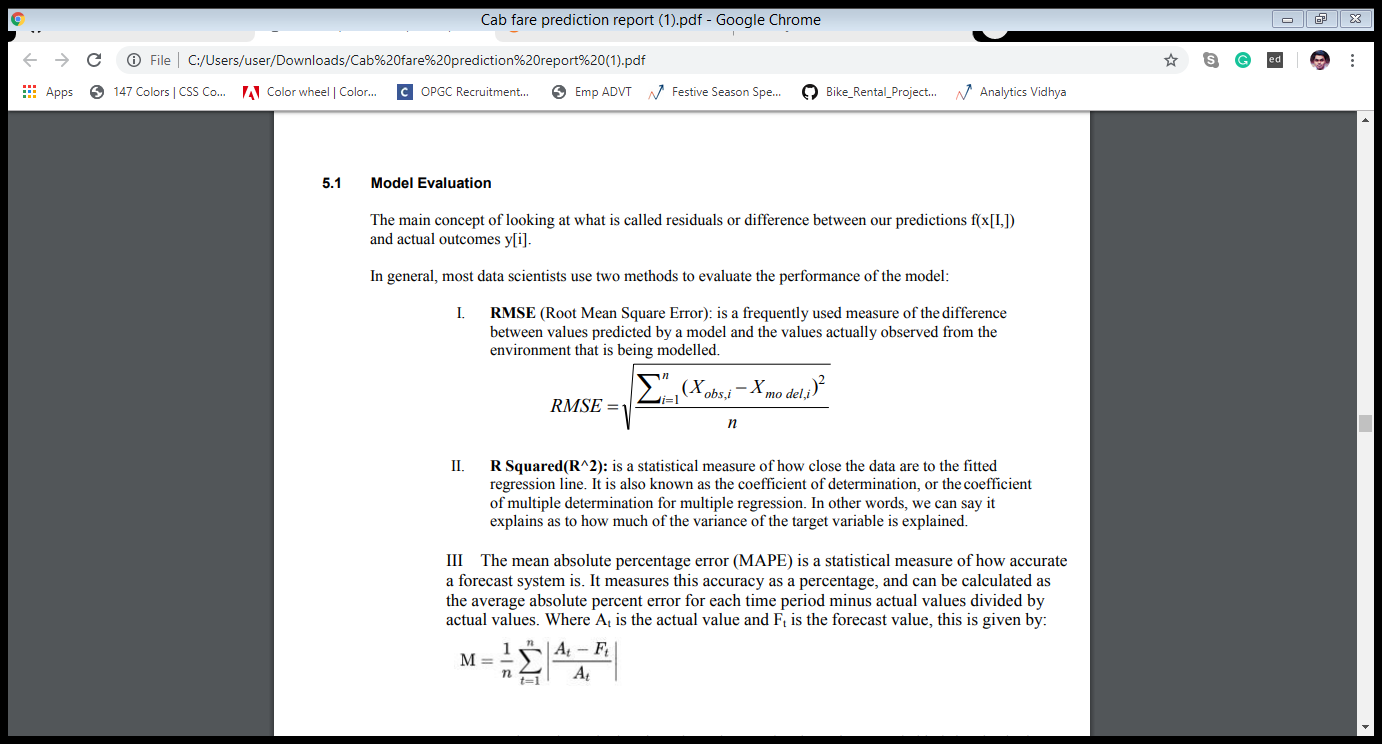
The main concept of looking at what is called residuals or difference between our predictions f(x[I,]) and actual outcomes y[i]. In general, most data scientists use two methods to evaluate the performance of the model:

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.

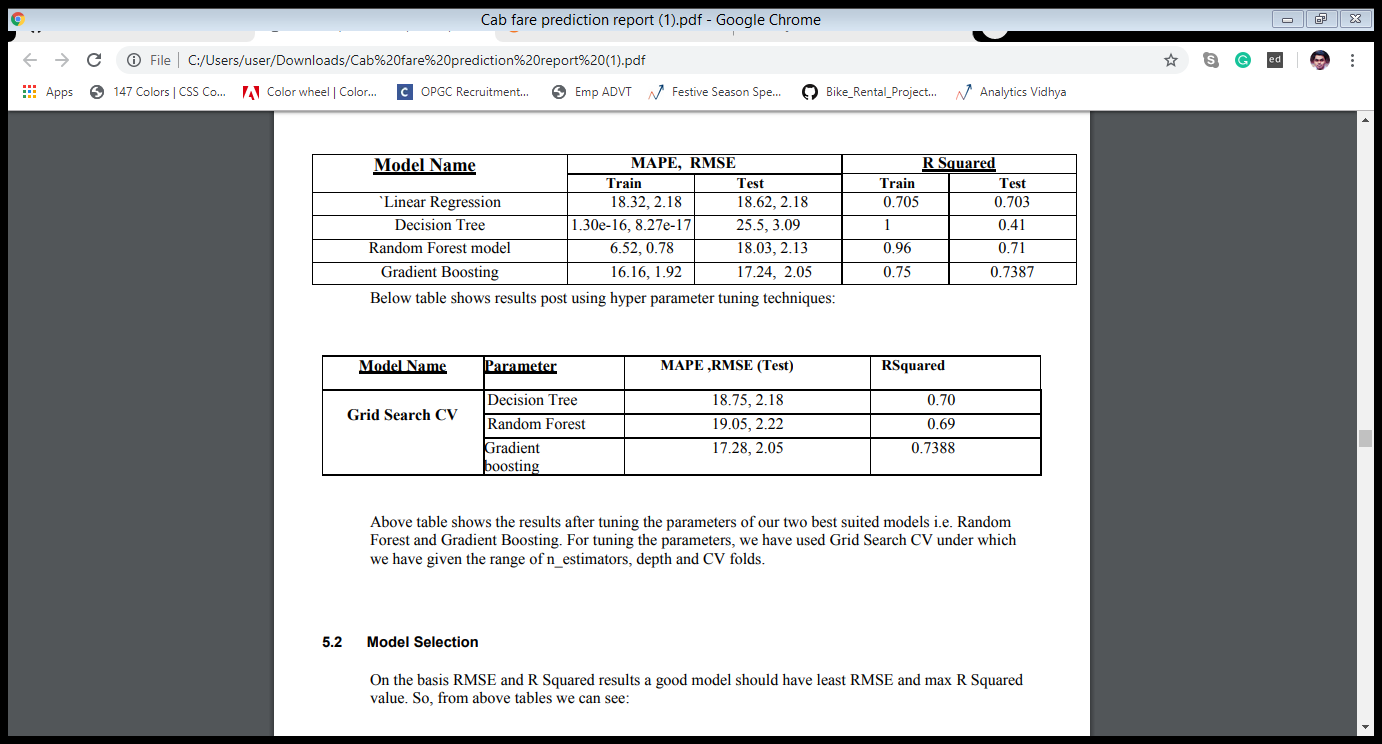


II. **R Squared(R^2)**: is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains as to how much of the variance of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

III The mean absolute percentage error (MAPE) is a statistical measure of how accurate a forecast system is. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values. Where At is the actual value and Ft is the forecast value, this is given by:



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

Below table shows the model results before applying hyper tuning

The previous table shows the results after tuning the parameters of our two best suited models i.e. Random Forest and Gradient Boosting. For tuning the parameters, we have used Grid Search CV under which we have given the range of n\_estimators, depth and CV folds.

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:

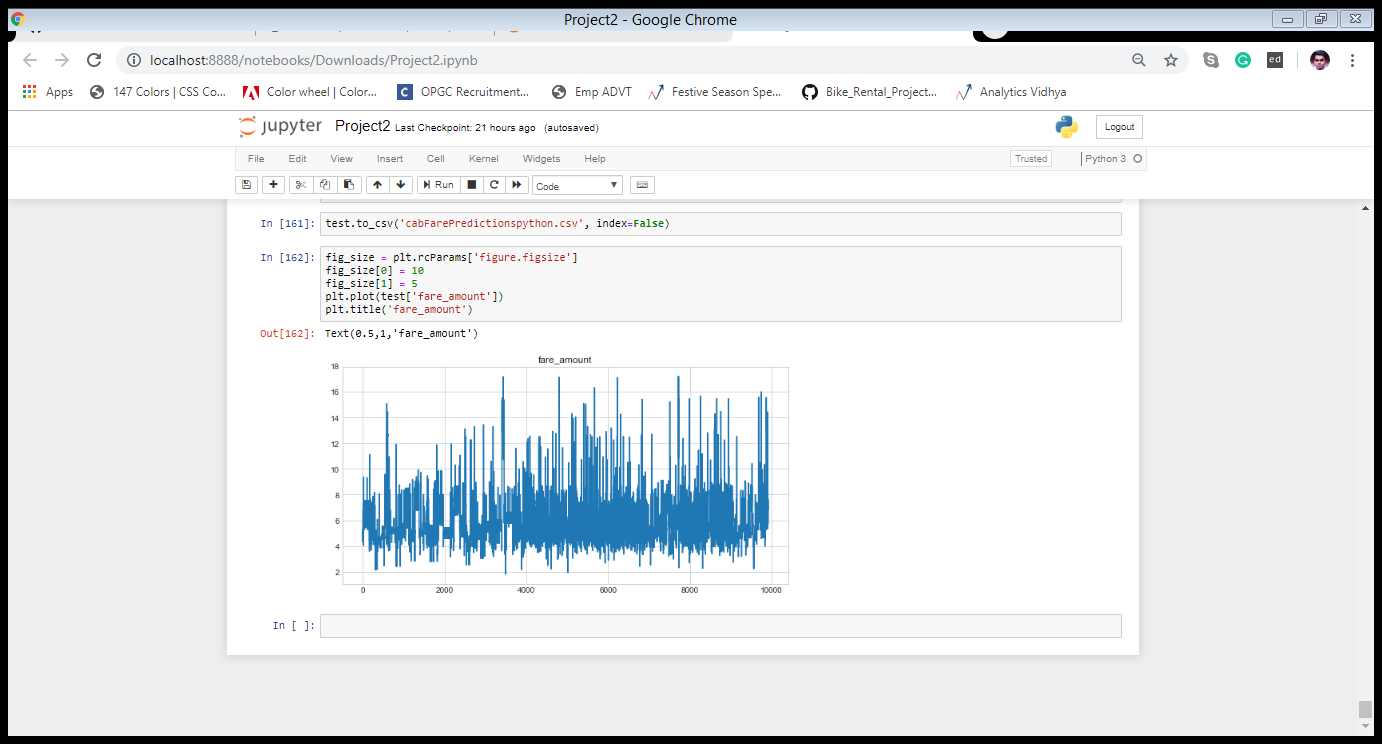
• From the observation of all RMSE Value and R-Squared Value we have concluded that,

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

• Compare to all other model, gradient boost model has 83.2% accuracy and RMSE is 2.02 and r^2 is 0.74. Hence, Gradient boost is best model for cab fare prediction

Finally, After analysing various models through different evaluation metrics, I conclude that applying the Gradient Boosting model gives us the best results in the trained dataset.

Thus applying the model in the test dataset helped me to predict the value of the target variable with utmost results. The test file with predicted output is shared along with the submission of all codes

Visualization expressing the predicted output of target variable of test dataset

Instructions to deploy and run the code.

• Save the python file as .py extension

• Open cmd prompt

• Before run the python file, make sure to set the path variable and install sklearn, numpy, mat plotlib, seaborn, pandas, fancy impute library in the system

• Type cd<path of the folder contains python file> and enter

EX: cd C:\Users\user\Downloads

• Python<name of the file with extension> and then enter

EX: python cab\_fare\_final.py

• The file will run in the cmd prompt

APPENDIX

R-CODE

rm(list = ls(all=TRUE)) #clearing the setup

setwd("C:/Users/user/Downloads")# setting the environment

getwd()

#load necessary libraries

x = c('ggplot2','corrgram','DMwR','caret','randomForest','unbalanced','C50','dummies','e1071','Information',

'MASS', 'rpart','gbm', 'ROSE','sampling', 'DataCombine', 'inTrees')

lapply(x, require, character.only = TRUE)

y = c("dplyr","plyr","ggplot2","data.table","GGally","magrittr","lubridate","tidyr","geosphere")

lapply(y, require, character.only = TRUE)

install.packages(c("dplyr","plyr","reshape","ggplot2","data.table","corrgram","geosphere","tidyr","scales"))

install.packages(c("GGally","DataCombine"))

library("dplyr")

library("plyr")

library("ggplot2")

library("data.table")

library("GGally")

library('geosphere')

library('tidyr')

library('scales')

library("caret")

###Load the csv file

train=read.csv("train\_cab.csv",header=T,na.strings = c(""," ","NA"))

# View(train)

test=read.csv("test.csv",header=T,na.strings = c(""," ","NA"))

# View(test)

#########################Explore the data###########################################

str(train)

str(test)

#Splitting the pickup\_date attribute into various entities like 'month', 'year', 'date', 'dayofweek', 'hour'

prepare\_datetime = function (x){

return(x %>%

mutate(

pickup\_datetime = ymd\_hms(pickup\_datetime),

month = month(pickup\_datetime),

year = year(pickup\_datetime),

day = day(pickup\_datetime),

dayOfWeek = wday(pickup\_datetime),

hour = hour(pickup\_datetime)

))

}

#Applyting the function into train & test dataset

train = prepare\_datetime(train)

test = prepare\_datetime(test)

###Fare amount can't be any value less than or equal to 0

#changing the datype of fare\_amount

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

#Function to remove values less than or equal to 0

filter\_fare\_amount = function(x){

x = x %>%

filter(fare\_amount>=1)

return(x)

}

#applying the function into train dataset

train=filter\_fare\_amount(train)

##Passenger Count can't be more than 6 and must not be in decimal numbers

##In given dataset, there are data with 0 passenger count thus remove it

#Function to apply the above stated

prepare\_passenger\_count = function(x){

x = x %>%

filter(passenger\_count>=1 , passenger\_count <=6 ) %>%

mutate( passenger\_count = floor(passenger\_count) )

# it removes na values as well

return(x)

}

#function applied on train & test dataset

train = prepare\_passenger\_count(train)

test = prepare\_passenger\_count(test)

#Checking if any longitude is < -180 or > 180

summary(train$pickup\_longitude)

summary(train$dropoff\_longitude)

#Checking if any latitude is < -90 or > 90

summary(train$pickup\_latitude)

summary(train$dropoff\_latitude)

#Function to remove values outside of prescribed values for latitute and longitutde

filter\_lat\_long = function(x){

x = x %>%

filter(pickup\_latitude<=90 , pickup\_latitude>=-90)

x = x %>%

filter(dropoff\_latitude<=90 , dropoff\_latitude>=-90)

x = x %>%

filter(pickup\_longitude<=180 , pickup\_longitude>=-180)

x = x %>%

filter(dropoff\_longitude<=180 , dropoff\_longitude>=-180)

return(x)

}

#Function applied on train & test datset

train = filter\_lat\_long(train)

test = filter\_lat\_long(test)

################################ Missing value analysis###############################

#To see if there are any missing values

Missing\_val = data.frame(sapply(train, function(x) sum(is.na(x))))

#changing the variable name of the dataframe

Missing\_val$Variables = row.names(Missing\_val)

row.names(Missing\_val) = NULL

names(Missing\_val)[1] = "Missing\_values"

#Show Percentage of missing value

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

train = na.omit(train)#removing the misisng values

######################## Calculating distance from pickup and drop coordinates###########

prepare\_distance = function(x){

x = x %>%

mutate(distance = by(x, 1:nrow(x), function(row) {

distHaversine(c(row$pickup\_longitude, row$pickup\_latitude), c(row$dropoff\_longitude,row$dropoff\_latitude))/1000}))

#Removing the distance which is 0

x= x[-which(x$distance == 0),]

return(x)

}

train = prepare\_distance(train)

test = prepare\_distance(test)

#Getting Summary of distance

summary(train$distance)

summary(test$distance)

#Removing the distance which are > 150

train = train[-which (train$distance > 150 ),]

#Removing the fare\_amount which is >1000

train = train[-which(train$fare\_amount > 1000),]

###########################Exploratory Data Analysis################################

###############################UNIVARIATE ANALYSIS###############################

#Function to visualize variables

Univariate\_distribtuion=function(value){

ggplot(train,aes\_string(x=value))+geom\_histogram(fill="DarkslateBlue", colour="Blue",bins=30)+

theme\_bw()

}

#Visualize the distribution of target variable "fare\_amount"

Univariate\_distribtuion(train$fare\_amount)

#Visualize the distribution of "passenger\_count"

Univariate\_distribtuion(train$passenger\_count)

#Visualize the distribution of "month"

Univariate\_distribtuion(train$month)

#Visualize the distribution of "year"

Univariate\_distribtuion(train$year)

#Visualize the distribution of "day"

Univariate\_distribtuion(train$day)

#Visualize the distribution of "dayofweek"

Univariate\_distribtuion(train$dayOfWeek)

#Visualize the distribution of "hour"

Univariate\_distribtuion(train$hour)

#Visualize the distribution of "distance"

Univariate\_distribtuion(train$distance)

##############################Bivariate Analysis##################################

#####################Distribution of passenger\_count over fare################

#PLoting the graph for passenger count and Fare

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

ggtitle("Time and Fare Plot") +

xlab("Passenger Count ") +

ylab("Fare")

#From the Graph, passenger count is not affecting the fare.

# we can see that single passengers are the most frequent travellers, and the highest fare also seems to come from cabs which carry just 1 passenger.

#####################Distribution of no of trips over the years################

train %>%

dplyr::count(year)

train %>%

dplyr::group\_by(year) %>%

dplyr::summarise(count = n() ) %>%

ggplot( aes(x = year, y = count, fill = year)) +

geom\_bar(stat="identity") +

theme(legend.position = "none") +

labs(title = "Distribution of no of trips over the years",

x = "passenger count",

y = "Total Count")

#From the graph,the no of trips over the year is uniform,maximum : 2012 and minimum : 2015

#####################Distribution of fare\_amount over the years##################

train %>%

dplyr::count(year)

train %>%

dplyr::group\_by(year)%>%

dplyr::summarise(fare\_amount = mean(fare\_amount) ) %>%

ggplot( aes(x = year, y =fare\_amount, fill = year)) +

geom\_bar(stat="identity") +

theme(legend.position = "none") +

labs(title = "Distribution of fare amount",

x = "Years",

y = "Fare amount")

#Avg Fare amount has beern increasing over the years.

#####################Distribution of fare\_amount over the months################

train %>%

dplyr::count(month)

train %>%

dplyr::group\_by(month)%>%

dplyr::summarise(fare\_amount = mean(fare\_amount) ) %>%

ggplot( aes(x = month, y =fare\_amount, fill = month)) +

geom\_bar(stat="identity") +

theme(legend.position = "none") +

labs(title = "Distribution of fare\_amount over the months",

x = "Months",

y = "fare amount")

#The fares throught the month mostly seem uniform, with the maximum fare received on the 10th

#####################Distribution of fare\_amount over hours##########################

#Hist of Hours

hist(as.numeric(train$hour), xlab = "Hours", main=paste("Hist of Hours"),col = "Green")

#from the graph,low frequency of the cab is found on morning hours

#PLoting the graph for Hours and Fare

ggplot(data=train, aes(x=train$hour, y=train$fare\_amount)) + geom\_point()+

ggtitle("Time and Fare Plot") +

xlab("Hours ") +

ylab("Fare")

#From the above graph we can see that the timeing is not affecting too much. Maximin dots are below 100.

#####################Distribution of fare\_amount over day of the week####################

hist(as.numeric(train$dayOfWeek), xlab = "DayOfWeek", main=paste("Hist of Dayofweek"),col = "Red")

#day of the week doesn't seem to have that much of an influence on the number of cab rides

#Plotting the graph for passenger Dayofweek and Fare

ggplot(data=train, aes(x=train$dayOfWeek, y=train$fare\_amount)) + geom\_point()+

ggtitle("Day count and Fare Plot") +

xlab("Day of Week ") +

ylab("Fare")

#The highest fares seem to be on a Sunday and Monday, and the lowest on Wednesday and Friday.

##############################Distribution of fare\_amount over distance################

ggplot(data=train, aes(x=train$distance, y=train$fare\_amount)) + geom\_point()+geom\_line()+

ggtitle("Distance and Fare Plot") +

xlab("Distance ") +

ylab("Fare")

# From the above graph, distance is found to be an important independent variable.

#################### Outlier Analysis ###############################################

### ## ## BoxPlots - Distribution and Outlier Check

numeric\_index = sapply(train,is.numeric) #selecting only integer

numeric\_data = train[,numeric\_index]

cnames = colnames(numeric\_data)

##function for plotting the boxplot

for( i in 1:length(cnames)) {

print(i)

assign(paste0('gn',i), ggplot( data = train,aes\_string( y= cnames[i], x = 'fare\_amount'),)+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="fare\_amount")+

ggtitle(paste("Box plot of cnt for",cnames[i])))

}

# # ## Plotting plots together

gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)

gridExtra::grid.arrange(gn3,gn4,gn5,ncol=3)

gridExtra::grid.arrange(gn6,gn7,gn8,ncol=3)

gridExtra::grid.arrange(gn9,gn10,gn11,ncol=3)

gridExtra::grid.arrange(gn12,ncol=1)

##Removing the outliers

cnames = c("distance", "fare\_amount")

for( i in cnames){

print(i)

val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]

train = train[ which(! train[,i] %in% val),]

}

##################Feature Selection################################################

## Correlation Plot

corrgram(train[,c('pickup\_longitude','pickup\_latitude','dropoff\_longitude','dropoff\_longitude','dropoff\_latitude','passenger\_count','month', 'year', 'day' ,'hour' ,'dayOfWeek', 'distance')],order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

# names(train)

## dimensionality reduction

##removing the redundant variables from dataset

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

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

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

#Normality check

#Train Data

cnames = c("distance")

for( i in cnames){

train[,i] = (train[,i] - min(train[,i])) / ( max(train[,i]) -min( train[,i]))

}

#Test data

for( i in cnames){

test[,i] = (test[,i] - min(test[,i])) / ( max(test[,i]) -min( test[,i]))

}

########################## Evaluation metrics#######################################

EVM = function( y\_actual, y\_predict) {

print("MAPE")

m = mean(abs( (y\_actual - y\_predict) / y\_actual) )

print(m )

difference = y\_actual - y\_predict

root\_mean\_square = sqrt(mean(difference^2))

print("RMSE")

print(root\_mean\_square)

print("R square value")

rss <- sum((y\_predict - y\_actual) ^ 2) ## residual sum of squares

tss <- sum((y\_actual - mean(y\_actual)) ^ 2) ## total sum of squares

rsq <- 1 - (rss/tss)

print(rsq)

}

#########################Model Development#######################################

#Clean the environment

rm(list=setdiff(ls(), c("train","test","EVM")))

set.seed(1234)

##Creating train and test data

train.index = createDataPartition(train$fare\_amount,p= 0.8,list = F)

train\_data = train[train.index,1:8]

test\_data = train[c(-train.index),1:8]

################################## LINEAR REGRESSION #########################################

linerModel <- lm(fare\_amount ~., data = train\_data)

summary(linerModel)

linear\_predict <- predict(linerModel,test\_data)

EVM(test\_data[,1],linear\_predict)

# "MAPE" =0.18024

# "RMSE" =1.8451

# R-squared = 0.64370

#################### DECISION TREE MODEL #########################################

DT = rpart(fare\_amount ~ ., data = train\_data, method = 'anova')

predictions\_DT = predict( DT, test\_data[,-1])

print(DT)

EVM(test\_data[,1], predictions\_DT)

# "MAPE" = 0.197386

# "RMSE" = 1.968644

# R-squared = 0.5935116

######################## RANDOM FOREST #########################################

RF = randomForest(fare\_amount ~ .,train\_data , importance = TRUE, ntree = 500)

predictions\_RF = predict( RF, test\_data[,-1])

# plot(RF)

EVM(test\_data[,1], predictions\_RF)

# "MAPE" = 0.17633

# "RMSE" = 1.76341

# R-squared = 0.6744

########################## GRADIENT BOOST#######################################

model\_gbm <- caret::train(fare\_amount ~ .,

data = train\_data,

method = "gbm",

distribution="gaussian",

trControl = trainControl(method = "repeatedcv",

number = 5,

repeats = 3,

verboseIter = FALSE),

verbose = 0)

data\_gbm = predict(model\_gbm, test\_data)

EVM(test\_data[,1], data\_gbm)

# "MAPE" = 0.1667488

# "RMSE" = 1.73542

# R-squared = 0.68719

## out of all the models, Gradient boost Model proves out to be the finest among all

## and is applied on the test data

test$fare\_amount <- predict(model\_gbm,test)

#graph between fare\_amount and distance of test data

gplot <- ggplot(data=test, aes(x=distance, y=fare\_amount)) + geom\_point()+ geom\_line()+

ggtitle("Distance and Fare Plot") +

xlab("Distance ") +

ylab("Fare")

gplot

##Test data file with predicted fare\_amount

write.csv(test, file="cab\_fare\_predictionR.csv", row.names = FALSE)