Cab Fare Prediction

SHREYAS AINAPUR

04/04/2020

CONTENTS

Chapter 1

Introduction

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

Our task is to build a regression model which will help in predicting the fare of a cab based on the various factors. Given below is a sample of the data set that we are using to predict the cab fare:

Table 1.1 : Training Data with first five observations

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1.0
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1.0
2	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2.0
3	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1.0
4	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1.0

Table 1.2 : Testing Data with first five observations

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2015-01-27 13:08:24 UTC	-73.973320	40.763805	-73.981430	40.743835	1
1	2015-01-27 13:08:24 UTC	-73.986862	40.719383	-73.998886	40.739201	1
2	2011-10-08 11:53:44 UTC	-73.982524	40.751260	-73.979654	40.746139	1
3	2012-12-01 21:12:12 UTC	-73.981160	40.767807	-73.990448	40.751635	1
4	2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	1

As you can see, training data has 7 variables and testing data has 6 variables. 7th variable in training data is ‘fare_amount’ which is the dependent/target variable. Using training data we have to build a model which can be used to predict fare_amount for testing data.

Below are the variables used in the model to predict cab fare amount.

Table 1.3 : Independent Variables

S.No.	Independent Variables
1	pickup_datetime
2	pickup_latitude
3	pickup_longitude
4	dropoff_latitude
5	dropoff_longitude
6	passenger_count

Values of these variables are independent of each other and hence the name, these can also be called predictor variables as these are used to predict the target variable.

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. 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 Exploratory Data Analysis. To start this process we will first need to perform **missing value analysis** and impute missing values with one of the methods among mean method or median or KNN imputation. By laying a framework the best method to impute the missing value is chosen.

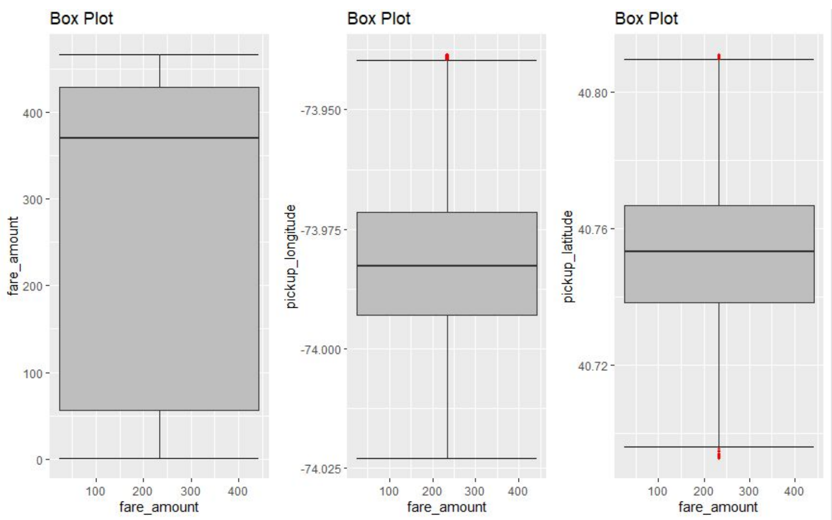
```
#Creating a framework to choose best method for the imputation
#train_data[3,4] is selected for the framework
#Actual Value = 40.76127
#Mean method = 39.91467
#Medain method = 40.7526
#KNN method = 40.73314
```

Fig 2.1

In Figure 2.1 we can see the data of the framework in which actual value was replaced by 'NA' and various imputation method was performed and the answer obtained by KNN imputation was the closest to the actual value and hence used for the imputation of the rest of the missing value.

2.1.1 Outlier Analysis

One of the other steps of **pre-processing** is checking the presence of outliers. In this case we use a classic approach of removing outliers, Tukey's method. We visualize the outliers using **boxplots**.



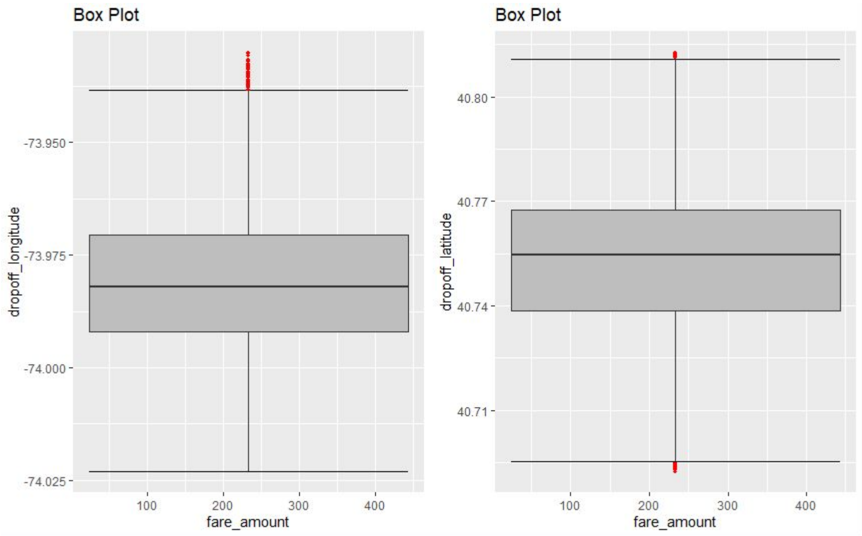


Fig 2.2

In Figure 2.2 we can see the plotting of boxplot five different variables against the target variable. The red dots in the individual images are the outliers. All the data points appearing below and above the lower and upper fence respectively are the outliers represented by red dots.

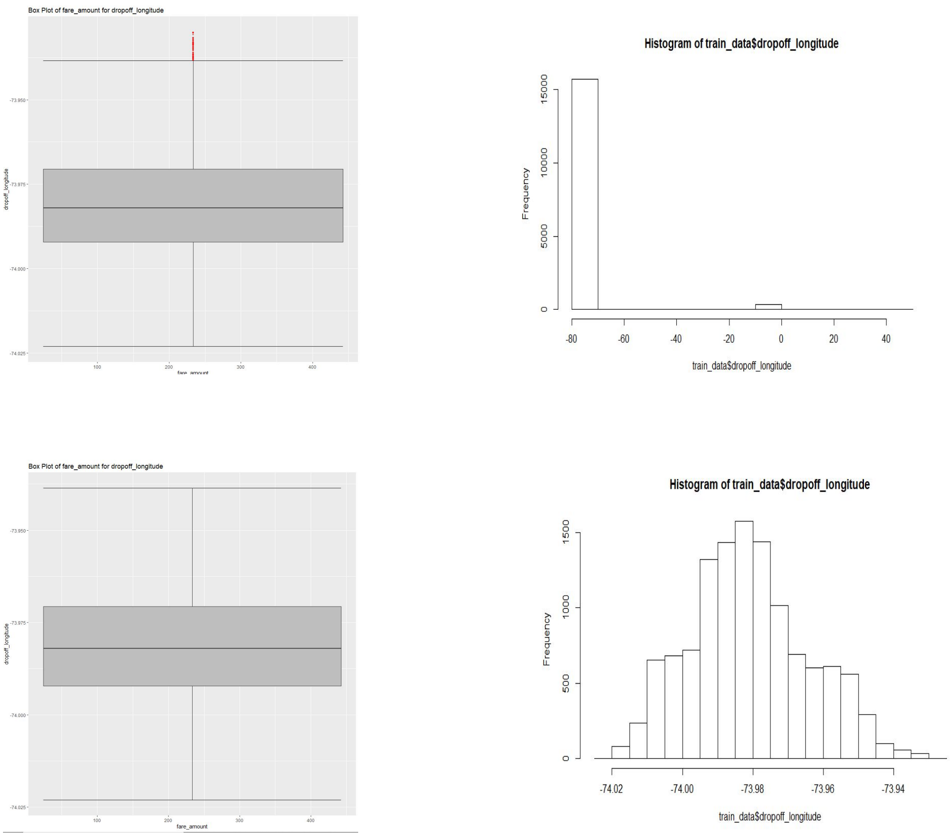


Fig 2.3 : Boxplot and data distribution of dropoff_longitude with and without outliers respectively

In Figure 2.3 we can see the shift of the median in the boxplot with and without the presence of outliers and also the change in the histogram of data distribution of dropoff_longitude with and without the presence of outliers.

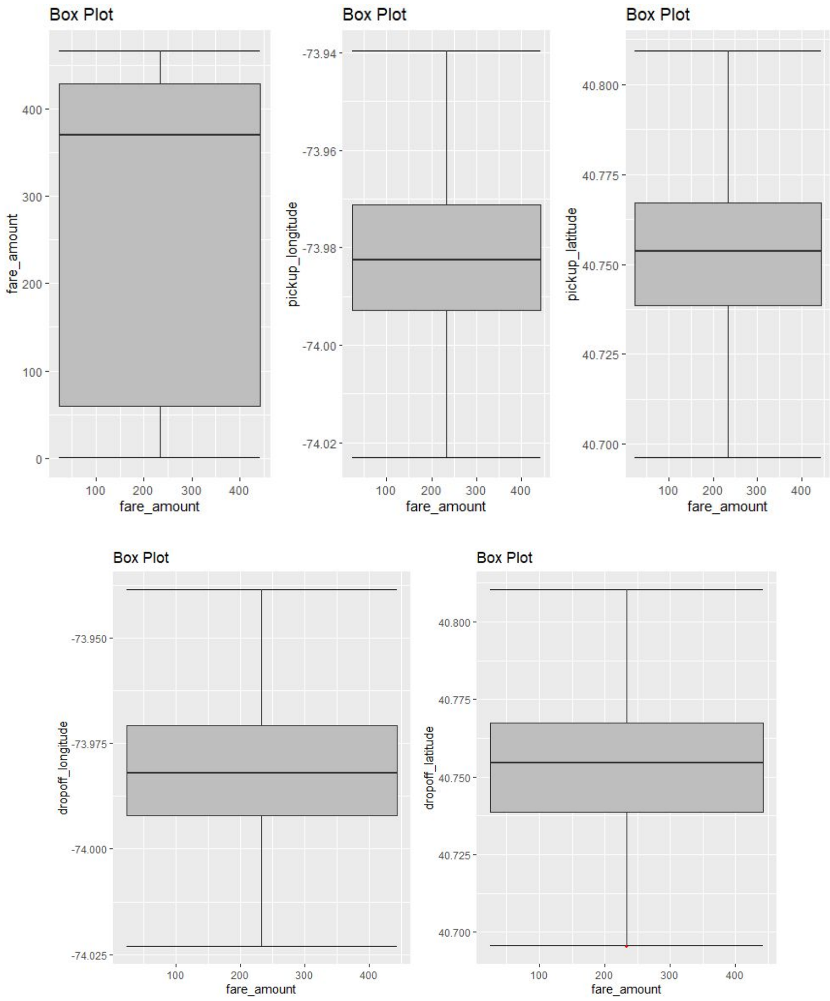


Fig 2.4 : Boxplot after removing outliers

In Figure 2.4 we can see that there are no red dots below or above the lower or upper fence respectively. The difference in the image Fig 2.2 and Fig 2.4 is the **shift in the median line** i.e after the outliers are removed the median value of the variables is changed.

2.1.2 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that.

In fig 2.4 we have plotted a correlation matrix to draw the relationship of each variable against the dependent variable. Variables with no correlation with the target (dependent) variable will be removed as it won't have any impact on the output.

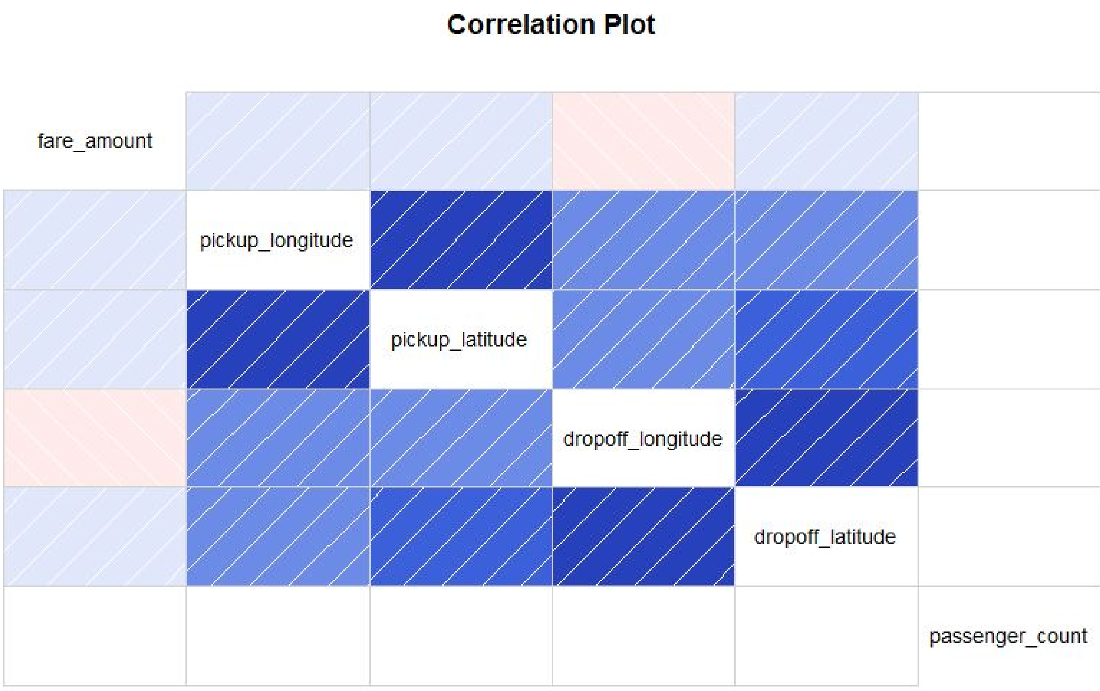


Fig 2.4 : Correlation Matrix

In the figure 2.4 blue colour represents positive correlativity and pink represents negative correlativity. White colour represents zero correlation. As we can see passenger_count has no correlation with fare_amount which is our target variable, hence, passenger_count can be removed as it won't have any impact on the output.

2.2 Modeling

2.2.1 Model Selection

If the dependent variable is Nominal the only predictive analysis that we can perform is Classification, and if the dependent variable is Interval or Ratio the normal method is to do a Regression analysis, or classification after binning. And for a dependent variable which is Ordinal, both classification and regression can be done.

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

1. Nominal
2. Ordinal
3. Interval
4. Ratio

The dependent variable which we are dealing is ordinal (continuous) and hence we perform Regression analysis on the model for the prediction of the target variable.

2.2.2 Multiple Linear Regression

```
No variable from the 4 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( dropoff_longitude ~ pickup_latitude ):  0.3236623
max correlation ( pickup_latitude ~ pickup_longitude ):  0.7033039

---------- VIFs of the remained variables --------
          Variables      VIF
1  pickup_longitude 2.206069
2   pickup_latitude 2.380909
3 dropoff_longitude 1.961892
4  dropoff_latitude 2.133487
```

Fig 2.5 : Variation Inflation Factor

In Figure 2.5 we have calculated VIF to check for the collinearity problem and there was no problem of collinearity for the variables.

Now we have generated a model and its summary is given below in Figure 2.6.

```
Call:
lm(formula = fare_amount ~ ., data = train[-2])

Residuals:
    Min      1Q  Median      3Q     Max 
-302.65 -216.08   85.61  146.47  243.41 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        266.392      6.979  38.173  < 2e-16 ***
pickup_longitude    21.018     15.265   1.377    0.169    
pickup_latitude     -3.692     16.314  -0.226    0.821    
dropoff_longitude  -75.273     14.062  -5.353 8.92e-08 ***
dropoff_latitude    87.341     15.204   5.745 9.57e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 173 on 7515 degrees of freedom
Multiple R-squared:  0.006054,	Adjusted R-squared:  0.005525 
F-statistic: 11.44 on 4 and 7515 DF,  p-value: 2.924e-09
```

Fig 2.6 : Summary of the model

From the summary of the model various factors as shown in the Figure 2.6 can be depicted. The variables with more stars are more significant variables.

Following were the values depicted from Figure 2.6 ;

Residual error = 173

Multiple R-squared = 60.54%

Adjusted R-Squared = 55.25%

F-Statistic = 11.44

P-value = 2.924e-09

2.2.3 Regression Trees

Now we will try and use a different regression model to predict our *fare_amount* target variable. We will use a decision tree to predict the values of our target variable.

The function used for the decision tree model training is “rpart” and the method used to solve is “annova” . After the model is trained it is used on the test data to predict its target variable using “predict” function.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using error metrics.

1. Classification Metrics
   - Confusion Matrix
   - Accuracy
   - Misclassification error
   - Specificity
   - Recall
2. Regression Metrics
   - MAE
   - RMSE
   - MAPE

In our case we will use one of the methods that comes under Regression Metrics as the criteria to compare and evaluate models. Regression Metrics can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

We have used MAPE method as our target variable is not time based data and as MAPE gives the error in terms of percentage which is easier to analyse the amount of error obtained unlike in MAE in which we get the value of the error.

```
# run regression model
lm_model = lm(fare_amount~., data = train[-2])

summary(lm_model)

predictions_LR = predict(lm_model, test[,-1])

mape(test[,1], predictions_LR)
#Error Rate = 3.95%
#Accuracy96.05
```

Fig 3.1 : MAPE on regression model

In Figure 3.1 obtained accuracy is 96.05% on regression model.

In Figure 3.2 given below obtained accuracy 94.46% on model using decision tree.

```
# rpart for regression
fit = rpart(fare_amount~., data = train, method = "anova")

#predict for new test cases
predictions_DT = predict(fit, test[,-1])

# Error metrics using MAPE
mape = function(y, yhat){
            mean(abs((y-yhat)/y))
}

mape(test[,1], predictions_DT)
# Error Rate = 5.53%
# Accuracy = 94.46%
# Thus this model can be used for the prediction
```

Fig 3.2 : MAPE on decision tree model

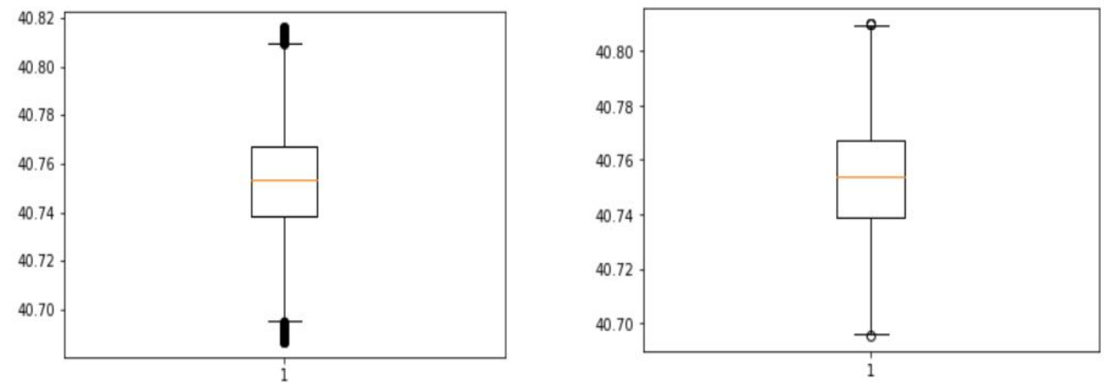
3.2 Model Selection

We can see that both the model has acceptable accuracy with very low error and hence any of the model can be used.

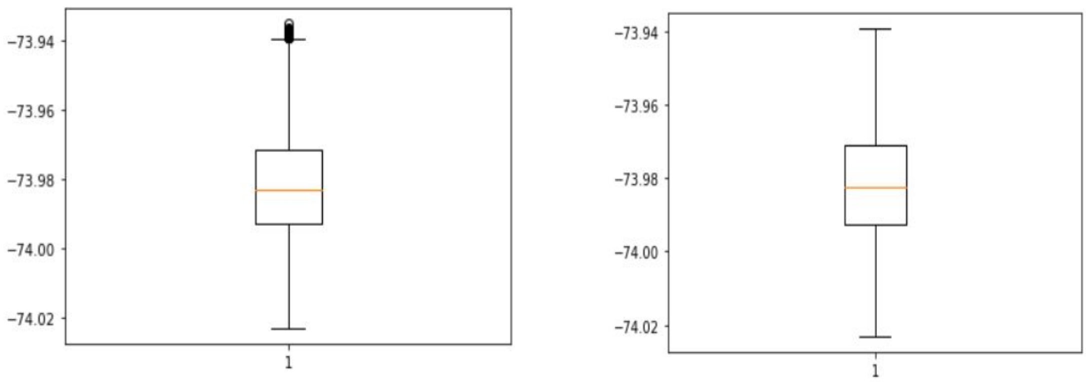
Appendix A - Figures from Jupyter Notebook

<u>Boxplot of independent variables</u>

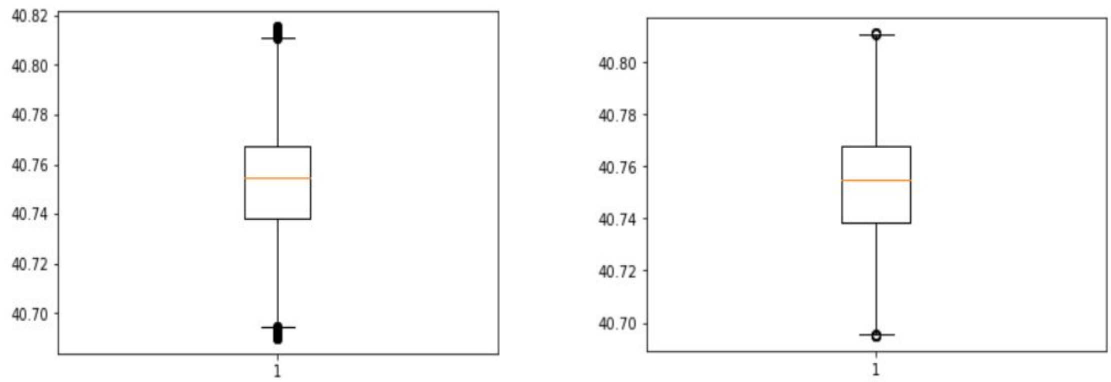
With Outliers | Without Outliers



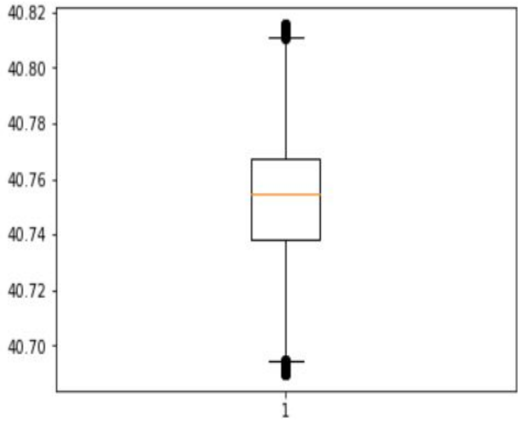
pickup_latitude

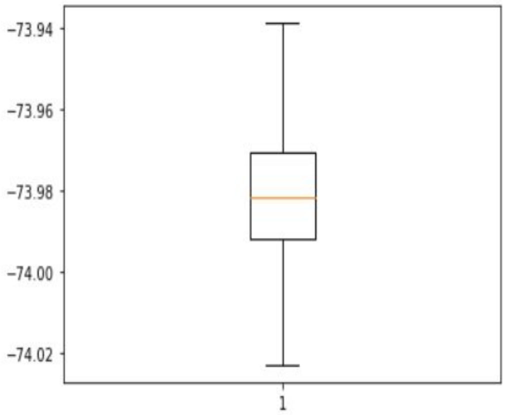


pickup_longitude



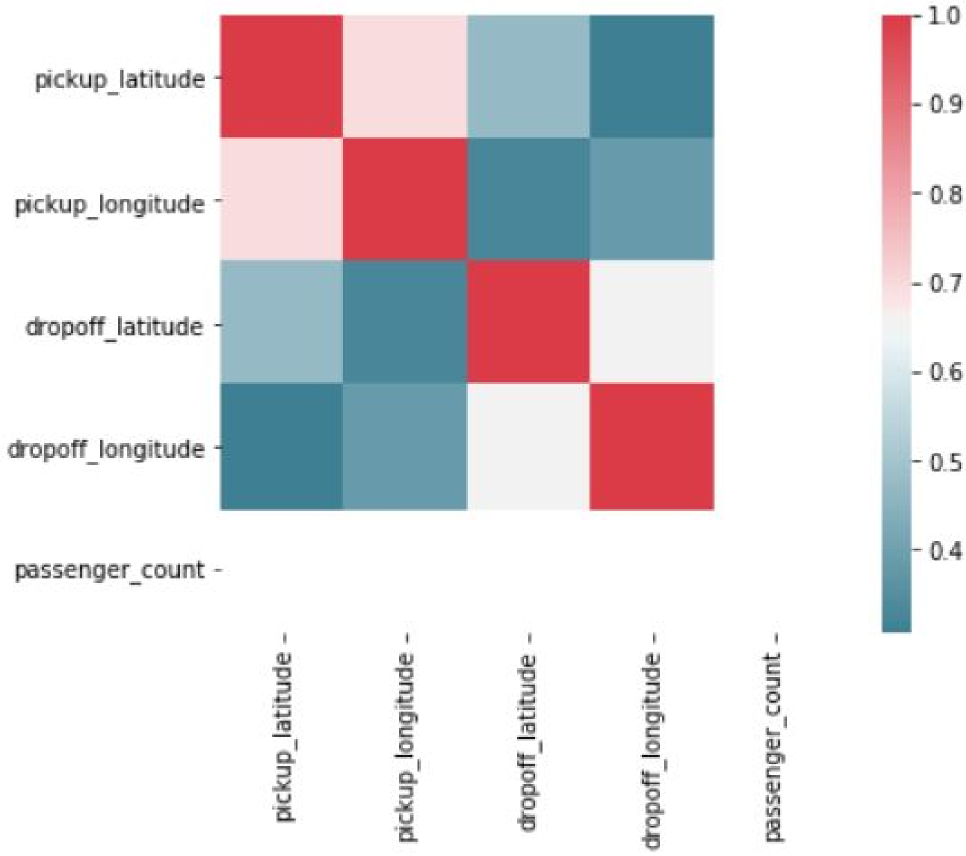
dropoff_latitude





dropoff_longitude

<u>Correlation Matrix</u>



Appendix B - R Code

```
rm(list=ls())
setwd("C:/Users/shreyas/Documents/Data Science/Project - 2/test")
# Loading Libraries

x = c("ggplot2", "corrgram","DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071",
      "MASS", "rpart", "gbm", "ROSE" )
lapply(x, require, character.only = TRUE)
rm(x)

# Loading CSV File
test_data = read.csv("test.csv", header = T, na.strings = c(" ", "", "NA"))
train_data = read.csv("train_cab.csv", header = T, na.strings = c(" ","", "NA"))

str(train_data)

#Data Manipulation : converting categorical variables into factor numeric type
train_data$fare_amount = as.numeric(train_data$fare_amount)

if(class(train_data[,2]) == "factor"){
    train_data[,2] = factor(train_data[,2], labels = (1:length(levels(factor(train_data[,2])))))
  }
###Imputing Missing Value
#checking if there is any missing value
missing_val = data.frame(apply(train_data, 2, function(x){sum(is.na(x))}))
missing_val$columns = row.names(missing_val)
row.names(missing_val) = NULL
names(missing_val)[1] = "missing_percentage"
missing_val$missing_percentage = (missing_val$missing_percentage/nrow(train_data))*100
missing_val = missing_val[order(-missing_val$missing_percentage),]
missing_val = missing_val[,c("columns", "missing_percentage")]


#Creating a framework to choose best method for the imputation
#train_data[3,4] is selected for the framework
#Actual Value = 40.76127
#Mean method = 39.91467
#Medain method = 40.7526
#KNN method = 40.73314

# Trial to choose best method
mean(train_data$pickup_latitude, na.rm = TRUE)
median(train_data$pickup_latitude, na.rm = TRUE)
knnImputation(train_data, k = 5)

#knnImputation method is choosen for replaceing missing values
train_data = knnImputation(train_data, k = 5)

#Check if there are still any missing value
sum(is.na(train_data))
```

```r
###Applying data pre-processing methods to remove unncessary variables/observations
#Outlier Analysis
df = train_data

numeric_index = sapply(train_data, is.numeric)
numeric_data = train_data[,numeric_index]

cnames = colnames(numeric_data)

for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "fare_amount"), data = train_data)+
           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")))
}

#plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3, ncol=3)

# Remove Outliers
val = train_data$pickup_longitude[train_data$pickup_longitude%in%boxplot.stats(train_data$pickup_longitude)$out]
train_data = train_data[which(!train_data$pickup_longitude%in%val),]

# Loop to remove outlier from all the variables
for (i in cnames){
  print(i)
  val = train_data[,i][train_data[,i]%in%boxplot.stats(train_data[,i])$out]
  train_data = train_data[which(!train_data[,i]%in%val),]
}

###Preparing model to check for collinearity among the variables and to train the model
#checking and plotting for Correlation
corrgram(train_data[,numeric_index],order = F,
         upper.pannel = panel.pie, text.panel = panel.txt, main = "Correlation Plot")

#Dimension reduction
train_data_deleted = subset(train_data,
                            select = -(passenger_count))
```

```
# Normalisation or Standardisation
qqnorm(train_data$pickup_longitude)
hist(train_data$dropoff_longitude)

# Standardisation
#conames = c("pickup_longitude", "pickup_latitude", "dropoff_longitude", "dropoff_latitude")
#for (i in conames){
 # print(i)
  #train_data_deleted[,i] = (train_data_deleted[,i]-mean(train_data_deleted[,i]))/
   #                           sd(train_data_deleted[,i])
#}

# Normalistion
conames = c("pickup_longitude", "pickup_latitude", "dropoff_longitude", "dropoff_latitude")

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

# Multiple Linear Regression
# Check Multicollinearity
# dividing into train and test data by 80-20
train_index = sample(1:nrow(train_data_deleted), 0.8*nrow(train_data_deleted))
train = train_data_deleted[train_index,]
test = train_data_deleted[-train_index,]

library(usdm)

vif(train_data_deleted[,c(-1, -2)])

vifcor(train_data_deleted[,c(-1, -2)], th = 0.9)

# run regression model
lm_model = lm(fare_amount~., data = train[-2])

summary(lm_model)

predictions_LR = predict(lm_model, test[,-1])

mape(test[,1], predictions_LR)
#Error Rate = 3.95%
#Accuracy96.05


# Using Decision Tree for predicting target variable
# dividing into train and test data by 80-20

train_index = sample(1:nrow(train_data_deleted), 0.8*nrow(train_data_deleted))
train = train_data_deleted[train_index,]
test = train_data_deleted[-train_index,]

# rpart for regression
fit = rpart(fare_amount~., data = train, method = "anova")

#predict for new test cases
predictions_DT = predict(fit, test[,-1])

# Error metrics using MAPE
mape = function(y, yhat){
            mean(abs((y-yhat)/y))
}

mape(test[,1], predictions_DT)
# Error Rate = 5.53%
# Accuracy = 94.46%
# Thus this model can be used for the prediction
```

Appendix C - Python Code

```
In [ ]: import os
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from scipy.stats import chi2_contingency
        from random import randrange, uniform
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.model_selection import train_test_split
```

```
In [ ]: os.getcwd()
```

```
In [ ]: training_data = pd.read_csv("train_cab.csv", encoding="ISO-8859-1")
        training_data = training_data.replace(["", " ", "NAN"], np.NAN)
```

```
In [ ]: testing_data = pd.read_csv("test.csv", encoding="ISO-8859-1")
```

```
In [ ]: # Create a data frame with missing values
        missing_val = pd.DataFrame(training_data.isnull().sum())
```

```
In [ ]: # reset the index
        missing_val = missing_val.reset_index()
```

```
In [ ]: # rename variables
        missing_val = missing_val.rename(columns = {'index' : 'variables', 0: 'Missing_percentage'})
```

```
In [ ]: # Calculate percentage
        missing_val['Missing_percentage'] = (missing_val['Missing_percentage']/len(training_data))*100
```

```
In [ ]: # descending order
        missing_val = missing_val.sort_values('Missing_percentage', ascending = False).reset_index(drop = True)
```

```
In [ ]: #save output results
        missing_val.to_csv("Missing_perc.csv", index = False)
        missing_val
```

```
In [ ]: # Laying a framework to choose best method for imputing missing value
        # Record the data
        # Actual Value = 40.71
        # median       = 40.75
        # mean         = 39.91
```

```
In [ ]: # create a missing value to choose the best method for imputation of missing values
        training_data['pickup_latitude'].loc[2] = np.NAN
```

```
In [ ]: #Imputation with mean method
        training_data['pickup_latitude'] = training_data['pickup_latitude'].fillna(training_data['pickup_latitude'].mean())
```

```
In [ ]: # Imputation with median method
        training_data['pickup_latitude'] = training_data['pickup_latitude'].fillna(training_data['pickup_latitude'].median())
```

```
In [ ]: # Correcting the created missing value with actual value
        training_data['pickup_latitude'].loc[2] = 40.71
```

```
In [ ]: # Only two variables 'passenger_count' and 'fare_amount' has missing value
        # Median method is very close to the actual value, hence median method is chosen to impute missing value
        training_data["passenger_count"] = training_data["passenger_count"].fillna(training_data["passenger_count"].median())
```

```
In [ ]: # Since "fare_amount" is of object type it cannot be imputed using mean or median
        training_data['fare_amount'] = training_data['fare_amount'].fillna(training_data['fare_amount'].value_counts().index[0])
```

```
In [ ]: # recheck for the missing values
        missing_val_new = pd.DataFrame(training_data.isnull().sum())
```

```
In [ ]: # Outlier Analysis using boxplot method
        # first store the data in another variable
        df = training_data.copy()
```

```
In [ ]: # Plot boxplot visualise outliers
        %matplotlib inline
        # boxplot can be plotted for individual variables as shown below
        plt.boxplot(training_data["dropoff_latitude"])
```

```
In [ ]: # Save numeric names
        cnames = ["pickup_latitude", "pickup_longitude", "dropoff_latitude", "dropoff_longitude", "passenger_count"]
```

```
In [ ]: # detect and delete outliers from data
        for i in cnames:
            print(i)
            q75, q25 = np.percentile(training_data.loc[:,i], [75,25])
            iqr = q75 - q25

            min = q25 - (1.5*iqr)
            max = q75 + (1.5*iqr)
            print(min)
            print(max)

            training_data = training_data.drop(training_data[training_data.loc[:,i] < min].index)
            training_data = training_data.drop(training_data[training_data.loc[:,i] > max].index)
```

```
In [ ]: # After outliers are removed the number of observations dropped from 16067 to 9457 using boxplot
```

```
In [ ]: ### Preparing model to check for collinearity among the variables and to train the model
        # Performing correlation Analysis
        num_corr = training_data.loc[:,cnames]
```

```
In [ ]: # Set the width and height of the plot
        h, wd = plt.subplots(figsize=(7,5))

        # Generate correlation Matrix
        corr = num_corr.corr()

        # Plot using seaborn library
        sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
                    square=True, ax=wd)
```

```
In [ ]: # Passenger_count is not correltaed to any one of the variable
```

```
In [ ]: # Chi-square test of independence
        chi2, p , dof, ex = chi2_contingency(pd.crosstab(training_data["fare_amount"],training_data["pickup_datetime"]))
        print(p)
```

```
In [ ]: # Since 'p' value is greater than 0.05 we approve the null hypothesis i.e it is not correlated to the dependent variable
        # and exclude "pick_datetime" variable
```

```
In [ ]: # dimension reduction i.e deleting variables with are not correlated to target variable and have no impact on output
        training_data_deleted = training_data.drop(["pickup_datetime", "passenger_count"], axis = 1)
```

```
In [ ]: ## Normality Check
        %matplotlib inline
        plt.hist(training_data["pickup_longitude"],bins = "auto")
```

```
In [ ]: # From the histogram we decide to scale the data using standardisation
        # Save numeric names
        cnames = ["pickup_latitude", "pickup_longitude", "dropoff_latitude", "dropoff_longitude"]
        cnames
```

In [ ]:
```
# Scaling the data using standardisation
for i in cnames:
    print(i)
    training_data_deleted[i] = (training_data_deleted[i] - training_data_deleted[i].mean())/training_data_deleted[i].std()
```

In [ ]:
```
# Using Decision Tree for training and testing data
# dividing into train and test data by 80-20
train, test = train_test_split(training_data_deleted, test_size = 0.2)
```

In [ ]:
```
# Decision tree regression
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(train.iloc[:,2:5],train.iloc[:,1])
```

In [ ]:
```
# Apply model on test data
predictions_DT = fit_DT.predict(test.iloc[:,2:5])
```

In [ ]:
```
# Calculation of Error
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred)/y_true))
    return mape
```

In [ ]:
```
MAPE(test.iloc[:,1], predictions_DT)
```

In [ ]:
```
# Error rate = 4.09%
# Accuracy = 95.91%
```