**Cab Fare Prediction**

By Gunjan Panda

21st June, 2019

**Contents**

**1. Introduction**  3

1.1 Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

1.2 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4

**2. Methodology**  5

2.1 Pre Processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.1.1 Missing Value Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.1.2 Outlier Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

2.1.3 Feature Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

2.1.4 Feature Engineering . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.2 Modeling . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.2.1 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

2.2.2 Multiple Linear Regression . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12

2.2.3 Regression Trees . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13

2.2.4 K Nearest Neighbour . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14

**3. Conclusion**  16

3.1 Model Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.1.1 R Square . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.1.2 Root Mean Squared Error (RMSE) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16

3.2 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

16. Appendix A - Extra Figures 18

17. Appendix B – R code 22

18. References 32

1. Introduction

1.1 Problem Statement

For a cab rental start-up company, the fare amount is dependent on a lot of factors. The aim of this project is to understand all patterns and to apply analytics for fare prediction. We need to design a system that predicts the fare amount for a cab ride in the city.

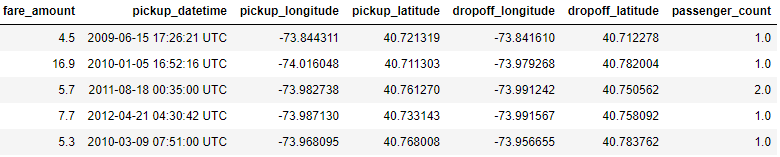
1.2 Data

Our task is to build regression models which will predict the continuous fare amount for each cab ride and help up predict depending on multiple time-based, positional and generic factors.

According to CRISP DM Process , this problem statement lies in the category of forecasting which deals with predicting continuous values for future(in our case the continuous value is the fare amount of the cab ride.) This is Time Series Forecasting as it deals with the timestamp variable and on the basis of this and other variables we are going to predict the future cab fare amount based on different times and different locations.

Given below is a sample of the data set that we are using to predict the fare amount of a cab ride:

Figure 1.1



As you can see, there are 6 predictor variables(i.e 6 independent variables) and 1 target variable(i.e dependent variable).

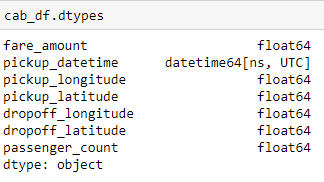
Predictors

1. Pickup\_datetime : timestamp value indicating when the cab ride started.
2. Pickup\_longitude: float for longitude coordinate of where the cab ride started.
3. Pickup\_latitude: float for latitude coordinate of where the cab ride started.
4. Dropoff\_longitude: float for longitude coordinate of where the cab ride ended.
5. Dropoff\_latitude: float for latitude coordinate of where the cab ride ended.
6. Passenger\_count: an integer indicating the number of passengers in the cab ride.

Target : fare\_amount

Data structure after proper data type conversion shown in figure 1.2 :

Figure 1.2



Methodology

2.1 Pre Processing

To build any model, the first step is to understand the problem statement and choose the appropriate category that fits in. Since our statement fits into forecasting we use Regression as it helps us in predicting continuous fare amount for the future. Regression is a Supervised Learning approach as we know the target variable beforehand which is fare amount.

After identifying the approach, the next step is preprocessing the data. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. Refer to Appendix A to see the plots.

To start this process we will first try and look at some of the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable. In figure 2.1.1 and figure 2.1.2 we have plotted the probability density functions of few variables available in the data as well as the dependent fare\_amount variable. The distributions clearly depict how skewed the datapoints are indicating the presence of outliers.

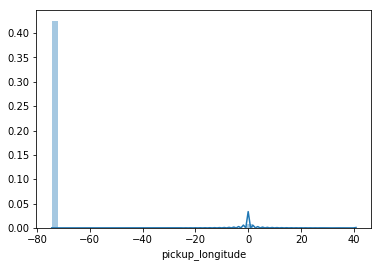


Figure 2.1.1

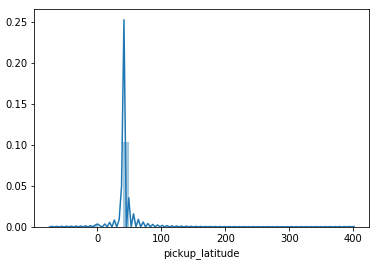


Figure 2.1.2

2.1.1 Missing Value Analysis

Once proper data conversion is done next step is to analyze the missing values. According to our data, the missing value percentage of the variables are as follows 🡪

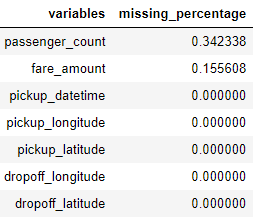


Figure 2.1.1.1

Since only passenger\_count and fare\_amount have missing values and the percentage is less than 30% so we need to impute the missing values.

By randomly assigning NA to one of the values for passenger\_count and then filling the value using 3 methods : mean,median and KNN we found out that median gives the closest value to the actual value.

Similarly for fare\_amount , mean gives the closest value to the actual value so the missing values for passenger\_count are filled with median and for fare\_amount are filled with mean.

After filling in missing values our data looks like shown in figure 2.1.1.2

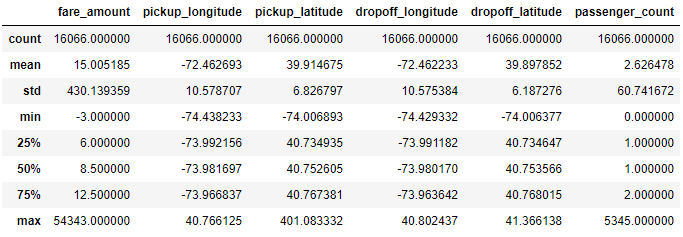


Figure 2.1.1.2

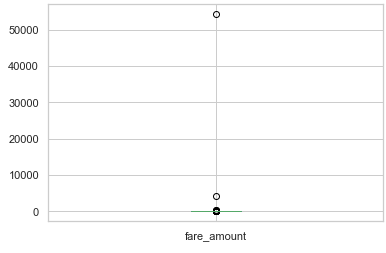
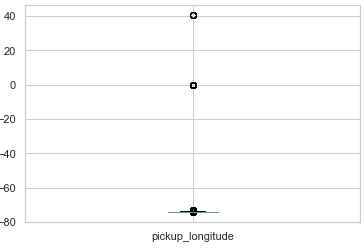
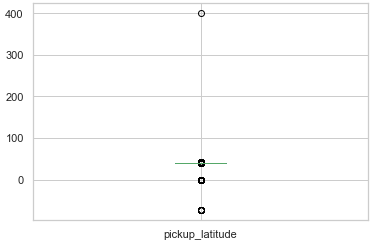
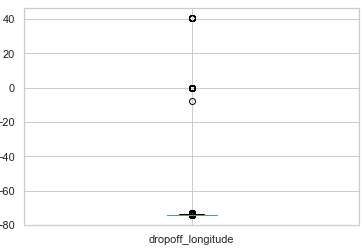
2.1.2 Outlier Analysis

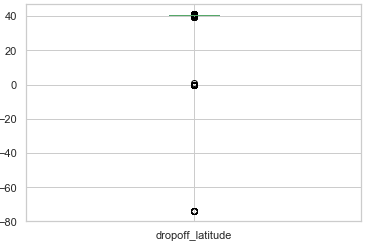
As we can see in the figure above there is a lot of noisy data so its important to clean the data for a better model performance.

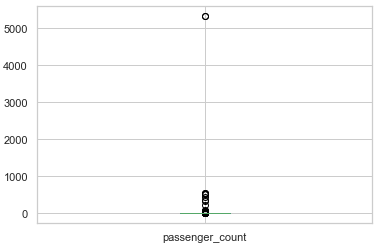
In this case we use a classic approach of removing outliers, Turkey’s method. We visualize the outliers using boxplots.

In figure 2.1.2.1 we have plotted the boxplots of the 6 predictor variables and the target variable. A lot of useful inferences can be made from these plots. We have a lot of outliers and extreme values in each of the data set.

Figure 2.1.2.1





After removing the outliers, our data is now clean and looks like shown in figure 2.1.2.2.

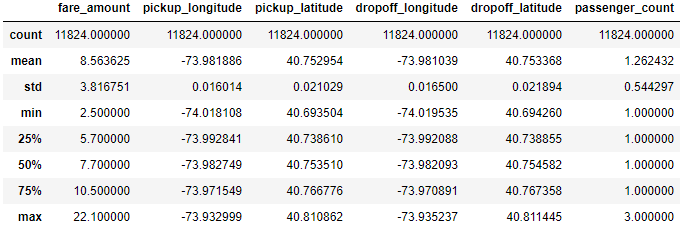


Figure 2.1.2.2

2.1.3 Feature Selection

Since all our variables our numeric so we can extract the important features using the correlation matrix. As we can see from figure it is quite clear that all the variables are important for predicting the fare\_amount as none of the variables have high correlation factor( considering threshold as 0.9).So we keep all the variables for model building.

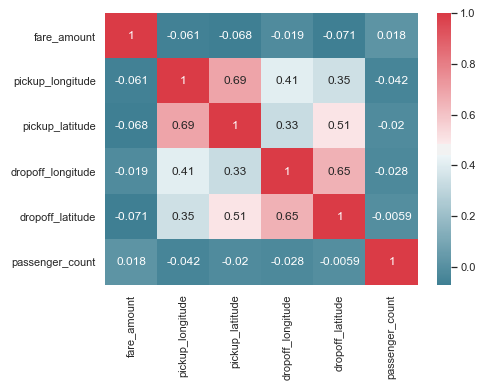


Figure 2.1.3.1

Another method for feature selection is Random Forest. Below is mentioned how we have used Random Forest to extract the importance of each variable.

model\_rf = randomForest(fare\_amount~., train,importance = TRUE, ntree = 300)

importance(model\_rf,type = 1)

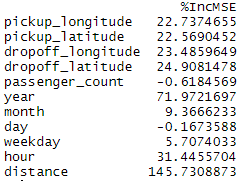


Figure 2.1.3.2

We can see that Distance has the highest prediction power for fare\_amount whereas passenger\_count and day have the least prediction power.

2.1.4 Feature Engineering

It is important to infer some knowledge from the existing data and come up with more valuable information. As the dataset already has datetime variable, we further calculated the year, month, day, weekday and hours that might have an effect on the fare and to further perform some Exploratory Data Analysis on the data.

Also as we have the longitude,latitude points we can easily calculate the distance travelled per ride and derive a relationship between the fare amount and the distance.

To calculate the distance we have used Haversine Distance Formula and calculated the distance in kilometers.

The **Haversine** formula calculates the shortest distance between two points on a sphere using their latitudes and longitudes measured along the surface. It is important for use in navigation. The haversine can be expressed in trignometric function as:



The haversine of the central angle (which is d/r) is calculated by the following formula:



where r is the radius of earth(6371 km), d is the distance between two points, phi1 & ph2 are latitude of the two points and lambda1 & lambda2 are longitude of the two points respectively.

2.2 Modeling

2.2.1 Model Selection

In our early stages of analysis during pre-processing we have come to understand that fare\_amount is dependent on multiple behaviours. Therefore, its important to build a model in such a way that it takes in all the required inputs and fits the model in such a way that it gives us the most accurate result amongst all the other models.

The dependent variable can fall in either of the four categories:

1. Nominal

2. Ordinal

3. Interval

4. Ratio

The dependent variable, in our case Fare Amount, is Ratio so the only predictive analysis that we can perform is a Regression analysis.

We always start our model building from the most simplest to more complex. Therefore we use Multiple Linear Regression at first.

2.2.2

1. **Multiple Linear Regression**

Call:

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

Residuals:

Min 1Q Median 3Q Max

-17.7488 -1.2908 -0.3835 0.8604 15.0773

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -3.072e+02 2.168e+02 -1.417 0.1566

pickup\_longitude 9.195e-01 1.986e+00 0.463 0.6433

pickup\_latitude 3.242e+00 1.586e+00 2.044 0.0410 \*

dropoff\_longitude 2.669e-01 1.835e+00 0.145 0.8844

dropoff\_latitude -1.134e+01 1.456e+00 -7.791 7.35e-15 \*\*\*

passenger\_count 9.641e-02 4.047e-02 2.382 0.0172 \*

year 3.619e-01 1.164e-02 31.090 < 2e-16 \*\*\*

month 5.986e-02 6.315e-03 9.479 < 2e-16 \*\*\*

day -8.153e-04 2.490e-03 -0.327 0.7434

weekday 4.272e-02 1.090e-02 3.920 8.93e-05 \*\*\*

hour 1.719e-02 3.405e-03 5.050 4.50e-07 \*\*\*

distance 1.969e+00 1.368e-02 143.869 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

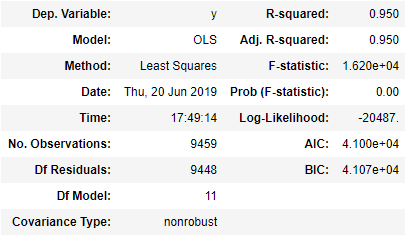
Residual standard error: 2.098 on 9448 degrees of freedom

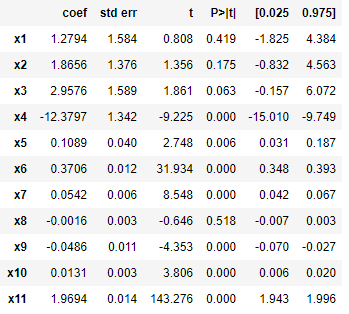
Multiple R-squared: 0.6981, Adjusted R-squared: 0.6978

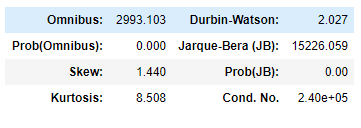
F-statistic: 1986 on 11 and 9448 DF, p-value: < 2.2e-16

As this is a Regression problem and we have multiple variables as our parameters so the first model to be applied is Multiple Linear Regression model. We use the OLS(Ordinary Least Squares) model to fit the model and evaluate its performance.

The summary report gives some valuable information regarding the model built.







 The summary provides several measures to give you an idea of the data distribution and behavior. From here we can see if the data has the correct characteristics to give us confidence in the resulting model. We aren't testing the data, we are just looking at the model's interpretation of the data. If the data is good for modeling, then our residuals will have certain characteristics.

* **Omnibus**– a test of the skewness and kurtosis of the residual. We hope to see a value close to zero which would indicate normalcy, but in our case its far away from 0.
* **Prob(Omnibus)-** performs a statistical test indicating the probability that the residuals are normally distributed. We hope to see something close to 1 here but our data shows 0 so this means data is not normally distributed.
* **Skew**– a measure of data symmetry. We want to see something close to zero, indicating the residual distribution is normal. Our data has the value at a higher side and therefore not recommended.
* **Kurtosis**- Greater Kurtosis can be interpreted as a tighter clustering of residuals around zero, implying a better model with few outliers.
* **Durbin-Watson**– tests for homoscedasticity (characteristic #3). We hope to have a value between 1 and 2. In this case, the data is close, but out of limits.
* **Jarque-Bera (JB)/Prob(JB)**– like the Omnibus test in that it tests both skew and kurtosis. We hope to see in this test a confirmation of the Omnibus test. In this case we do.
* **Condition Number**– This test measures the sensitivity of a function's output as compared to its input (characteristic #4). When we have multicollinearity, we can expect much higher fluctuations to small changes in the data, hence, we hope to see a relatively small number, something below 30. In our case it’s a much greater number.
* R Squared Value- It tells how well the model fits in the data and also explains the variance of the target variable. In this case, R squared value is quite good(95%) which means it explains 95% variability of the target variable. But as the rest of the characteristics are not satisfactory, its better to use some other model for prediction.

RMSE score : 2.1319464655728355

1. **Decision Tree**

Decision trees are non linear. Unlike Linear regression there is no equation to express relationship between independent and dependent variables. So for a better result the next model that we can choose is Decision Tree Regression.

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question; and the leaves represent the actual output or class label.

Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

maxDepth : 5 larger the dataset harder to visualize so we have taken the maximum branching to be 5 as of now.

fit = DecisionTreeRegressor(max\_depth=5).fit(train.iloc[:,1:],train.iloc[:,0])

Choosing the maxDepth as 5, we get an RMSE score of :

RMSE value for Decision Tree is: 2.1091782572686593

Which is less than the RMSE score obtained by multiple linear regression. So this algorithm definitely performs better. But there is still a scope of improvement.

So we move on to the next model.

1. **Random Forest**

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. Ensembling is nothing but a combination of weak learners (individual trees) to produce a strong learner.

Random Forest can be used to solve regression and classification problems. In regression problems, the dependent variable is continuous. In classification problems, the dependent variable is categorical.

1. **KNN**

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

KNN makes predictions using the training dataset directly.

Predictions are made for a new instance (x) by searching through the entire training set for the K most similar instances (the neighbors) and summarizing the output variable for those K instances. For regression this might be the mean output variable, in classification this might be the mode (or most common) class value.

To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance).

Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.

EuclideanDistance(x, xi) = sqrt( sum( (xj – xij)^2 ) )

**Conclusion**

**3.1 Model Evaluation**

The quality of a regression model is how well its predictions match up against actual values, but how do we actually evaluate quality? Luckily, smart statisticians have developed **error metrics** to judge the quality of a model and enable us to compare regressions against other regressions with different parameters.

As our model deals with regression so we will choose amongst those error metrics that are used for regression.

If our collection of residuals are small, it implies that the model that produced them does a good job at predicting our output of interest. Conversely, if these residuals are generally large, it implies that model is a poor estimator.

Thus, statisticians have developed summary measurements that take our collection of residuals and condense them into a single value that represents the predictive ability of our model.

**For our problem we have used RMSE and R squared as we are dealing with time series forecasting and continuous variables.**

The Root Mean Squared Error measures the square root of the average of the squared difference between the predictions and the ground truth.

1. RMSE is in the same units as the dependent variable. Hence, RMSE must be compared with the dependent variable.
2. Smaller the result, the better our model is performing.

To understand how well the independent variables “explain” the variance in our model, the R-Squared formula is used.

1. For the R-Squared, the closer the value to 1, the better our model is performing.

According to our model, the following table describes its error metrics:

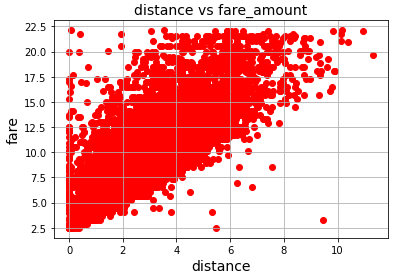
|  |  |  |
| --- | --- | --- |
| Model | RMSE Score | R Square |
| Multiple Linear Regression | 2.1319464655728355 | 0.95 |
| Decision Tree | 2.1091782572686593 | 0.6790662606031613 |
| Random Forest | 1.9953043471990368 | 0.7127850099199918 |
| K Nearest Neighbour | 2.6025145530297236 | 0.5297013732475271 |

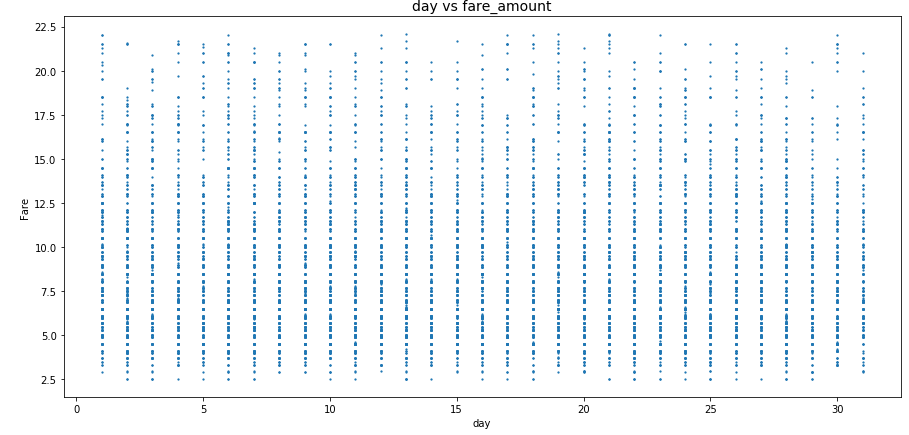
We reject Multiple Linear Regression as the key assumptions of a Linear regression model are being violated. Out of the 3 models left, Random Forest is the best model as it has the lowest RMSE score and highest R Square which explains the highest variability and tells us how well the model fits in the data.

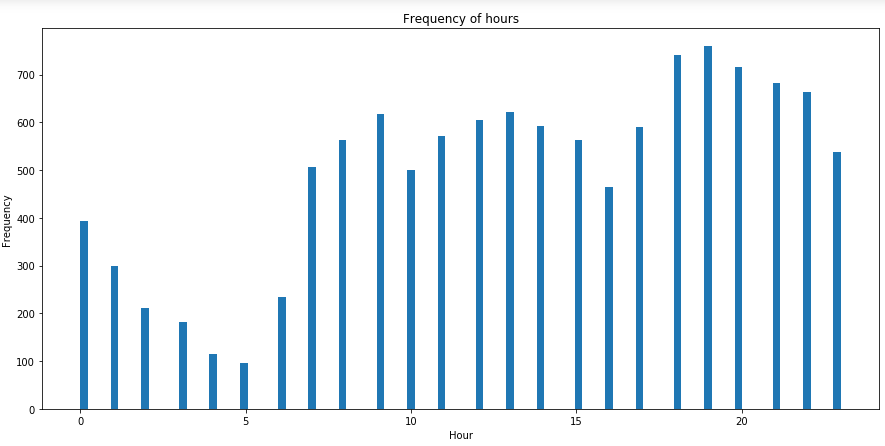
**3.2 Model Selection**

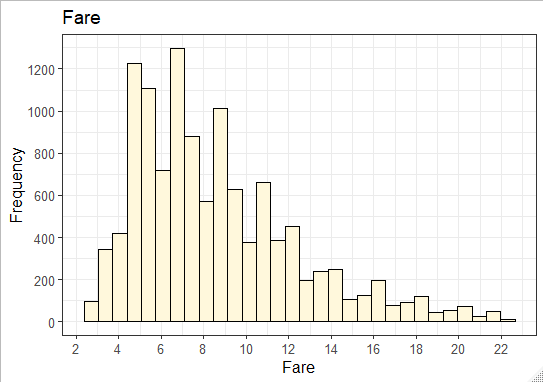
We select Random Forest as our final model because of lowest RMSE score and highest R square.

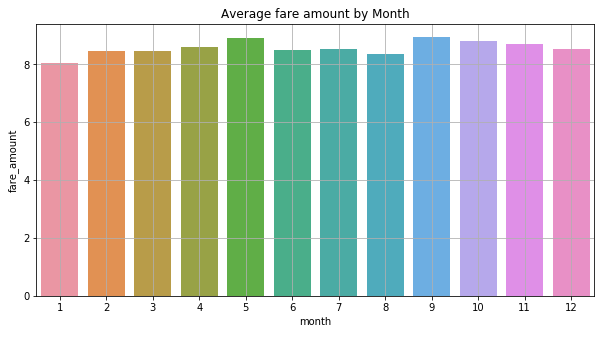
Appendix A - Extra Figures

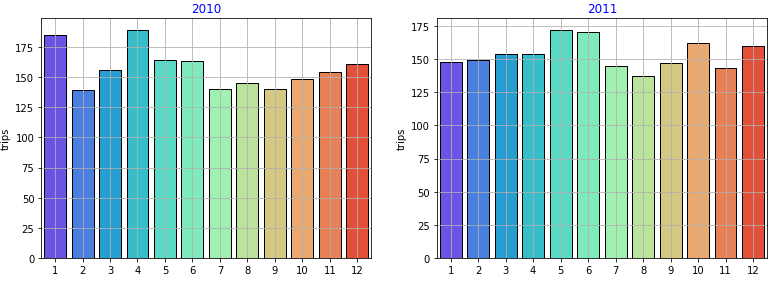


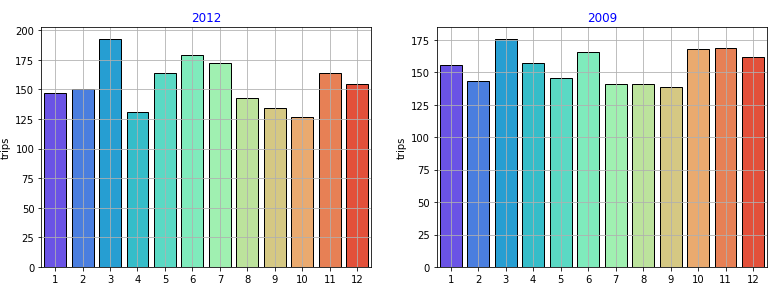


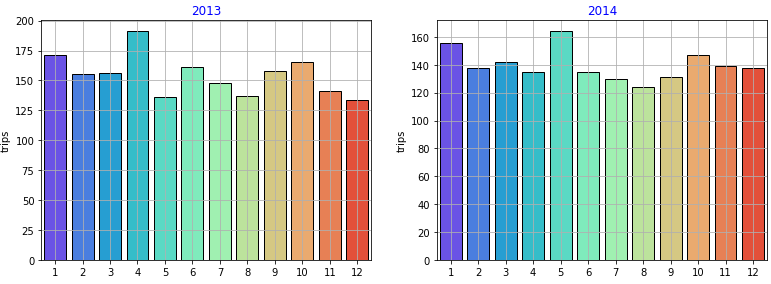












Appendix B - R Code

rm(list = ls())

setwd('C:/Users/admin/Desktop')

cab\_df = read.csv('train\_cab.csv',header = T)

x = c("ggplot2","corrgram","DMwR","caret","randomForest","unbalanced",

"C50","dummies","e1071","Information","MASS","rpart","gbm","ROSE","lubridate"

,"dplyr")

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

rm(x)

str(cab\_df)

cab\_df$fare\_amount = as.numeric(as.character(cab\_df$fare\_amount))

cab\_df$pickup\_datetime = parse\_date\_time(cab\_df$pickup\_datetime,orders = "ymd HMS")

missing\_val = data.frame(apply(cab\_df, 2,function(x)(sum(is.na(x))) ))

missing\_val$columns = row.names(missing\_val)

names(missing\_val)[1] = "Missing\_Percentage"

missing\_val$Missing\_Percentage = (missing\_val$Missing\_Percentage/nrow(cab\_df))\*100

missing\_val = missing\_val[order(-missing\_val$Missing\_Percentage),]

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

row.names(missing\_val) = NULL

colSums(is.na(cab\_df))

cab\_df = subset(cab\_df,!is.na(pickup\_datetime))

#Actual value : 6.9

#mean : 15.01

#median : 8.5

#knn : 7.01

cab\_df[71,7] = NA

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

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

cab\_df$pickup\_datetime = as.numeric(cab\_df$pickup\_datetime)

cab\_df = knnImputation(cab\_df,k = 7)

cab\_df$pickup\_datetime = as.POSIXct( as.numeric( as.POSIXct( cab\_df$pickup\_datetime, origin = '1970-01-01', tz = "UTC" ) ),

origin = '1970-01-01', tz = "UTC" )

cab\_df$passenger\_count = round(cab\_df$passenger\_count)

summary(cab\_df)

numeric\_index = sapply(cab\_df, is.numeric)

numeric\_data = cab\_df[,numeric\_index]

cnames = colnames(numeric\_data)

boxplot(cab\_df$fare\_amount,cab\_df$pickup\_longitude,cab\_df$pickup\_latitude)

boxplot(cab\_df$dropoff\_longitude,cab\_df$dropoff\_latitude,cab\_df$passenger\_count)

for (i in cnames){

print(i)

val = cab\_df[,i][cab\_df[,i]%in%boxplot.stats(cab\_df[,i])$out]

print(length(val))

cab\_df = cab\_df[which(!cab\_df[,i]%in%val),]

}

summary(cab\_df)

cab\_df = subset(cab\_df,fare\_amount >= 1)

cab\_df = subset(cab\_df,passenger\_count>=1)

distance = function (long1, lat1, long2, lat2)

{

rad <- pi/180

a1 <- lat1 \* rad

a2 <- long1 \* rad

b1 <- lat2 \* rad

b2 <- long2 \* rad

dlon <- b2 - a2

dlat <- b1 - a1

a <- (sin(dlat/2))^2 + cos(a1) \* cos(b1) \* (sin(dlon/2))^2

c <- 2 \* atan2(sqrt(a), sqrt(1 - a))

R <- 6378.145

d <- R \* c

return(d)

}

cab\_df$year = year(cab\_df$pickup\_datetime)

cab\_df$month = month(cab\_df$pickup\_datetime)

cab\_df$day = day(cab\_df$pickup\_datetime)

cab\_df$weekday = wday(cab\_df$pickup\_datetime)

cab\_df$hour = hour(cab\_df$pickup\_datetime)

cab\_df$pickup\_datetime = NULL

cab\_df$distance = distance(cab\_df$pickup\_longitude,cab\_df$pickup\_latitude,

cab\_df$dropoff\_longitude,cab\_df$dropoff\_latitude)

corrgram(cab\_df[,numeric\_index],order = F,

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

library("scales")

library("psych")

library("gplots")

ggplot(cab\_df,aes\_string(x=cab\_df$distance,y=cab\_df$fare\_amount))+

geom\_point(inherit.aes = TRUE,size=3)+

theme\_bw()+ylab("Fare")+xlab("Distance")+ggtitle("Scatter Plot b/w distance and fare")+

theme(text=element\_text(size = 15))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+

scale\_y\_continuous(breaks = pretty\_breaks(10))

ggplot(cab\_df,aes\_string(x = cab\_df$hour))+

geom\_bar(stat = "count",fill = "DarkSlateBlue")+theme\_bw()+

xlab("Hours")+ylab("Frequency")+scale\_y\_continuous(breaks = pretty\_breaks(20))+

ggtitle("Frequency of hours")+theme(text = element\_text(size = 12))

ggplot(cab\_df,aes\_string(x=cab\_df$hours,y=cab\_df$fare\_amount))+

geom\_point(aes\_string(cab\_df$hour,cab\_df$fare\_amount),size=3)+

theme\_bw()+ylab("Fare")+xlab("Hours")+ggtitle("Scatter Plot b/w hours and fare")+

theme(text=element\_text(size = 15))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+

scale\_y\_continuous(breaks = pretty\_breaks(10))

ggplot(cab\_df,aes\_string(x = cab\_df$passenger\_count))+

geom\_histogram(fill = "red",colour="black",bins = 15)+geom\_density()+

scale\_y\_continuous(breaks = pretty\_breaks(10))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+theme\_bw()+

xlab("Passengers")+ylab("Frequency")+

ggtitle("Frequency of Passengers")+theme(text = element\_text(size = 10))

ggplot(cab\_df,aes\_string(x = cab\_df$weekday))+

geom\_histogram(fill = "green",colour="black",bins = 70)+geom\_density()+

scale\_y\_continuous(breaks = pretty\_breaks(10))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+theme\_bw()+

xlab("Weekdays")+ylab("Frequency")+

ggtitle("Frequency of weekdays")+theme(text = element\_text(size = 10))

ggplot(cab\_df,aes\_string(x = cab\_df$fare\_amount))+

geom\_histogram(fill = "cornsilk",colour="black")+geom\_density()+

scale\_y\_continuous(breaks = pretty\_breaks(10))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+theme\_bw()+

xlab("Fare")+ylab("Frequency")+

ggtitle("Fare")+theme(text = element\_text(size = 12))

ggplot(cab\_df,aes\_string(x=cab\_df$day,y=cab\_df$fare\_amount))+

geom\_point(aes\_string(cab\_df$day,cab\_df$fare\_amount),size=3)+

theme\_light()+ylab("Fare")+xlab("Day")+ggtitle("Scatter Plot b/w day and fare")+

theme(text=element\_text(size = 15))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+

scale\_y\_continuous(breaks = pretty\_breaks(10))

ggplot(cab\_df,aes\_string(x=cab\_df$passenger\_count,y=cab\_df$fare\_amount))+

geom\_point(aes\_string(cab\_df$day,cab\_df$fare\_amount),size=3)+

theme\_light()+ylab("Fare")+xlab("Passengers")+ggtitle("Scatter Plot b/w Passenger and fare")+

theme(text=element\_text(size = 15))+

scale\_x\_continuous(breaks = pretty\_breaks(10))+

scale\_y\_continuous(breaks = pretty\_breaks(10))

library(rsq)

library(usdm)

vif(cab\_df[,2:8])

vifcor(cab\_df[,2:8],th = 0.9)

set.seed(1234)

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

train = cab\_df[train\_index,]

test = cab\_df[-train\_index,]

#Multiple Linear Regression

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

summary(lm\_model)

predictions\_lr = predict(lm\_model,test[,2:12])

RMSE(predictions\_lr,test$fare\_amount)

#2.17

#Decision Tree

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

predictions\_dt = predict(fit,test[,-1])

RMSE(predictions\_dt,test$fare\_amount)

#2.30

#Random Forest

model\_rf = randomForest(fare\_amount~.,

train,importance = TRUE, ntree = 300)

RF\_predictions = predict(model\_rf,test[,2:12])

RMSE(RF\_predictions,test$fare\_amount)

#1.99

test\_df = read.csv('test.csv',header = T)

summary(test\_df)

str(test\_df)

test\_df$pickup\_datetime = parse\_date\_time(test\_df$pickup\_datetime,orders = "ymd HMS")

num\_index = sapply(test\_df, is.numeric)

num\_data = test\_df[,num\_index]

c\_names = colnames(num\_data)

for (i in c\_names){

print(i)

val = test\_df[,i][test\_df[,i]%in%boxplot.stats(test\_df[,i])$out]

print(length(val))

test\_df = test\_df[which(!test\_df[,i]%in%val),]

}

summary(test\_df)

test\_df$year = year(test\_df$pickup\_datetime)

test\_df$month = month(test\_df$pickup\_datetime)

test\_df$day = day(test\_df$pickup\_datetime)

test\_df$weekday = wday(test\_df$pickup\_datetime)

test\_df$hour = hour(test\_df$pickup\_datetime)

test\_df$pickup\_datetime = NULL

test\_df$distance = distance(test\_df$pickup\_longitude,test\_df$pickup\_latitude,

test\_df$dropoff\_longitude,test\_df$dropoff\_latitude)

test\_df = subset(test\_df,distance>=1)

predicted\_fare = predict(model\_rf,test\_df[,])

test\_df$predicted\_fare = predicted\_fare

summary(test\_df)

**References**

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An Introduction to Statistical Learning. Vol. 6. Springer. Wickham, Hadley. 2009. Ggplot2: Elegant Graphics for Data Analysis. Springer Science & Business Media.