***Cab Fare Amount***

Aishwarya Kapoor

17th April 2019

**Content**

Section 1 ………………………………………………………………………………………. 3

Problem Statement

Section 2 ………………………………………………………………………………………. 5

Pre-processing of Data

Section 3 ………………………………………………………………………………………. 13

Modelling

Section 4 ………………………………………………………………………………………. 14

R-codes

Appendix ………………………………………………………………………………………. 21

**Project Name – Cab Fare Prediction**

***Section 1***

*Problem Statement*

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

*Data Type*

We have a continuous data type. We have 6 variables in which one of them is “depended variable” and rest 5 are independent variables. Among that 5 independent variable there is one date variable.

**Number of attributes:**

* pickup\_datetime - timestamp value indicating when the cab ride started.
* pickup\_longitude - float for longitude coordinate of where the cab ride started.
* pickup\_latitude - float for latitude coordinate of where the cab ride started.
* dropoff\_longitude - float for longitude coordinate of where the cab ride ended.
* dropoff\_latitude - float for latitude coordinate of where the cab ride ended.
* passenger\_count - an integer indicating the number of passengers in the cab ride.

**Depended Variable**

Fare\_amount – tell us the amount of each ride.

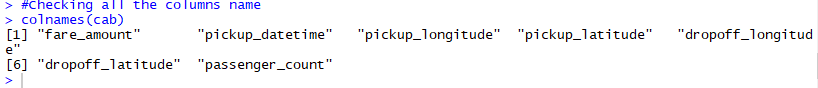


Image 1 – All the names of the variables present in our data

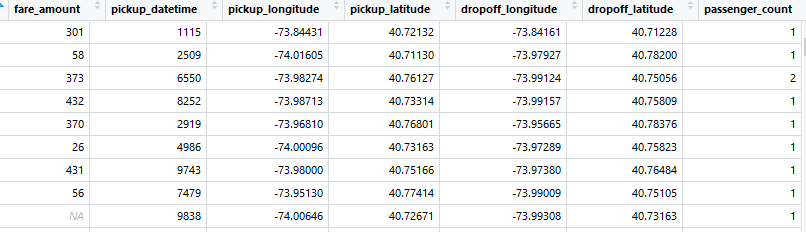


Image 2 – It the head of the data

Some central tendency of data to us about the data like – mean, medium, etc. about each variable.

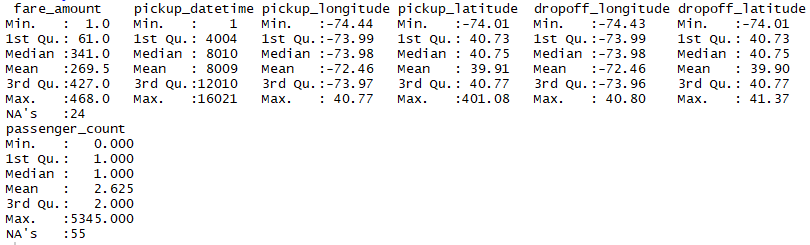


Image 3 – Central tendency of each variable

***Section 2***

*Pre-processing of the data*

As, we have seen in the above part our data is not normally distribute. In our data we have our target variable – fare\_amount, and date is one of the variable and other 4 independent variables.

We don’t need the “Date” variable as we are not analyzing on the time or date. That is why it is better to remove this variable so that it will us to reduce the complexity of model.

For pre-processing there are many steps which we are require to do in which very first step is “Removing NA’s” .

Step 1:

**Removing NA’s**

First we will check whether NA’s are present in our data or not. For which in R there is a very simple code(image is provided below for the reference)



Image 4 – Missing value count

So, we can see missing value is present in our data. Now, we need to know in which variable it is present and what is the percentage of missing values.

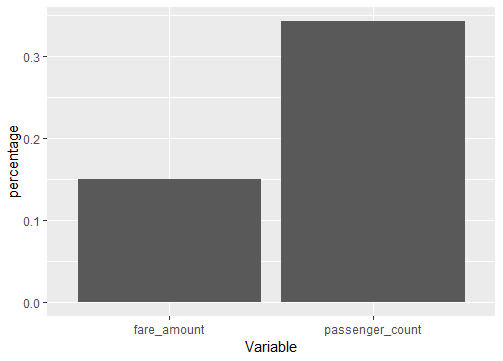


Image 5- Missing value Bar graph

Missing value is present in 2 variables – “passenger\_count” and “fare\_amount”.

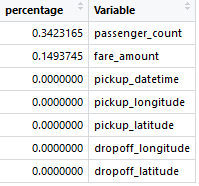


Image 6 – Missing value Table

Now, we will find the missing value for “fare\_amount” and “passanger\_count” variables.

We have 3 methods to impute the missing value in the data:

1. Mean method
2. Median method
3. Knn- Imputation

We used all the 3 methods in the end the method which we are using for our model is “Median” because it have the nearest value.

*Step 2:*

Detecting the **outliers** in the data

To detect the outlier “Boxplot” is one of the best method to use it will visualize the data and while seeing only we can identify the outliers in the data and Boxplot works on median method.

We are going to plot for each variable to check the outliers.

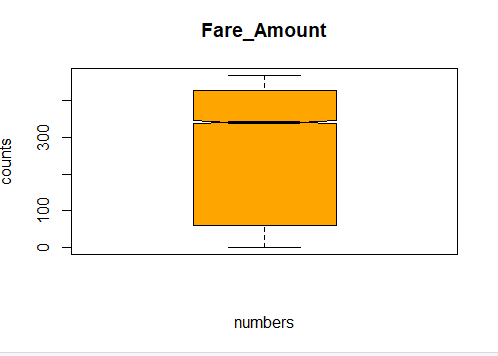


Image 7 – Boxplot for fare\_amount

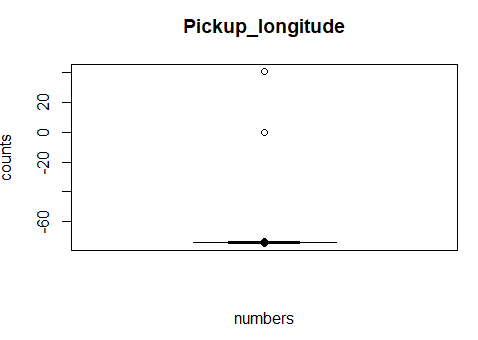


Image 8 – Boxplot for Pickup\_longitude

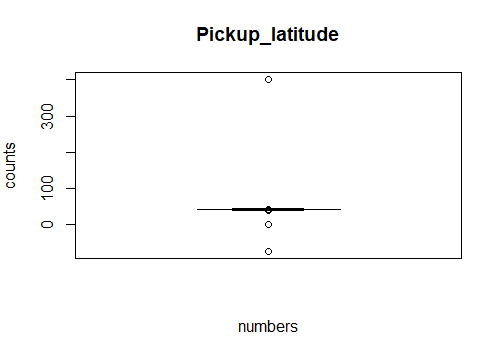


Image 9 – Boxplot for Pickup\_latitude

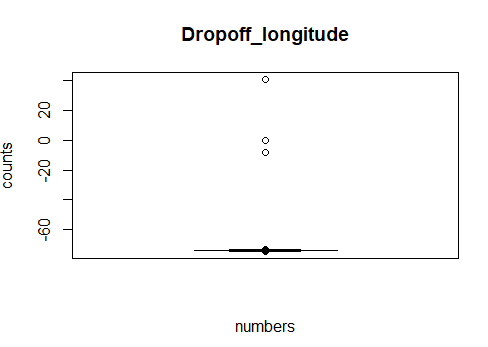


Image 10 – Boxplot for Dropoff\_latitude

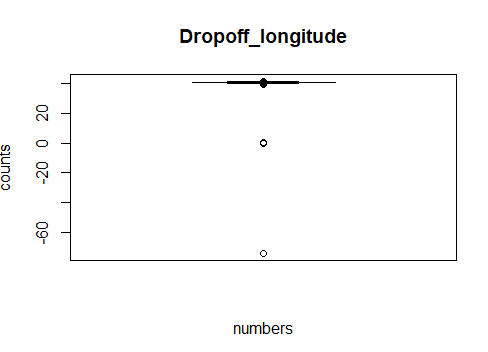


Image 11 – Boxplot for Dropoff\_longitude

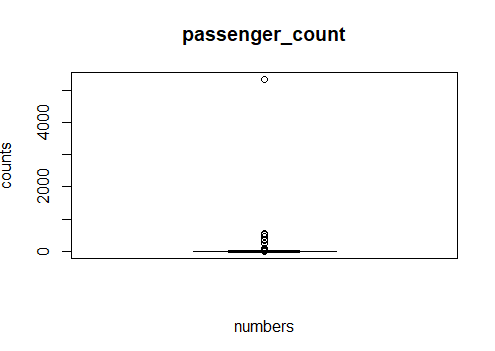


Image 12 – Boxplot for Passenger\_count

As, there are very few numbers of the outliers in the data we have decided to remove the outliers from the data.

*Step 3:*

**Feature selection**

Feature selection is nothing but to select the all independent variables which are required to build to model.

We do the correlation test in the data to do so we have plot correlation graph.

Tukey method is based in the boxplot method. In this it will consider all the values above the 3rd quartile and all the values below the 1st quartile is consider as the outliers it will use the following formula:

* Outlier on the upper side – 3rd quartile +1.5\*IQR
* Outlier on the lower side – 1st quartile -1.5\*IQR

Here, IQR stands for the interquartile range.

By using this

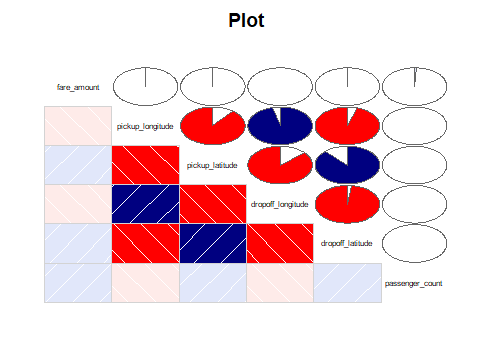


Image 13- Corrgram

In this graph it “Red color” define the negative correlation between 2 variables and “Dark blue” shows the positive correlation between 2 variables.

In this

* pickup\_longitutude is highly positively correlated to dropoff\_longitude
* Pickup\_latitude is highly positively correlated to dropoff\_latitude
* Pickup\_longitude is highly negatively correlated to pickup\_latitude
* Dropoff\_latitude is highly negatively correlated to dropoff\_longitude.

We have decided to take only 2 variables from this data that are :

* Pickup\_longitude
* Dropoff\_latitude

Now, we have 3 variables in our data

* Fare\_amount – which is our target variable
* Pickup\_latitude – which tell us where the cab ride started
* Dropoff\_longitude – which tell us where the cab ride ended

*Step 4:*

**Feature scaling**

We are using the normalization to scale the data because our data is not normally distributed.

We scale our data because it helps to normalize the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.

After this process our data will look this:

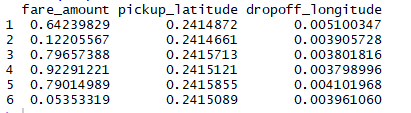


Image 14 – Data after scaling

***Section 3***

*Modelling*

We are using Machine learning algorithms are procedures to automatically generalize from historical observations

We have continuous data for which we need to run the linear regression algorithms. We are using the supervised machine learning.

We have selected 2 supervised machine learning for our project and those are:

* Decision Tree
* Linear Regression

First we will do “Decision Tree”. The question which come up is why we select this one because it

fit a regression model to the target variable using each of the independent variables. Then for each independent variable, the data is split at several split points. At each split point, the "error" between the predicted value and the actual values is squared to get a "Sum of Squared Errors (SSE)". The split point errors across the variables are compared and the variable/point yielding the lowest SSE is chosen as the root node/split point. This process is recursively continued.

Before running the model we need to take the train data and test data for our model. After taking that we will run the Decision tree model.

Our Second Model is Linear Regression Model why we have selected this model because Linear Regression is a machine learning algorithm based on supervised learning. It performs aregression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting.

Bases on our Error matrix we would like to freeze the “Linear regression model”

The reason being is MAPE (mean absolute percentage error) is low as compare to decision tree.

***Section 4***

*R codes*

#setting the working directory

setwd("C:/Users/Lenovo/Desktop/Edwisor/Project 2")

#Loading the library

library(DMwR)

library(rpart)

install.packages("ggplot")

library(ggplot2)

library(corrgram)

install.packages("ggpubr")

library(ggpubr)

install.packages("devtools")

install.packages("dpylr")

install.packages("C50")

library(C50)

library(MASS)

#reading the data before the clearing the R enviorment

rm(list = ls(all=T))

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

read.csv("test.csv", header = T, na.strings = c(" ", "", "NA")) -> cab\_pre

#converting into Numeric

cab[] <- lapply(cab[], as.numeric)

cab\_pre[] <- lapply(cab\_pre[], as.numeric)

#creating some basic graphs

qqplot(cab)

hist(cab$fare\_amount, col = "green")

hist(cab$pickup\_longitude, col = "blue")

hist(cab$pickup\_latitude, col = "red")

hist(cab$dropoff\_longitude, col = "orange")

hist(cab$dropoff\_latitude, col = "yellow")

hist(cab$passenger\_count, col = "grey")

#Checking all the columns name

colnames(cab)

View(cab)

#Summary of the data

summary(cab)

#checking the count of all the na's in data

#creating dataframe only for missing values

sum(is.na(cab))

names(cab)

miss\_values = data.frame(apply(cab, 2, function(x) {sum(is.na(x))}))

data.frame(miss\_values)

class(miss\_values)

miss\_values$columns = row.names(miss\_values)

row.names(miss\_values) = NULL

names(miss\_values)[1] = "percentage"

names(miss\_values)[2] = "Variable"

miss\_values$percentage = (miss\_values$percentage/ nrow(cab))\*100

miss\_values = miss\_values[order(-miss\_values$percentage),]

View(miss\_values)

miss\_values[1:2,] -> miss

miss

#Plotting the bar graph for the Missing variables

ggplot(miss, aes(x = Variable, y = percentage)) + geom\_col()

+ xlab = ("variable") + ylab = ("Percentage") + main = ("Missing\_data") + col = ("blue")

#dropping the Date variable as well

cab <- cab[,-2, drop = FALSE]

head(cab)

#treating the NA

#lets create the missing value for one point

cab[6,1]

cab[6,1] = NA

cab[6,6]

cab[6,6] = NA

#Actual Value = 26 / 1

#Mean method value = 269.4907 / 2.625

#Median method value = 341 /1

#knn IMpoutation value = 393.625 / 1.0000

# 1st will use the central tendancy in which we will use "MEAN" because it have only numeric data

cab$fare\_amount[is.na(cab$fare\_amount)] = mean(cab$fare\_amount, na.rm = T)

cab$passenger\_count[is.na(cab$passenger\_count)] = mean(cab$passenger\_count, na.rm = T)

#Median method

cab$fare\_amount[is.na(cab$fare\_amount)] = median(cab$fare\_amount, na.rm = T)

cab$passenger\_count[is.na(cab$passenger\_count)] = median(cab$passenger\_count, na.rm = T)

#Knn - imputaation

cab$fare\_amount =knnImputation(cab, k= 5)

cab$passenger\_count =knnImputation(cab, k= 5)

#We are going to use the mean method for the missing value imputation

#Outliers

#we use using the univariate method to check the outliers in the following variables

boxplot(cab$fare\_amount, main = "Fare\_Amount", xlab = "numbers", ylab = "counts", col = "orange", horizontal = FALSE, notch = TRUE)

boxplot(cab$pickup\_longitude, main = "Pickup\_longitude", xlab = "numbers", ylab = "counts", col = "orange", horizontal = FALSE, notch = TRUE)

boxplot(cab$pickup\_latitude, main = "Pickup\_latitude", xlab = "numbers", ylab = "counts", col = "orange", horizontal = FALSE, notch = TRUE)

boxplot(cab$dropoff\_longitude, main = "Dropoff\_longitude", xlab = "numbers", ylab = "counts", col = "orange", horizontal = FALSE, notch = TRUE)

boxplot(cab$dropoff\_latitude, main = "Dropoff\_longitude", xlab = "numbers", ylab = "counts", col = "orange", horizontal = FALSE, notch = TRUE)

boxplot(cab$passenger\_count, main = "passenger\_count", xlab = "numbers", ylab = "counts", col = "pink", horizontal = FALSE, notch = TRUE)

#decting the outlires

outlier1 = cab$pickup\_longitude[cab$pickup\_longitude %in% boxplot.stats(cab$previous) $out]

outlier1

outlier2 = cab$pickup\_latitude[cab$pickup\_latitude %in% boxplot.stats(cab$previous) $out]

outlier2

outlier3 = cab$dropoff\_longitude[cab$dropoff\_longitude %in% boxplot.stats(cab$previous) $out]

outlier3

outlier4 = cab$dropoff\_latitude[cab$dropoff\_latitude %in% boxplot.stats(cab$previous) $out]

outlier4

outlier5 = cab$passenger\_count[cab$passenger\_count %in% boxplot.stats(cab$previous) $out]

outlier5

#removing the outlier from the data

cab = cab[which(!cab$pickup\_longitude %in% outlier1), ]

cab = cab[which(!cab$pickup\_latitude %in% outlier2), ]

cab = cab[which(!cab$dropoff\_longitude %in% outlier3), ]

cab = cab[which(!cab$dropoff\_latitude %in% outlier4), ]

cab = cab[which(!cab$passenger\_count %in% outlier5), ]

# As, outliers are dected as NULL

#Feature selection

#we are checking the correlation between the varaibles

library(corrgram)

corrgram(cab, order = F, upper.panel = panel.pie, text.panel = panel.txt, main = "Plot")

#using this plot wee can study that pickup\_longitude is positively correlated

#with dropoff\_longitude and negatively correlated with pickup\_latitude and also

#pickup\_latitude is negatively correlated witb dropoff\_longitude and positively correlated with

#dropoff\_latitude. so we can drop the pickup\_latitude and pickup\_longitude variables.

#dropping off the variables

cab = subset(cab, select = -c(pickup\_longitude, dropoff\_latitude, passenger\_count))

head(cab)

#changing dropoff to latitude

#Scatter plot

ggqqplot(cab$fare\_amount, ylab = "FARE")

ggqqplot(cab$pickup\_longitude, ylab = "pickup\_longitude")

ggqqplot(cab$pickup\_latitude, ylab = "dropoff\_latitude")

ggqqplot(cab$dropoff\_longitude, ylab = "dropoff\_longitude")

ggqqplot(cab$dropoff\_latitude, ylab = "dropoff\_latitude")

#Feature Scaling

#Normalization and standardization

#checcking the normality of the data

qqnorm(cab$dropoff\_longitude)

hist(cab$fare\_amount)

hist(cab$dropoff\_latitude)

hist(cab$pickup\_latitude)

#this data is not normally distributed, we will use Normalization to scale the data

#normalization method

cnames = c("fare\_amount", "pickup\_latitude", "dropoff\_longitude")

for (i in cnames) {

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

}

head(cab)

#Building model

#Decision Tree

#creating smaple for the data using simple sample method because all variables are continous

#we are using only Train data for our testing of data will use Test data later

#train\_index = sample(1:nrow(cab), 0.8\*nrow(cab))

#train\_data = cab[train\_index,]

#test\_data = cab[-train\_index,]

train\_data = cab

test\_data = cab\_pre

#Decision Tree

library(rpart)

fit = rpart(fare\_amount ~., data = train\_data, method = "anova")

fit

predictons = predict(fit, test\_data)

predictons

#Error matrix

#MAPE amount of error we get

mape = function(y, yhat) {

mean(abs((y - yhat)/y))\*100

}

mape(test\_data[,1], predictons)

regr.eval(test\_data[,1], predictons, stats = c("mae", "mape", "rmse", "mse"))

#MAPE for Decision tree is 99.80%

#MAPE for liner model is 98.19%

#Liner regression model

install.packages("usdm")

library(usdm)

vif(cab[,-1])

indep\_vars = c("pickup\_latitude", "dropoff\_longitude")

vifcor(subset(cab, select = (indep\_vars)), th=0.9)

lm\_model = lm(fare\_amount ~., data = train\_data)

summary(lm\_model)

prediction\_lm\_model = predict(lm\_model, test\_data)

mape(test\_data[,1], prediction\_lm\_model)

regr.eval(test\_data[,1], predictons, stats = c("mae", "mape", "mae", "mse"))

***Appendix***

*Libraries and packages which are used*:

* install.packages("usdm")
* library(usdm)
* library(DMwR)
* library(rpart)
* install.packages("ggplot")
* library(ggplot2)
* library(corrgram)
* install.packages("ggpubr")
* library(ggpubr)
* install.packages("devtools")
* install.packages("dpylr")
* install.packages("C50")
* library(C50)
* library(MASS)

*Histogram*

