Cab Fare Prediction

Abstraction:

In this project we need to predict the fare_amount of the cab using attributes like drop -off latitude, drop-off longitude, pickup date-time, passenger_count, pickup longitude, pickup latitude. Our project helps the organisation to take the right decisions, so that the organisation may run successfully without any losses.

Here we used R studio and Jupyter Notebook as platforms to work on the project. At first, we need to clean data using Exploratory Data Analysis like check for Missing Values, Outliners, then Feature Extraction, Feature Selection, Feature Scaling. Achieving the goal was quite challenging. Using the help of Machine Learning Algorithms like Multi Linear Regression, Random Forest, Decision Tree, enhance the probability of reaching goal early.

Project Index

1. INTRODUCTION:

Cab rental system is becoming a new frontier in business, particularly in major cities all over the world. They are many cab rental organisations who are competing with one another in race to achieve profit and fame.

In this project, our objective is to predict the cab fare-amount. The test data contains 16067 observations and 7 variables i.e fare-amount which is our target variable, pickup latitude, pickup longitude, drop-off latitude, drop-off longitude, passenger-count, pickup date-time.

Our aim is to develop a model that helps in predicting the fare amount, for future references using the past data.

2. 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.

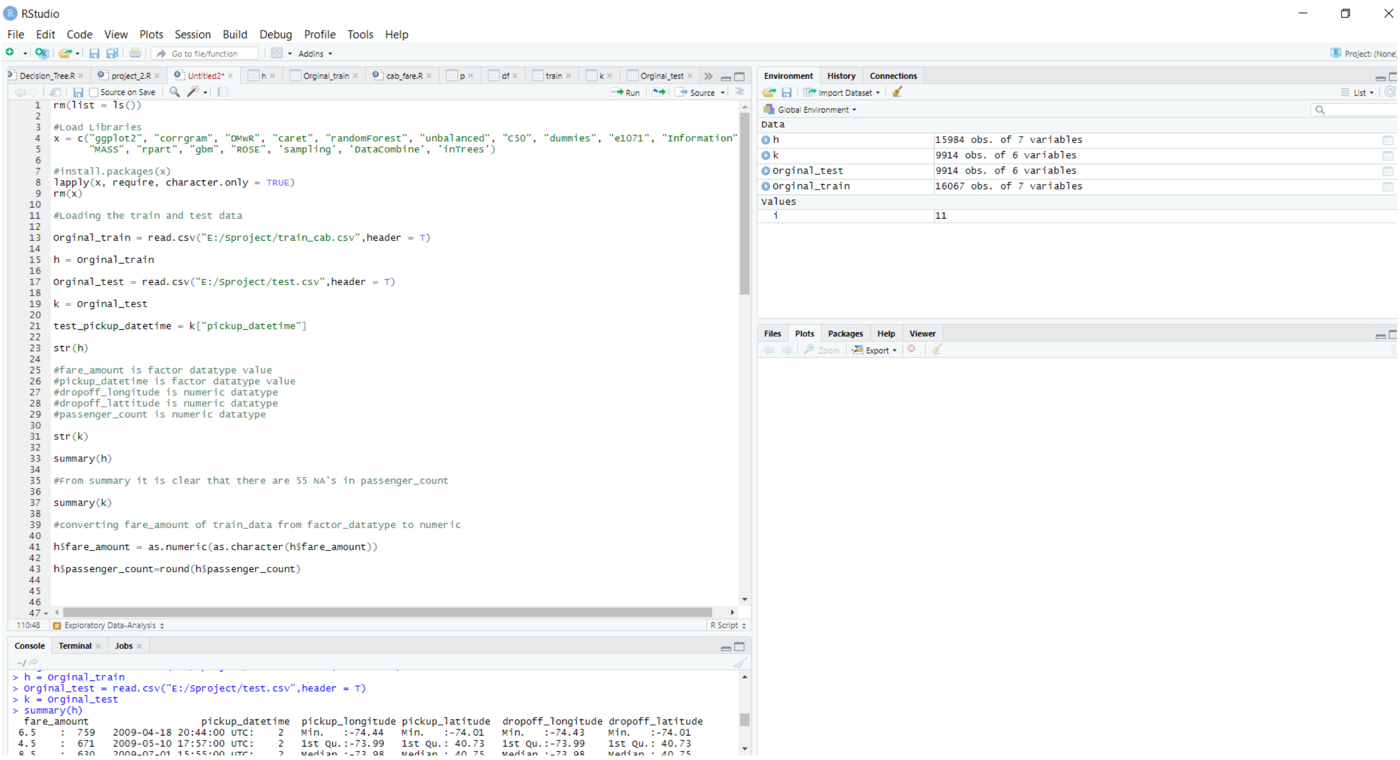
Number of attributes:

- 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.

3. Loading Data in R:

Here we are using R studio and we are loading data into R- environment. Using following command:

- Orginal_train = read.csv("E:/Sproject/train_cab.csv",header = T)
- Orginal_test = read.csv("E:/Sproject/test.csv",header = T)



We also need to load the required packages like ggplot2, corrgram, DMwR, caret, randomForest, unbalanced, C50, dummies, e1071, Information,

4. Data Processing / Exploratory Data Analysis:

I. Fare amount has some negative values. it is also having 0 values, so we need to remove these fields. We used following code to overcome this:

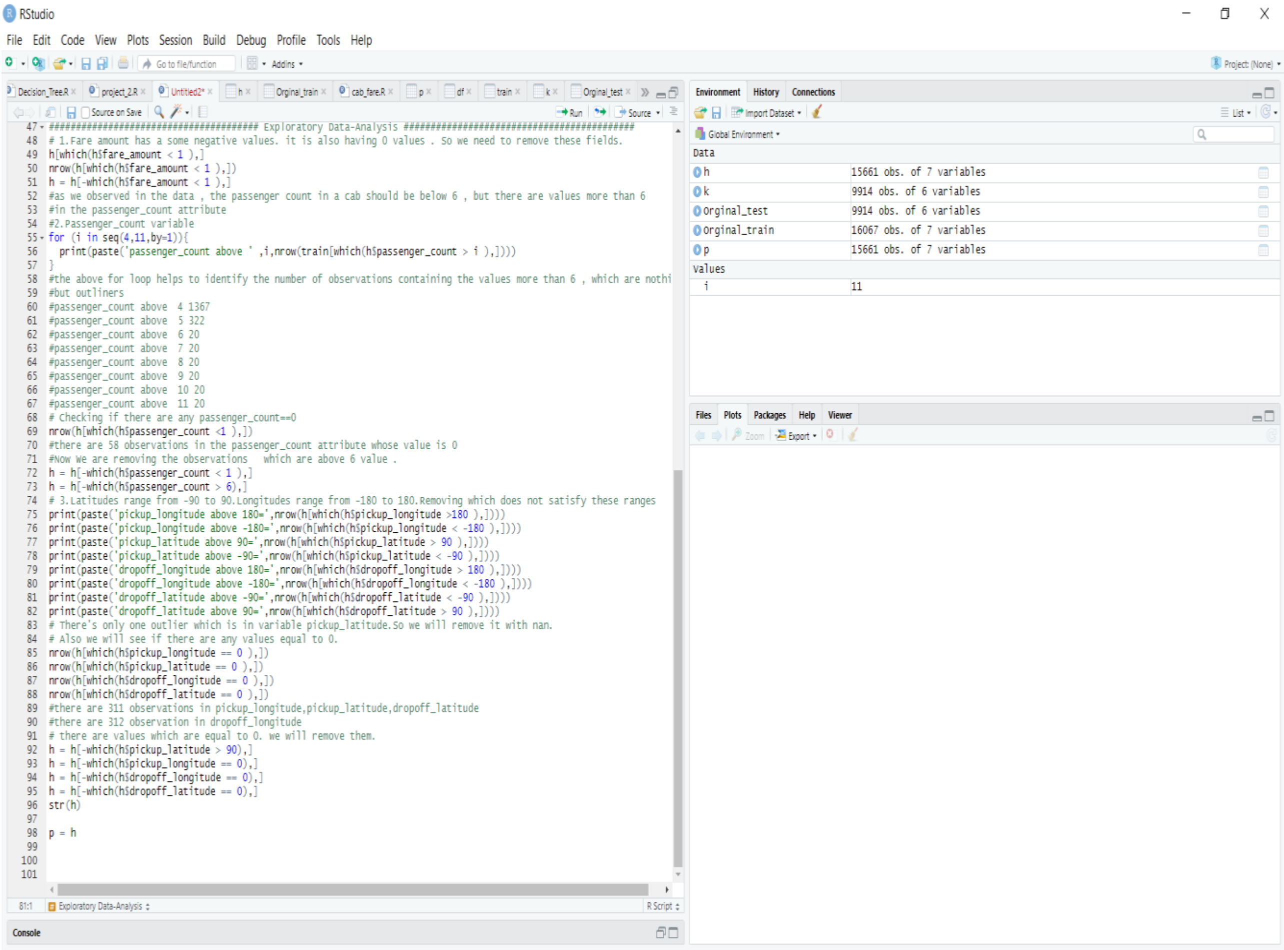
```
h = h[-which(h$fare_amount < 1 ),]
```

II. The passenger count in a cab should be below 6, but there are values more than 6 values.

```
h = h[-which(h$passenger_count < 1 ),]
h = h[-which(h$passenger_count > 6),]
```

III. Latitudes range from -90 to 90. Longitudes range from -180 to 180. Removing the values does not satisfy these ranges.

```
h= h[-which(h$pickup_latitude > 90),]
h= h[-which(h$pickup_longitude == 0),]
h= h[-which(h$dropoff_longitude == 0),]
h= h[-which(h$dropoff_latitude == 0),]
```



5. Missing Value Analysis:

Here we are check for missing values in the dataset like empty rows which was filled with Na. we found some missing values in our dataset. Now missing values can be found by using different techniques like mean, median and Knn imputation.

First, we are selecting 1000 rows randomly and performing mean, median, Knn imputation, so that we can choose any one by comparing actual value and predicted value.

1. For Passenger_count: Actual value = 1, Knn = 1

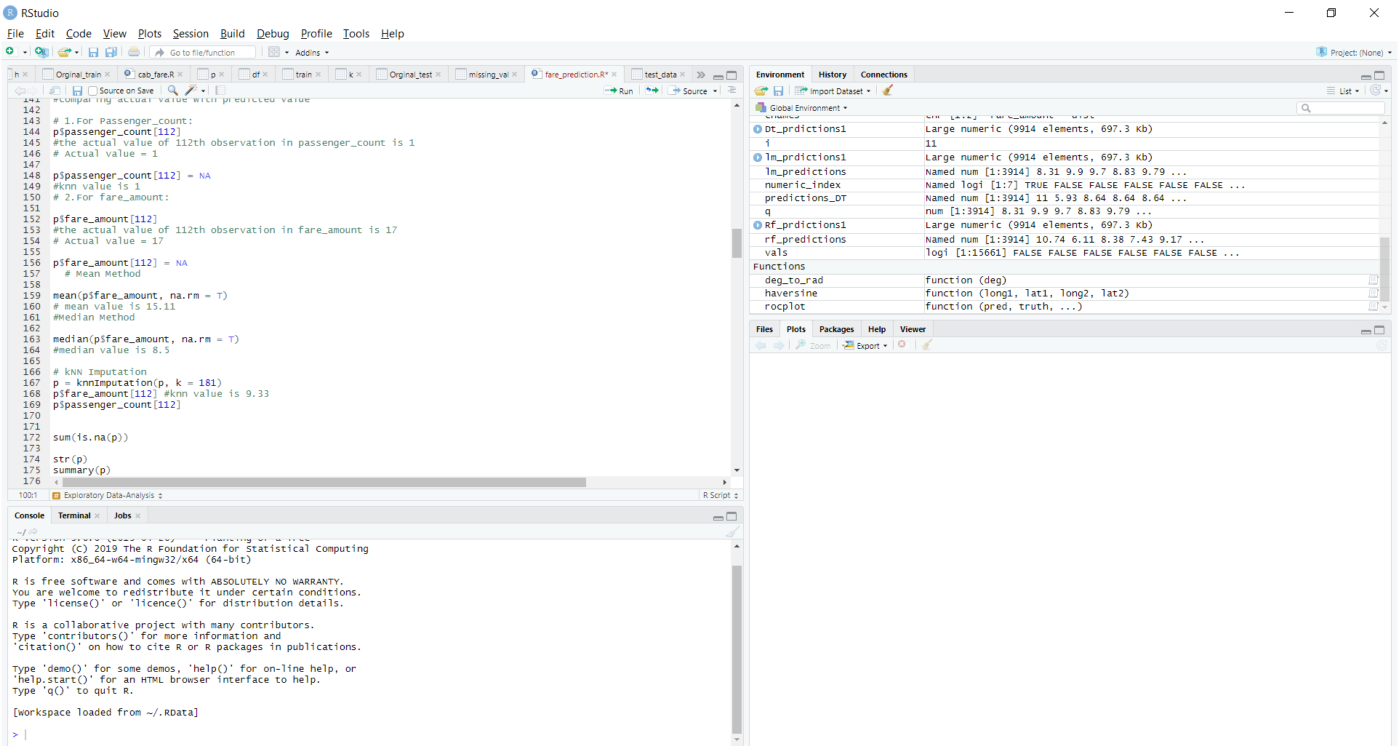
2. For fare_amount:

Actual value = 7.0,

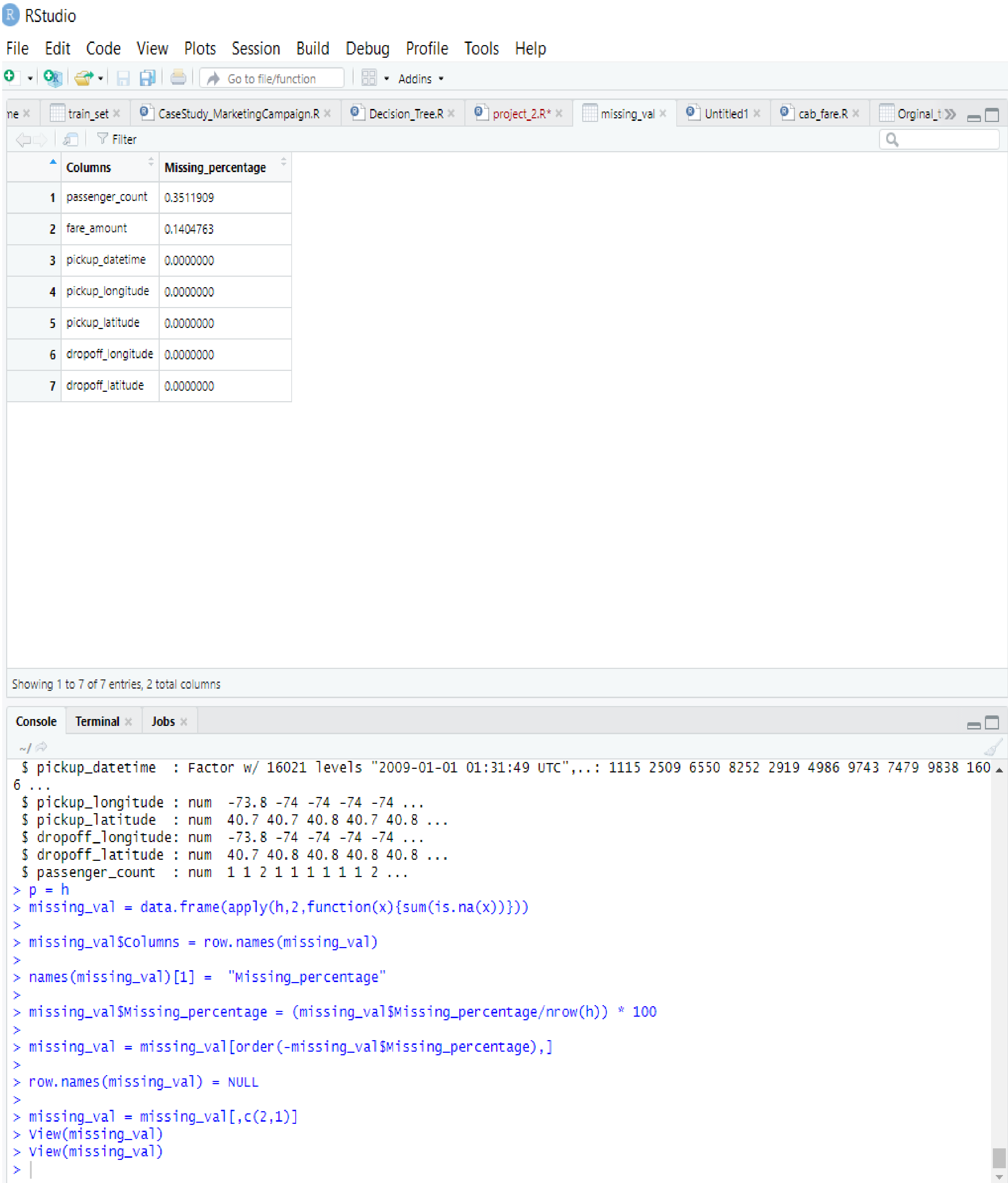
Mean = 15.117,

Median = 8.5,

KNN = 7.369801. So, we choose Knn.



After analysis it is decided to choose to use Knn imputation. After checking here are the missing values percentage of every variable.



As passenger-count variable has 34% of missing values & fare-amount has 14% of missing values.

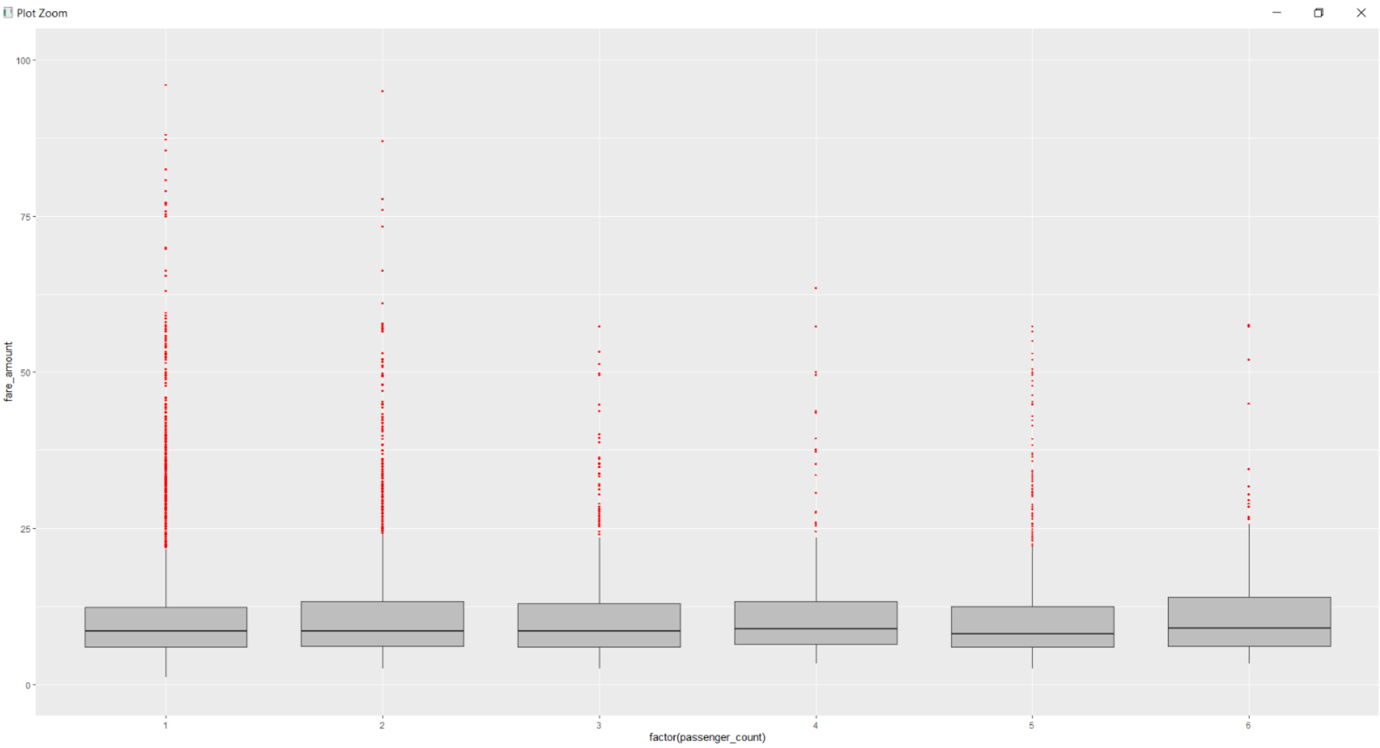
After conducting Knn imputation, all missing values are replaced with k nearest neighbour.

6. Outliner Analysis:

Here we are performing **Outliner Analysis** only on fare-amount which is our dependent variable.

Using the below code, we are representing the outliner graphically.

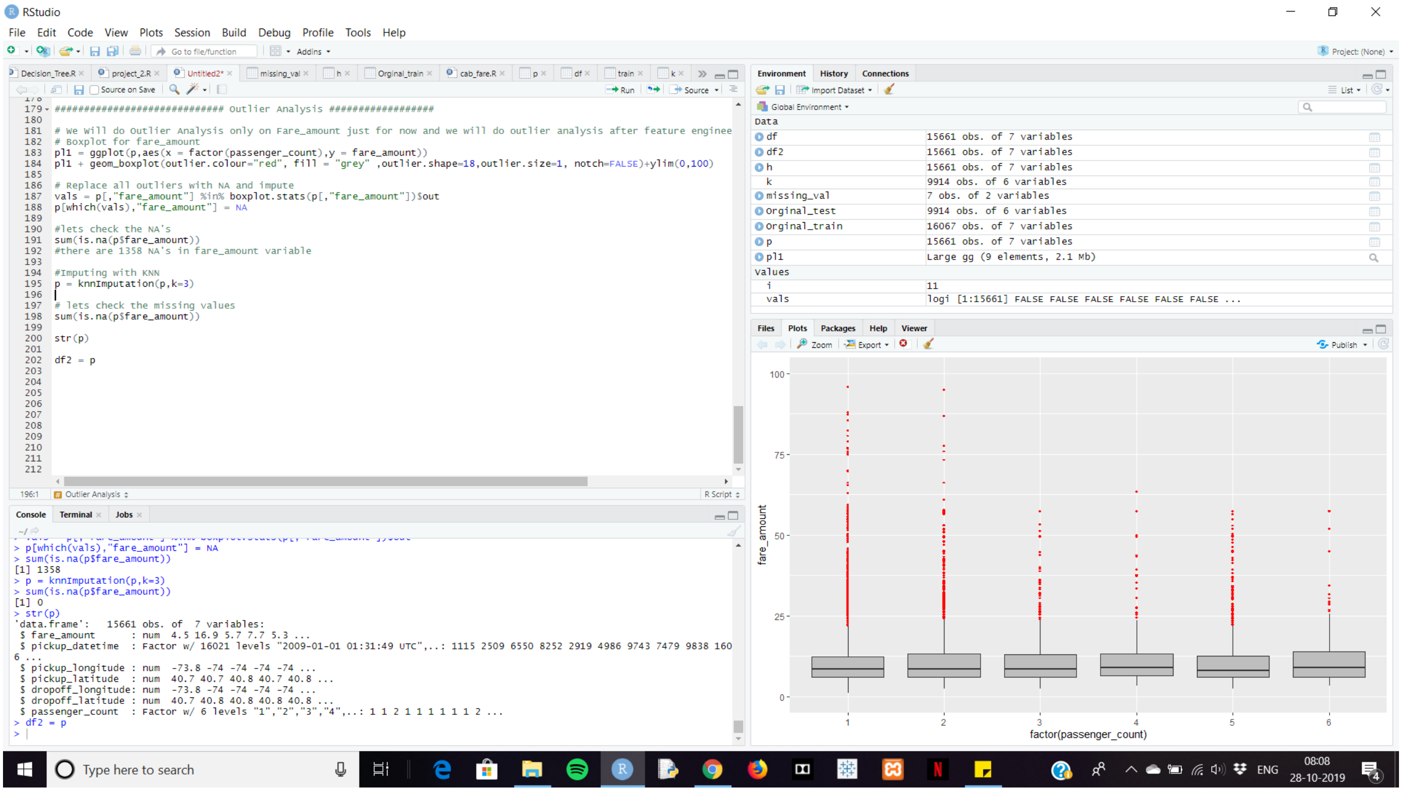
```
pl1 = ggplot(p,aes(x = factor(passenger_count),y = fare_amount))
pl1 + geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,outlier.size=1, notch=FALSE)+ylim(0,100)c
```



We have found 1358 outliners in fare-amount variable.

We have successfully replaced them with NA and removed them using KNN imputation.

```
vals = p[,"fare_amount"] %in% boxplot.stats(p[,"fare_amount"])$out
p[which(vals),"fare_amount"] = NA
```



7. Feature Extraction:

As we observed the given data is containing is a time-stamp variable called **pickup date-time**. We create new variables like year, month, day from the existing feature.

We extract the year, month, day, hour using following commands.

p$pickup_date = as.Date(as.character(p$pickup_datetime))

p$pickup_weekday = as.factor(format(p$pickup_date,"%u"))

p$pickup_mnth = as.factor(format(p$pickup_date,"%m"))

p$pickup_yr = as.factor(format(p$pickup_date,"%Y"))

pickup_time = strptime(p$pickup_datetime,"%Y-%m-%d %H:%M:%S")

p$pickup_hour = as.factor(format(pickup_time,"%H"))

We need to calculate the distance using longitudes and latitudes. We use following commands to calculate the distance.

```
deg_to_rad = function(deg){
  (deg * pi) / 180
}
haversine = function(long1,lat1,long2,lat2){
  #long1rad = deg_to_rad(long1)
  phi1 = deg_to_rad(lat1)
  #long2rad = deg_to_rad(long2)
  phi2 = deg_to_rad(lat2)
  delphi = deg_to_rad(lat2 - lat1)
  dellamda = deg_to_rad(long2 - long1)

  a = sin(delphi/2) * sin(delphi/2) + cos(phi1) * cos(phi2) *
    sin(dellamda/2) * sin(dellamda/2)

  c = 2 * atan2(sqrt(a),sqrt(1-a))
  R = 6371e3
  R * c / 1000 #1000 is used to convert to meters
}
```

We created a function called haversine to calculate distance.

```
p$dist = haversine(p$pickup_longitude,p$pickup_latitude,p$dropoff_longitude,p$dropoff_latitude)
```

Now we are removing variables that are not useful.

p = subset(p,select = -c(pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude))

8. Feature Selection:

In this stage we select the variables which are used for target variable prediction. The variables which are not relevant can be removed, by doing so, we can create model which hold high accuracy of data prediction.

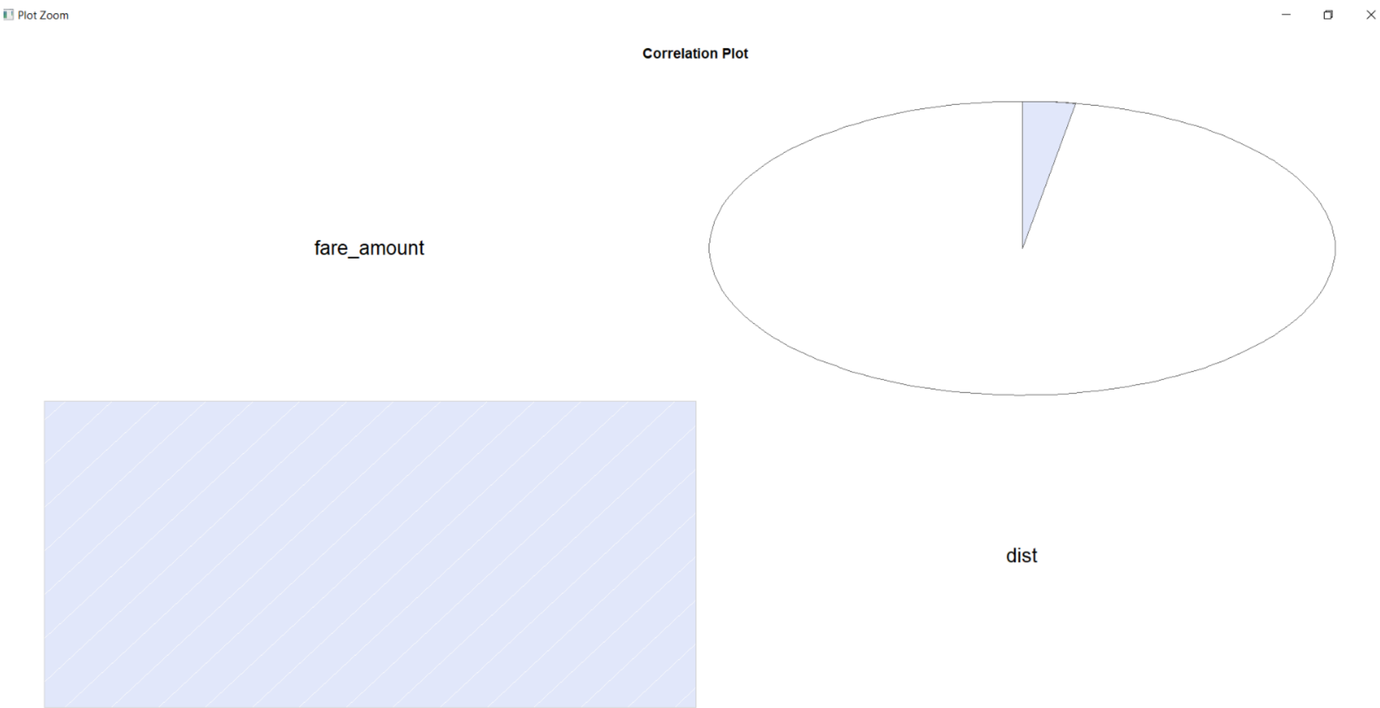
As our dataset contains both categorical and numerical variables. We use **Correlation** for numeric data and **Anova & Chi-square test** for categorical data.

We conduct correlation on numerical data i.e fare-amount and passenger-count.

Correlation Analysis: It helps to find the correlation between two independent variables. The correlation between independent variable and dependent variable must be high. If two independent variables are correlated to each other, then we need to choose any one of it.

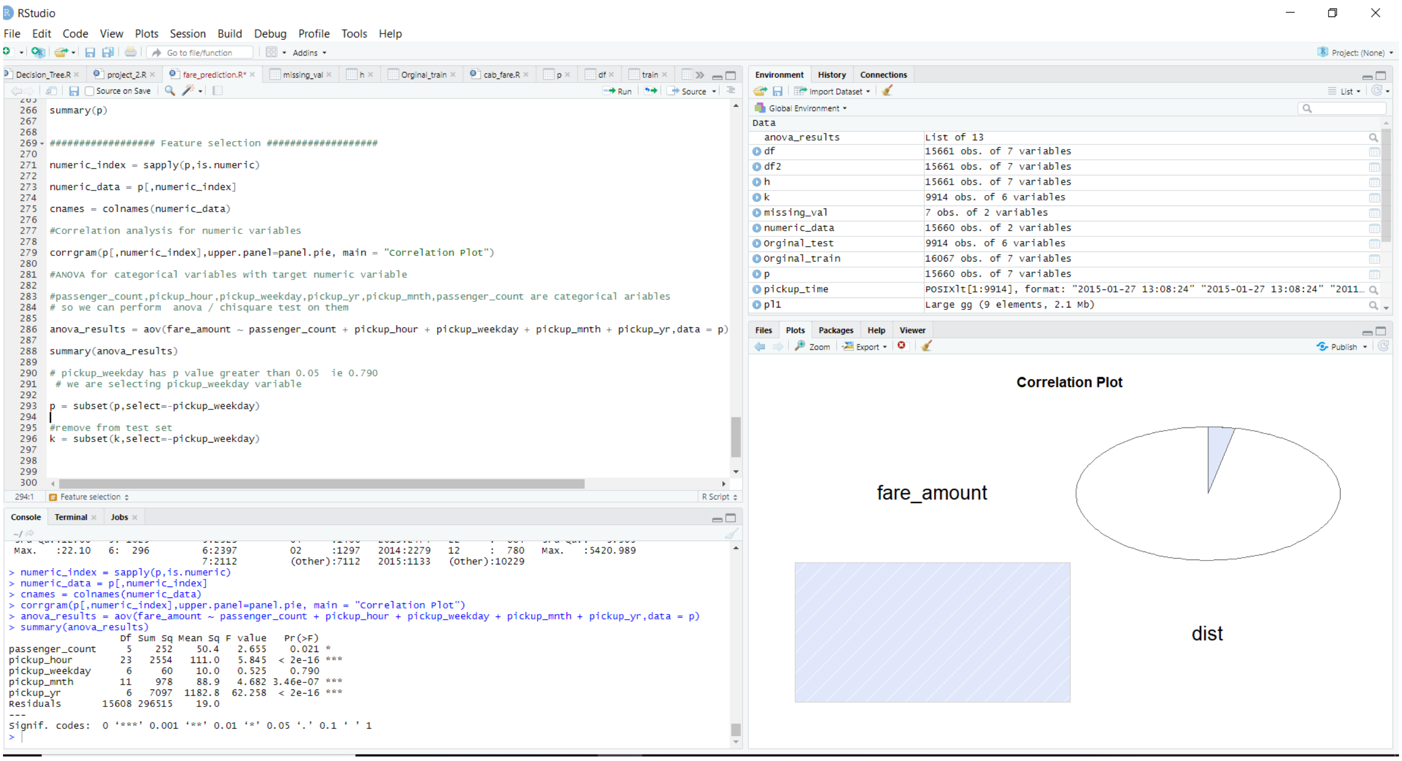
We correlate numeric variables using following command:

corrgram(p[,numeric_index],upper.panel=panel.pie, main = "Correlation Plot")



Anova: Here for categorical variables, we are using anova. If p-value is greater than 0.05 then we accept our Null hypothesis saying that two variables are independent. If p-values is less than 0.05, we reject the Null hypothesis saying that two variables are dependent. We using following command to conduct Anova test on categorical variables.

anova_results = aov(fare_amount ~ passenger_count + pickup_hour + pickup_weekday + pickup_mnth + pickup_yr,data = p)



9. Feature Scaling:

It helps in scaling/measuring data on same units. As, we are aware that different dataset contains different observations of different units, in order to scale them on same units, we use scaling.

Normalisation: It is calculated by dividing the data by its length. It ranges from 0 to 1.

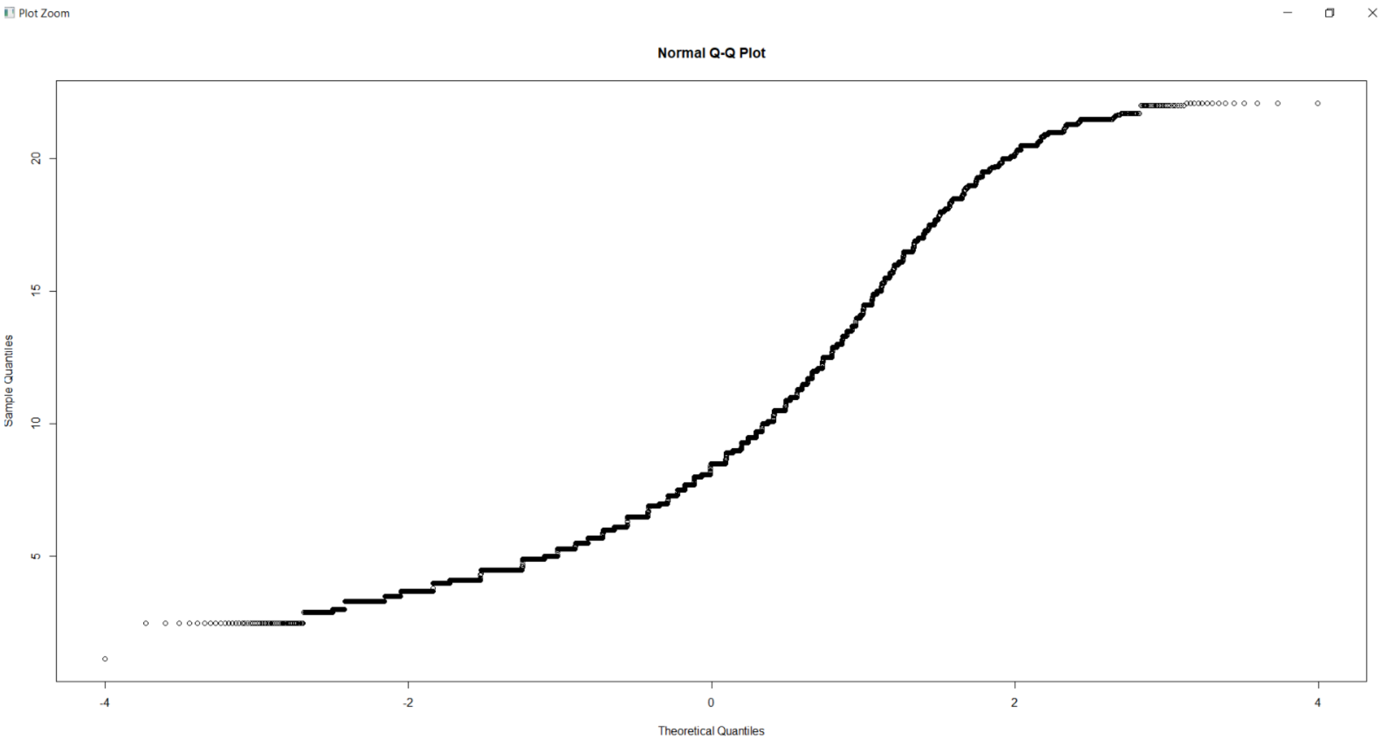
It is checked by using following command:

print('dist')

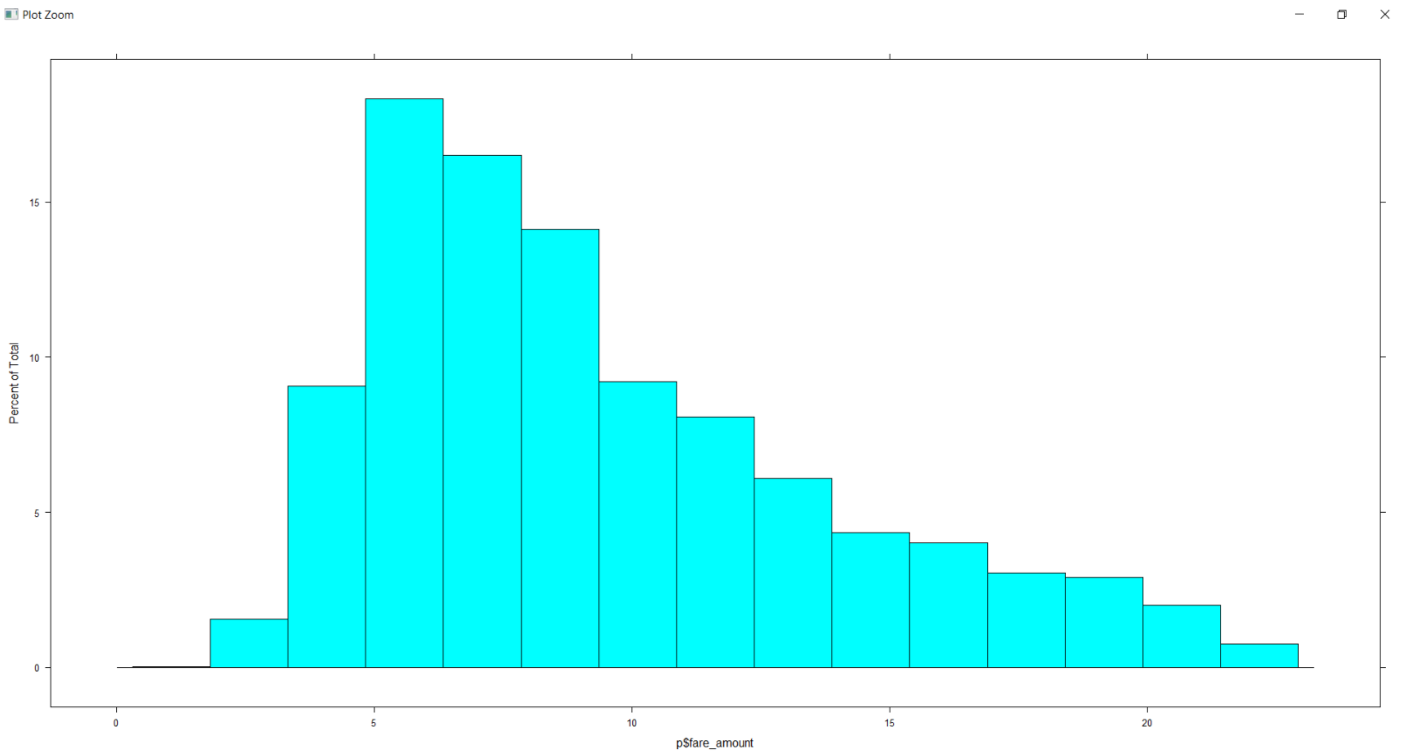
p[,'dist'] = (p[,'dist'] - min(p[,'dist']))/

(max(p[,'dist'] - min(p[,'dist'])))

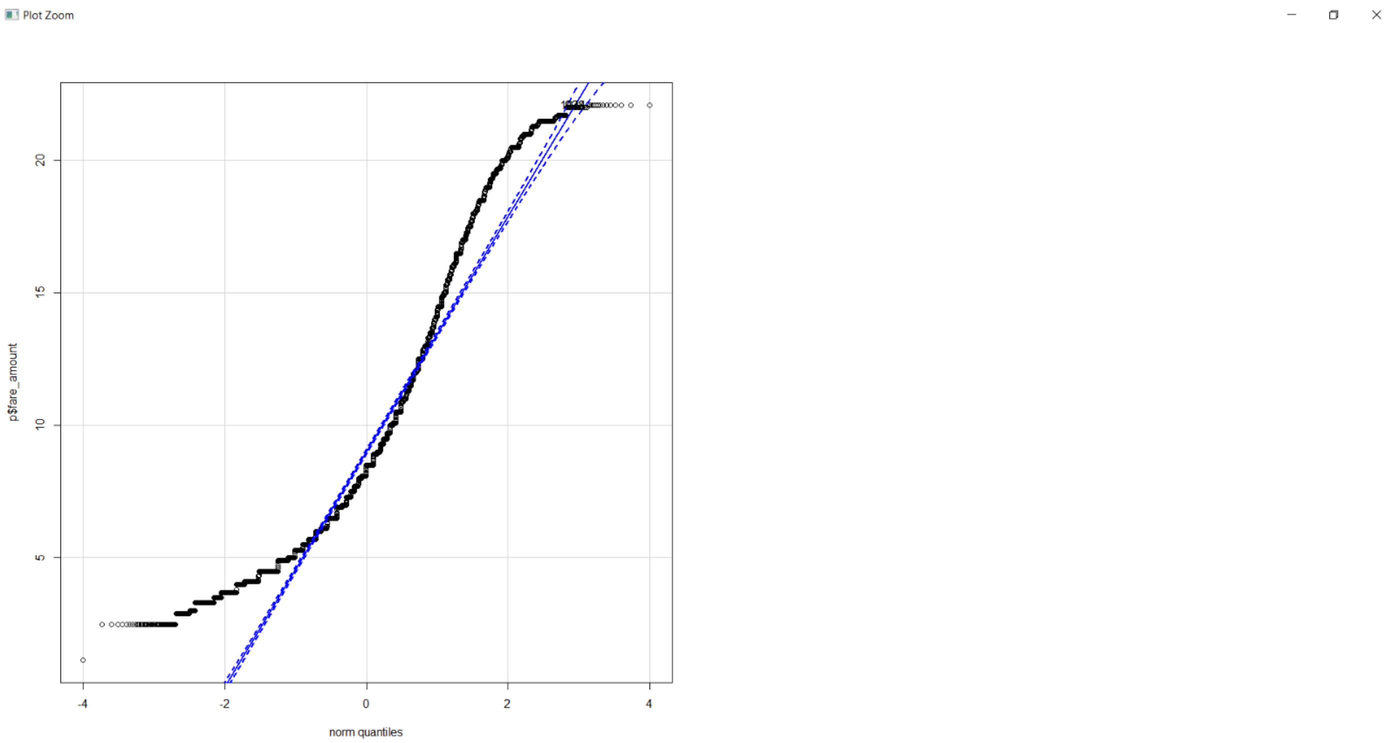
Normality check for dependent variable fare-amount:

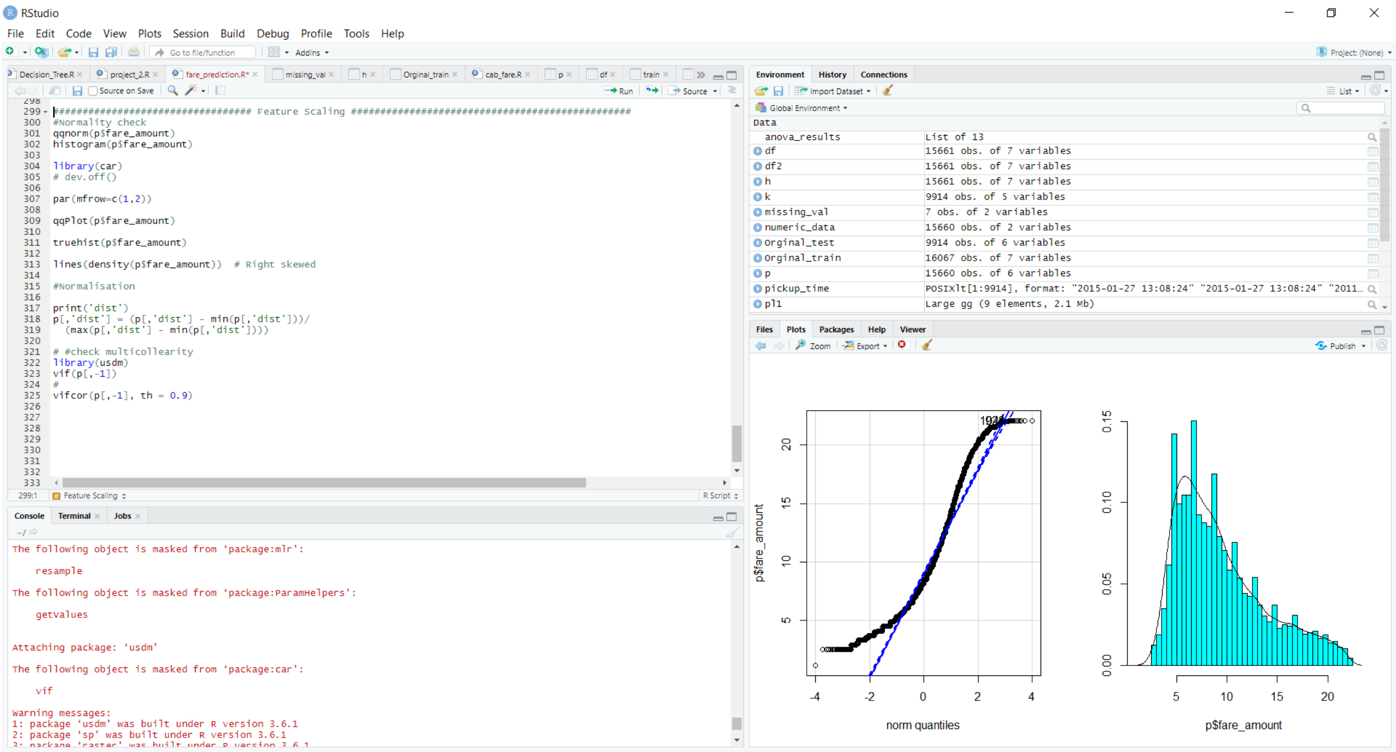


Histogram on dependent variable:



Norm qualities on dependent variable:





10. Splitting data into train and test:

Now before creating model, we need to split the data into train and test data. Here train data has 75% of original train data and test data has 25% of original train data. Using following command:

train_index = createDataPartition(p$fare_amount,p=0.75,list = FALSE) #

train_data = p[train_index,]

test_data = p[-train_index,]

p = original train data

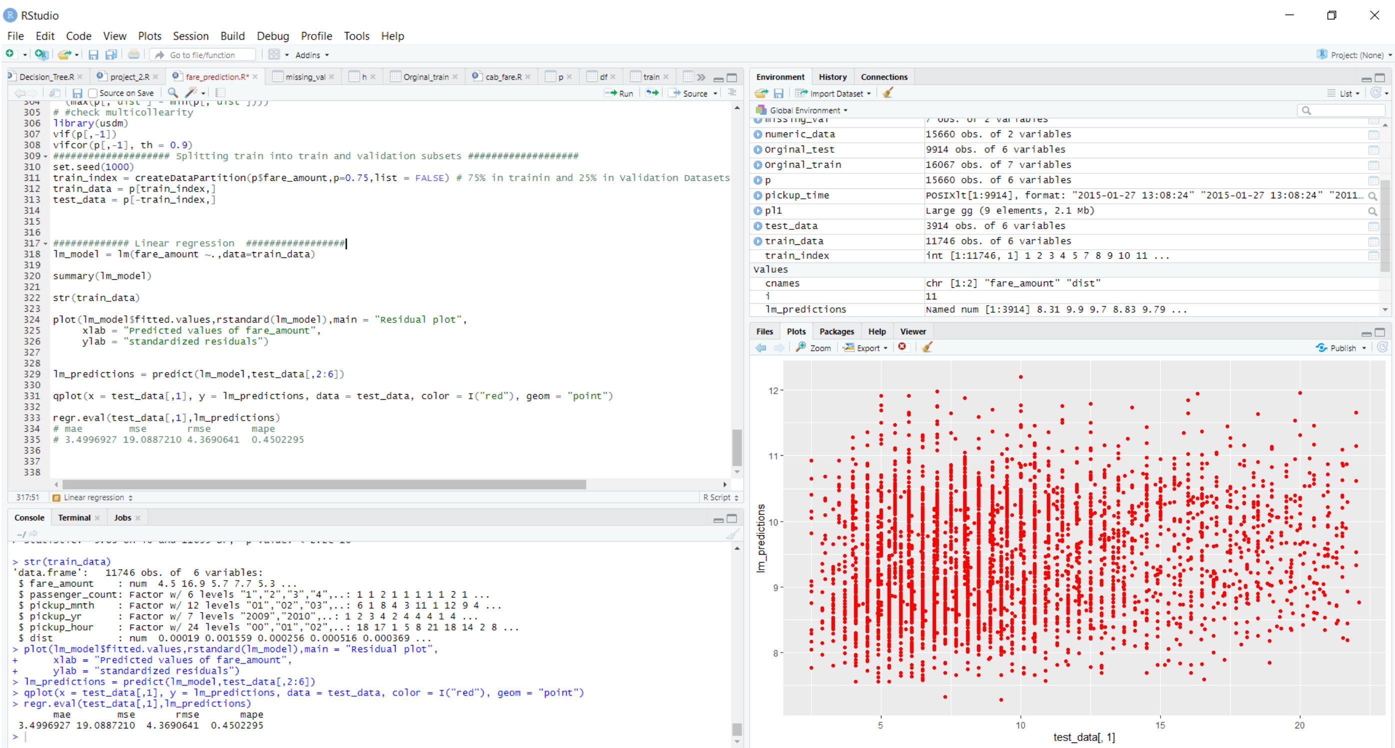
11. Model Development:

I. Using Multiple Linear Regression:

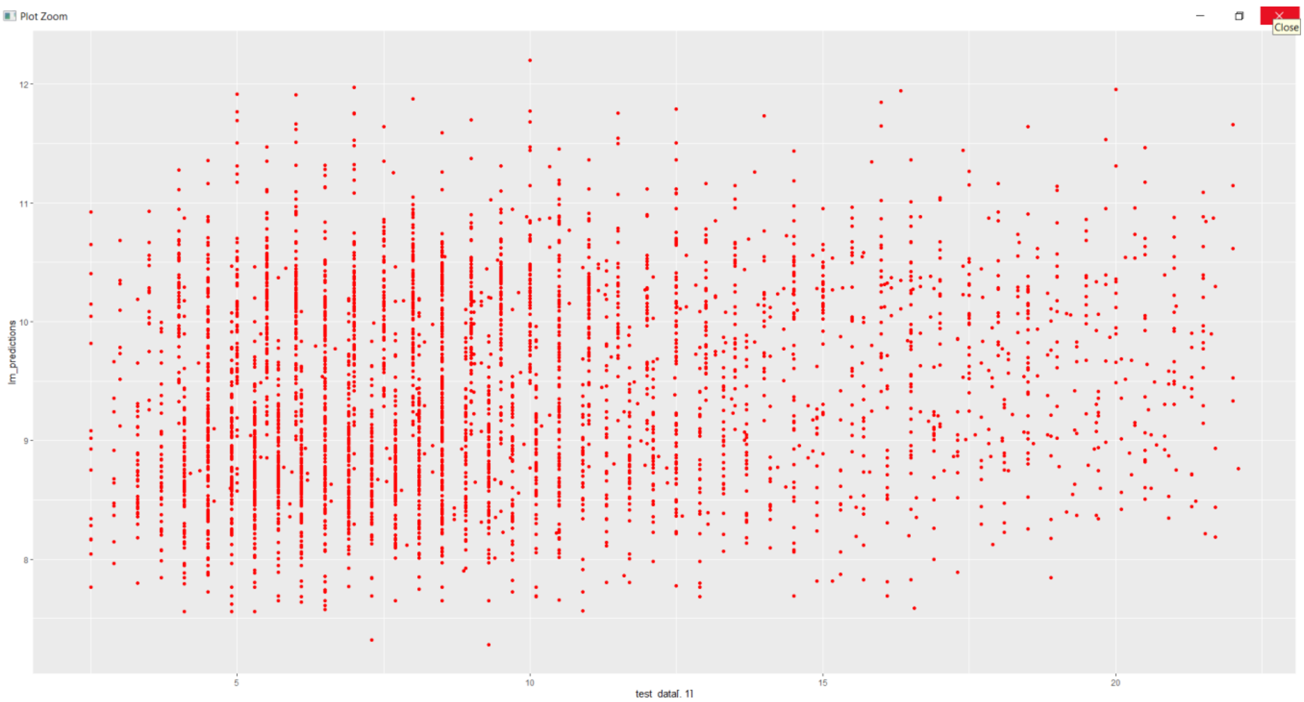
When we use Multiple Linear Regression, we got following results, while we perform error metrices.

MAE	MSE	RMSE	MAPE
3.4996927	19.0887210	4.3690641	0.4502295

- Here we are taking RMSE (Root mean square error) into consider as we are dealing with time-series.
- In time series we don't measure using MAPE.
- The lower, the value of RMSE, the better, the model will be.



Residual plot for Linear Regression:

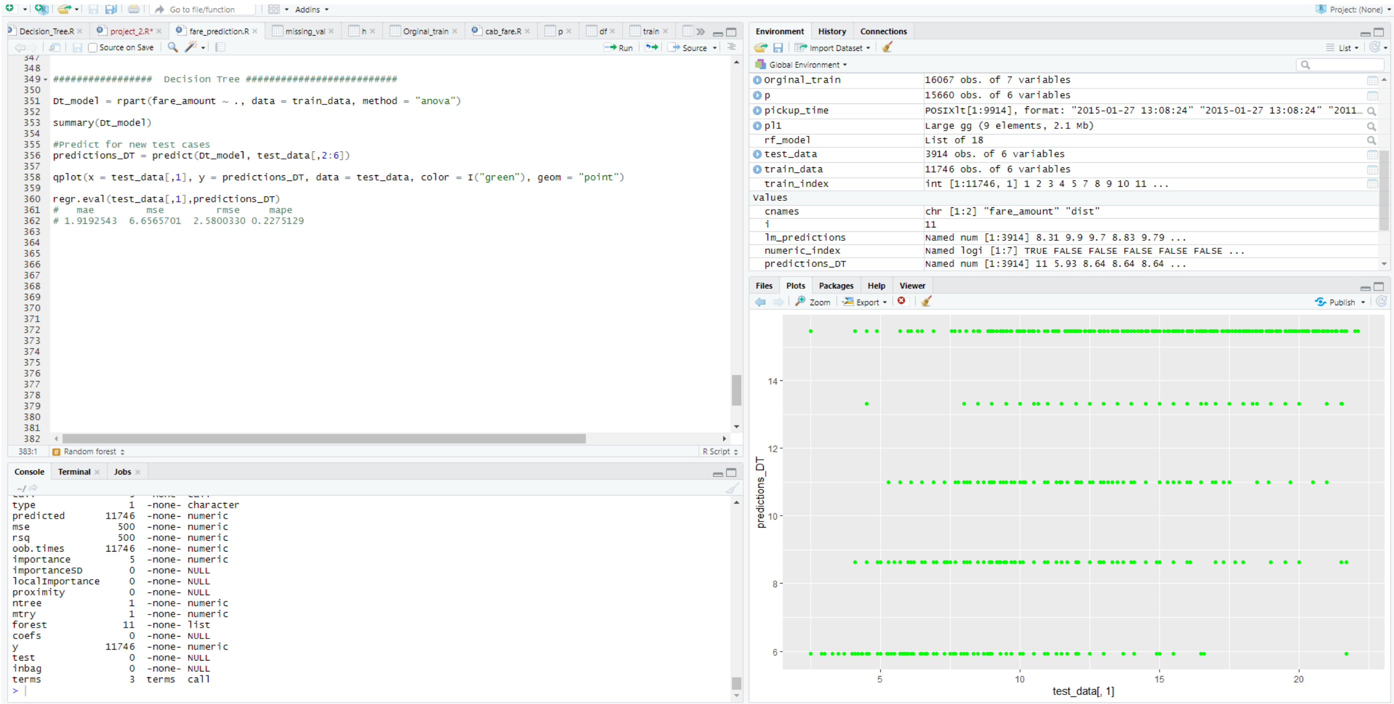


II. Using Decision Tree:

When we use Decision Tree, we got following results, while we perform error metrices.

MAE	MSE	RMSE	MAPE
1.9192543	6.6565701	2.5800330	0.2275129

Here the RMSE for Decision Tree is lower than for Linear Regression, which is good.



Residual plot for Decision Tree:

III. Using Random Forest:

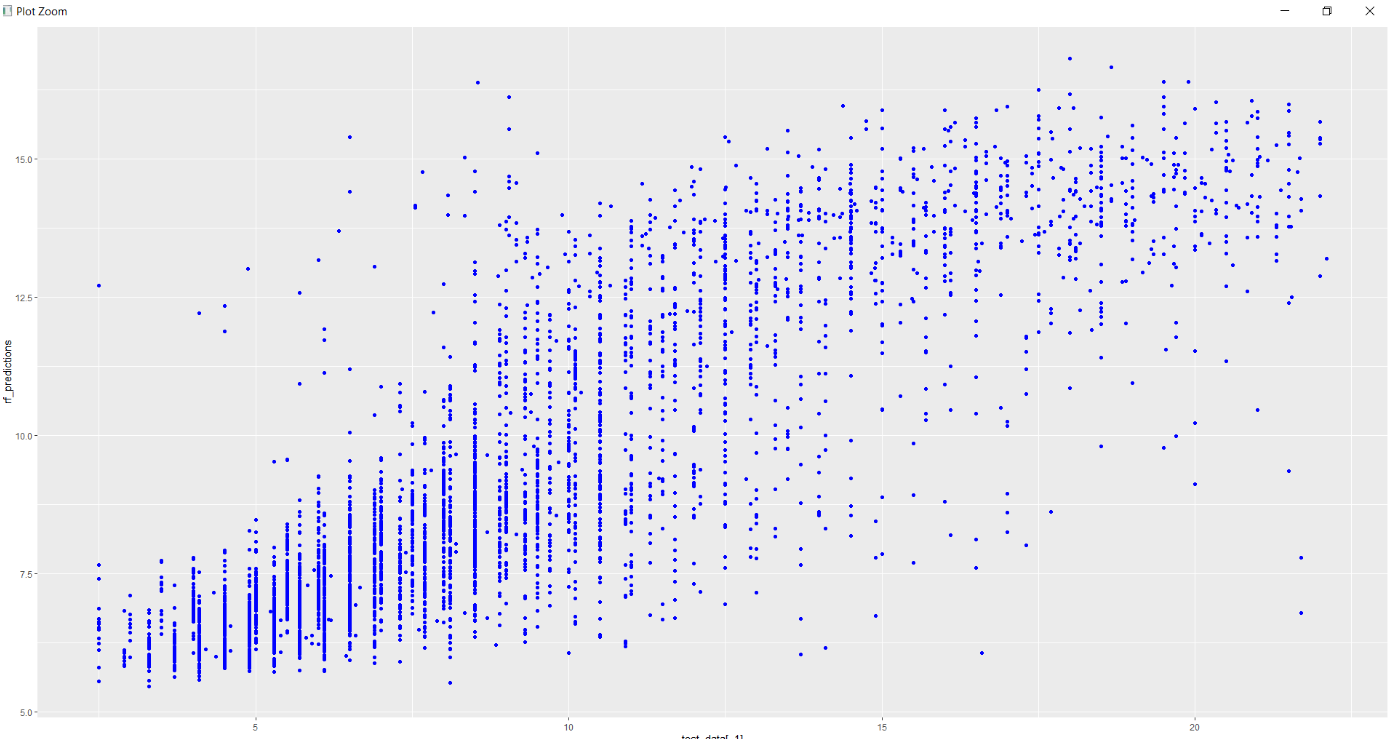
When we use Random Forest, we get the following results. Here what we get when we perform error metrices.

MAE	MSE	RMSE	MAPE
1.8898471	6.3913687	2.5281156	0.2326638

Here we get RMSE value, which is less than Decision tree and multiple linear Regression.

So, we use Random Forest to build our Model.

Residual plot for Random Forest:



Using Model on original Test data:

As RandomForest Model has least RMSE value, we choose model built using RandomForest algorithm. We use following commands to predict the test data.

Rf_prdictions1 = predict(rf_model, k)

- Here k is our test-data, rf_model is our Model developed by RandomForest Algorithm.

Rf_result = data.frame(Rf_prdictions1)

write.csv(DT_result,"E:/Sproject/Random_forest_fare_amount_prediction.csv", row.names = F)

12. Conclusion:

The overall Project is quite challenging, as it takes so much time to sort out the variables and extract the date (year, month, hour, day) from pickup date-time and pickup, dropoff -longitudes and latitudes which helps in calculating distance.

The model we developed can be used for future purpose to predict the cab fare-amount. All we need is to enter the valid data belonging to respected variable.