**CAB FARE PREDICTION**

**Aravind Satheesh**

**28 June 2019**

**Contents**

**1 Introduction** **3**

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

1.2 Data…………………………………………………………………………………………….3

**2 Methodology** **5**

2.1 Pre Processing………………………………………………………………………………… 5

2.1.1 Missing Value Analysis……………………………………………………………………5

2.1.2 Outlier Analysis……………………………………………………………………………6

2.1.3 Feature Selection………………………………………………………………………….10

2.1.4 Feature Scaling……………………………………………………………………………12

2.2 Modeling………………………………………………………………………………………14

2.2.1 Model Selection…………………………………………………………………………...14

2.2.2 Model Evaluation………………………………………………………………………….15

2.2.3 Linear Regression...……………………………………………………………………..…17

2.2.3.1 Implementation...……………………………………………………………………....17

2.2.4 Decision Tree……………………………………………………………………………....19

2.2.4.1 Implementation………………………………………………………………………...20

2.2.5 Random Forest …………………………………………………………………………….21

2.2.5.1 Implementation…..………………………………………………………………….…22

**3 Conclusion** **24**

3.1 Model Selection………………………………………………………………………………..24

3.2 Output………………………………………………………………………………………….25

**Appendix A - Extra Figures** **26**

**Appendix B - Code** **29**

Complete Python Code…………………………………………………………………………….34

Complete R Code…………………………………………………………………………………..42

**References**  **47**

**Chapter 1**

**Introduction**

* 1. **Problem Statement**

Our cab rental start-up company have successfully run the pilot project and now we want to launch our cab service across the country. We have collected the historical data from our pilot project and now 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.

* 1. **Data**

In this project, our task is to predict the fare amount for a cab ride in the city on a particular day based on the historical data. The sample data given below is a sample from the whole population which is used to predict the fare amount for a cab ride:

**Table 1.1 Cab Fare Prediction Sample Train Data (Columns1-7)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **fare\_amount** | **pickup\_datetime** | **pickup\_longitude** | **pickup\_latitude** | **dropoff\_longitude** | **dropoff\_latitude** | **passenger\_count** |
| 4.5 | 2009-06-15 17:26:21 UTC | -73.844311 | 40.721319 | -73.84161 | 40.712278 | 1 |
| 16.9 | 2010-01-05 16:52:16 UTC | -74.016048 | 40.711303 | -73.979268 | 40.782004 | 1 |
| 5.7 | 2011-08-18 00:35:00 UTC | -73.982738 | 40.76127 | -73.991242 | 40.750562 | 2 |
| 7.7 | 2012-04-21 04:30:42 UTC | -73.98713 | 40.733143 | -73.991567 | 40.758092 | 1 |
| 5.3 | 2010-03-09 07:51:00 UTC | -73.968095 | 40.768008 | -73.956655 | 40.783762 | 1 |

From the given data from the **train dataset**, we have to predict the variable, **‘fare\_amount’**, in the **test dataset**, using the pickup timestamp, latitude and longitude details and the data on total number of passengers travelling. The complete structure of the data is as follows:

**Table 1.2 Cab Fare Prediction Dataset Structure**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Description** | **Predictor/Response** |
| **pickup\_datetime** | Timestamp value indicating when the cab ride started in GMT | Predictor |
| **pickup\_longitude** | Float value of longitude coordinate of where the cab ride started. | Predictor |
| **pickup\_latitude** | Float value of latitude coordinate of where the cab ride started. | Predictor |
| **dropoff\_longitude** | Float value of longitude coordinate of where the cab ride ended. | Predictor |
| **dropoff\_latitude** | Float value of latitude coordinate of where the cab ride ended. | Predictor |
| **passenger\_count** | An integer indicating the number of passengers in the cab ride. | Predictor |
| **fare\_amount** | The total cost of cab fare of the trip | Response |

As we have looked into the dataset, using these variables, we have to predict the fare\_amount. But instead of using these variables directly, we can derive few other variables from these variables to predict the fare amount effectively. Generally, in our day to day life, the fare\_amount varies depending on various factors, but basically it depends upon the distance covered and the number of passengers travelled.

So using the latitude and longitude details, we can measure the **distance covered** and it can be used for effectively predicting the fare amount. Additionally, we can even derive the time and date data separately from the timestamp data for reference. These can be seen in the data pre-processing section which is being followed.

**Chapter 2**

**Methodology**

* 1. **Pre Processing**

Before entering into the modelling phase, initially we have to analyse the data which we have. We need to clean the data, select the required variables for modelling and analyse the data to impute the missing values; detect and handle the outliers; normalise the data for bringing all the data under the same scale for giving equal weightage for all variables. For performing these functions, we can even visualise the data for easy processing using plots and graphs. This process is termed as **Exploratory Data Analysis (EDA).** In this project, response (dependent) variable is a numerical variable and so this is a **regression** problem. For regression, the data must usually be normally distributed. In the following pages, we will look more about the various pre processing techniques which we carry out in this project.

* + 1. **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. On analysing the data, we find that, there are few missing values in all the variables except pickup\_datetime of the given data. It has been represented in a tabulated format as follows:

**Table 2.1 Before Missing Value Analysis**

|  |  |  |
| --- | --- | --- |
|  | **Variables** | **Missing\_percentage** |
| **0** | **pickup\_longitude** | **1.960540** |
| **1** | **pickup\_latitude** | **1.960540** |
| **2** | **dropoff\_longitude** | **1.954316** |
| **3** | **dropoff\_latitude** | **1.941868** |
| **4** | **passenger\_count** | **0.697081** |
| **5** | **fare\_amount** | **0.161822** |
| **6** | **pickup\_datetime** | **0.000000** |

Generally, the missing value analysis is performed and the variables which are having missing values greater than 30% are dropped as per industry standards. Here we can see that, only less than 2% of data are missing in all columns. We can even drop these rows and proceed, but around 320-350 rows will be dropped and it is better to impute those values. There are 2 methodologies through which these values can be imputed. They are using 'central tendency' or using 'KNN imputation method'. So by trial and error method, we can find the best methodology for this dataset by importing a known value and by checking with the dataset.

Upon doing trial and error of all methods, **‘KNN Imputation Method’** seems to impute the values better. So using ‘KNN Imputation method’, we have performed missing value analysis and have imputed the unknown values. The result is as follows:

**Table 2.2 After Missing Value Analysis**

|  |  |  |
| --- | --- | --- |
|  | **Variables** | **Missing\_percentage** |
| **0** | **pickup\_longitude** | **0.0** |
| **1** | **pickup\_latitude** | **0.0** |
| **2** | **dropoff\_longitude** | **0.0** |
| **3** | **dropoff\_latitude** | **0.0** |
| **4** | **passenger\_count** | **0.0** |
| **5** | **fare\_amount** | **0.0** |
| **6** | **pickup\_datetime** | **0.0** |

Now, as the missing values have completely been handled, we can proceed with the next pre-processing technique.

* + 1. **Outlier Analysis**

The next pre-processing technique that we usually carry out is the **Outlier Analysis**. Now, our data is free of missing values. Now, Outlier can be defined as a data point that is unusually different when compared to the remaining data of the variable. These Outliers need to be handled before the modelling phase because, these may make the model biased to a particular variable and alter the output.

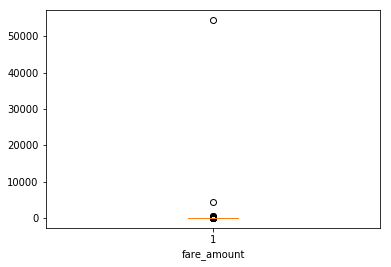
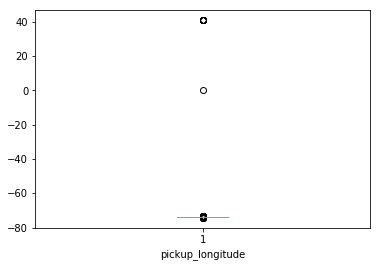
Here, we are going to handle the outliers based on the following steps:

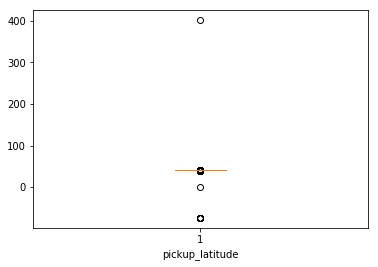
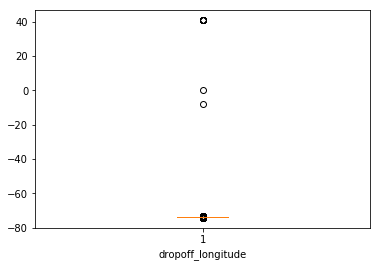
* create boxplots to check for outliers.
* check the correlation between the independent and dependent variables before and after the removal of outliers and handle them accordingly.

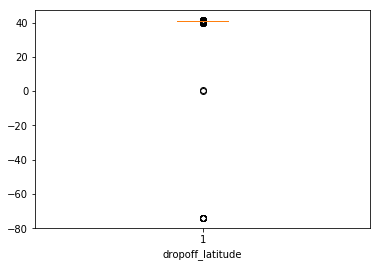
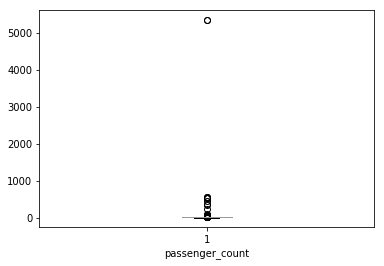
**Note**: Outlier Analysis need to be carried out carefully and it must be made sure that no useul information gets removed due to this.

Now let us look upon the **box and whisker plot** for these variables.

**Fig 2.1 Outlier Analysis - Box and Whisker Plots** (Python and R Code in Appendix B)

Figures 2.1 is the Python plots using matplotlib library. The R plots (Figure A.1), created by using ggplot2 library, is given in the Appenidix A.

**Outliers**

fare\_amount 1397

pickup\_longitude 831

pickup\_latitude 532

dropoff\_longitude 945

dropoff\_latitude 778

passenger\_count 1703

From the box and whisker plots, we can find that, though they give a huge number of data points as outliers, majority of data points are very close to the whisker range and there are only very few points that are very far from the range. So, we must handle these outliers with care and make sure that none of the useful information is removed.

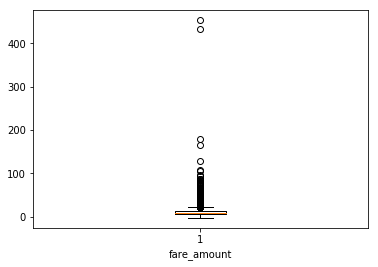
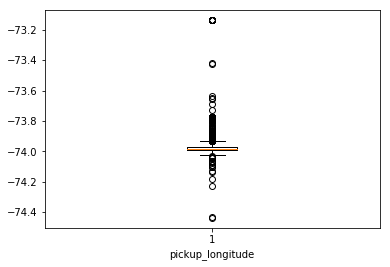
So, let us handle only those data points which are far from the complete set.

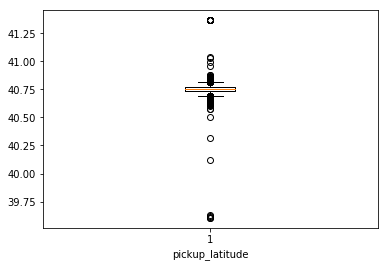
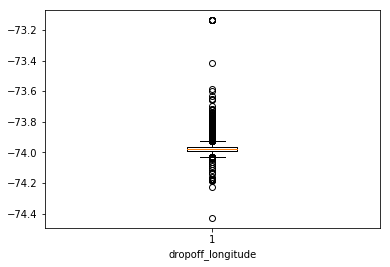
In the variables, **dropoff\_latitude, pickup\_longitude, pickup\_latitude, dropoff\_longitude, fare\_amount,** we can omit only those few data points which are far from the complete range based on the box and whisker plots.

Generally, in a cab, maximum 7 passengers can be boarded at a time. So those values beyond 7 can be omitted in **'passenger\_count'** variable.

After imputation of outlier points, we can take the box nd whisker plot of the resulting data and analyse it.

**Fig 2.2 Box and Whisker Plot(After Outlier analysis)** (Python and R Code in Appendix B)

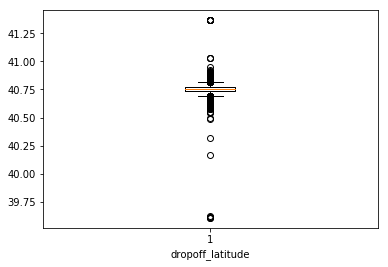
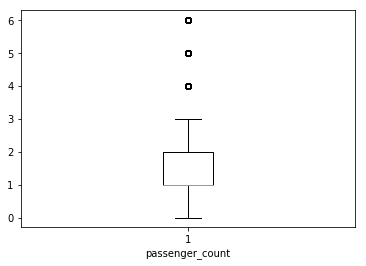
 

Figure 2.2 is the Python plot using matplotlib library. The R plot (Figure A.2), created by using ggplot2 library, is given in the Appenidix A

At this point, instead of handling outliers in the latitude and longitude points, let us convert this latitude and longitude points into distance travelled so that it will be easy to make predictions based on it.

For calculating the distance travelled, let us look upon a formula termed as **‘Haversine formula’.** It gives us approximate distances between the latitude and longitude points.

The haversine formula determines the [great-circle distance](https://en.wikipedia.org/wiki/Great-circle_distance) between two points on a [sphere](https://en.wikipedia.org/wiki/Sphere) given their [longitudes](https://en.wikipedia.org/wiki/Longitude) and [latitudes](https://en.wikipedia.org/wiki/Latitude). Important in [navigation](https://en.wikipedia.org/wiki/Navigation), it is a special case of a more general formula in [spherical trigonometry](https://en.wikipedia.org/wiki/Spherical_trigonometry), the law of haversines, that relates the sides and angles of spherical triangles.

So using haversine formula, the **distance\_travelled** data is fetched and even the time, day, month and year data is fetched from the **pickup\_datetime** variable and the latitude and longitude and timestamp variables are dropped from the dataset.

Now, the final dataset looks similar to this:

|  | **fare\_amount** | **distance\_travelled** | **passenger\_count** | **Year** | **Month** | **Day** | **Hour** | **Time** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 4.5 | 1.030764 | 1 | 2009 | 6 | 15 | 17.0 | 17:26:21 |
| **1** | 16.9 | 4.827250 | 1 | 2010 | 1 | 5 | 16.0 | 16:52:16 |
| **2** | 5.7 | 1.389525 | 2 | 2011 | 8 | 18 | 0.0 | 00:35:00 |
| **3** | 7.7 | 2.799270 | 1 | 2012 | 4 | 21 | 4.0 | 04:30:42 |
| **4** | 5.3 | 1.999157 | 1 | 2010 | 3 | 9 | 7.0 | 07:51:00 |

Now again outlier analysis is carried out in **fare\_amount** and **distance\_travelled** variables and the final box and whisker plot after outlier analysis is as follows:

**Fig 2.3 Final Box and Whisker Plot(After Outlier analysis)** (Python and R Code in Appendix B)

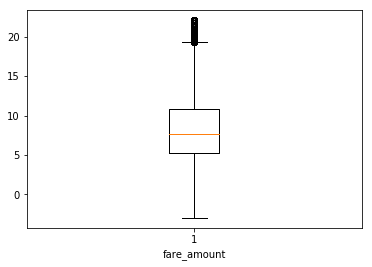
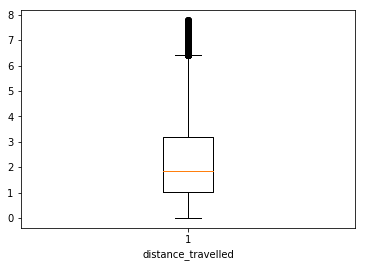
 

Figure 2.3 is the Python plot using matplotlib library. The R plot (Figure A.3), created by using ggplot2 library, is given in the Appenidix A

Here, we can find that, the outliers in 'fare\_amount' and 'distance\_travelled' have been considerably reduced and brought close to the whiskers of the plot. Hence these outliers are not again treated and they are considered as part of data as they wont bias the model.

Now, as the outliers are handled, we can move on to the next pre-processing technique of **Feature Selection.**

* + 1. **Feature Selection**

Machine learning models work on a simple rule – if we put garbage in, we will only get garbage to come out. Here, by garbage, I mean noise in data. This becomes even more important when the number of variables is very large. We need to use only the required number of variables for creating an algorithm, or else, it might lead to **‘overfitting’**. **Overfitting** is a modelling error which occurs when a function is too closely fit to a limited set of data points. At times, when building models, less is better.

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

* The correlation between two independent variables should be less and
* The correlation between Independent and Target variables should be high.

Initially, we will form the correlation matrix and then check the correlation between variables using the heatmap as follows.

**Fig 2.4 HeatMap** (Python and R Code in Appendix B)

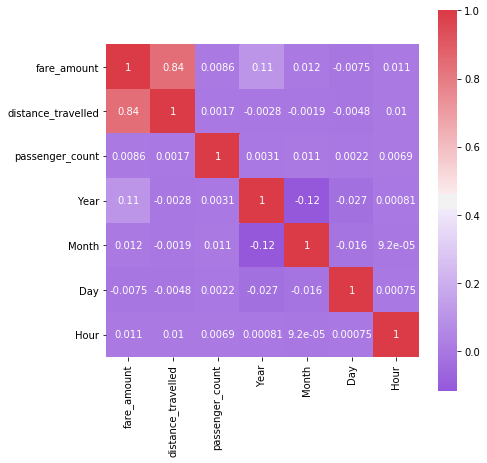
****

Figure 2.4 is the Python plot using **seaborn** library. The R plot (Figure A.4), created by using ggplot2 library, is given in the Appenidix A

From the heatmap, we can get the inference of the following:

* We know that the correlation between Independent and Target variables should be high (i.e., away from zero). Let us keep a cutoff region of (-0.25 to 0.25). The variables having a correlation with ‘casual’, ‘registered’ and ‘cnt’ variables within this region can be removed while proceeding further while modelling as they describe the target variable very less and so they are less important in prediction of target variable. Under this criteria, only **‘distance\_travelled’** variable gets selected as it has high correlation with **‘fare\_amount’.**
* Additionally, we know that the correlation between two Independent variables should be low (i.e., close to zero). Let us keep a cutoff region (>0.5 and <-0.5). The independent variables having a correlation in this region should be removed while going into modelling phase, as one among the two variables is sufficient enough to predict the target variable. This will reduce **overfitting.** Under thiscriteria, we find that none fall under this category.

But, as the variables, ‘passenger\_count’,’Year’,’Month’,’Day’, ‘Hour’ ave limited number of values, they can be considered categorical and when independent variables are categorical and dependent variable is numeric, we can prefer **ANOVA** testing for correlation.

**ANOVA** stands for **Analysis of variance**. It is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

So, a null hypothesis is described as if no dependency exists between two variables. So based on ANOVA test, F-statistic value and p-value are determined. If we have a cutoff of 0.05 for p-value, then if **p-value < 0.05, then null hypothesis is rejected and so dependency exists between the two variables.** So basically, p-value must be close to zero and F-statistic value must be higher for dependency to occur between two variables.

**p-value**

|  |  |
| --- | --- |
| ‘Year’ | 0.00000000e+00 |
| ‘Month’ | 1.31281529e-05 |

Based on ANOVA test**, ‘Year’ and ‘Month’** also prove to be having dependency with the **‘fare\_amount’** and so they are also considered going forward.

As part of **Dimensionality Reduction**, the dataset is proceeded with the variables, **‘distance\_travelled’, ‘Year’, ‘Month’, ‘fare\_amount’** to the next preprocessing section.

* + 1. **Feature Scaling**

**Feature Scaling** typically means to bring all the variable data under a common scale so that none of the variables overweigh during modelling phase. **For example:** If a column value ranges between 0 to 5 and another column value ranges between 1 million to 10 million, then the second variable will overweigh the model during modelling phase. So reduce this effect, **Feature Scaling** is carried out. Generally Normalization and Standardization are the two types of Feature Scaling methods.

**Normalization** typically means that the range of values are **normalized** to be from 0.0 to 1.0.

**Standardization** typically means that the range of values are **standardized** to measure how many standard deviations the value is from its mean.

**Normalization:**  Xchanged=X−Xmin / (Xmax−Xmin)

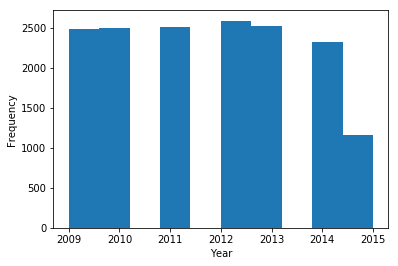
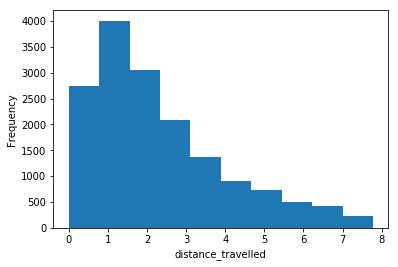
**Standardization:** Xchanged=X−μ / σ

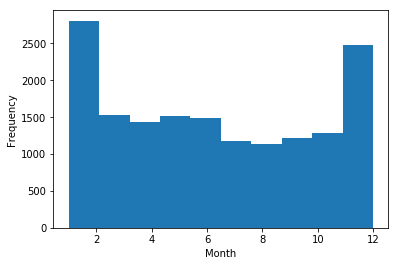
Normally, when the given **data set is uniformly distributed** (i.e., it forms a bell-shaped curve when plotted), we use **standardization**, as it will be easier for us to measure, how many standard deviations away, the particular value is located.

But, when the data set is left skewed or right skewed or **ununiformly distributed** when plotted, we use **Normalization** method, as standardization wont be so efficient as Normalization here due to accumulation of huge data at a particular area.

Here, after Feature Selection; the variables, **‘distance\_travelled’, ‘Year’** and **‘Month’** are to be Normalized as it will be easier for evaluation of models during Modelling phase as all the metrics will be brought under common scale. As the data is not distributed perfectly in uniform, we choose Normalization. The corresponding histograms are again given for reference.

**Fig 2.5(A) Histograms – Feature Scaling** (Python and R Code in Appendix B)





After normalization, the whole of the data will be scaled according to their correspondig ranges. A sample of it is as follows:

**Fig 2.5(B) Data after preprocessing**

|  | **distance\_travelled** | **Year** | **Month** | **fare\_amount** |
| --- | --- | --- | --- | --- |
| **0** | 0.132538 | 0.000000 | 0.454545 | 4.5 |
| **1** | 0.620698 | 0.166667 | 0.000000 | 16.9 |
| **2** | 0.178668 | 0.333333 | 0.636364 | 5.7 |
| **3** | 0.359936 | 0.500000 | 0.272727 | 7.7 |
| **4** | 0.257056 | 0.166667 | 0.181818 | 5.3 |

At this moment, we have undergone the data into various pre-processing techniques such as **Missing Value Analysis, Outlier Analysis, Feature Selection** and **Feature Scaling.** Now, from the raw data, which we got from the dataset, we have come to a point where we have the data made ready to be fed into a machine learning model and train it to make predictions in the future using the same model.

These pre-processing techniques have made it possible to increase the accuracy and reduce the inefficiency of model due to noise (errors) in the data. Now, we can move on to the next phase of Modelling where we have to build a model based on the data and make predictions if data is fed into it in the future.

* 1. **Modeling**

The next phase our project is the Modeling phase. Till now, our data is pre-processed and it is made ready to train the model to make predictions. But initially we must be specific on which model to be used based on our dataset.

* + 1. **Model Selection**

Based on our dataset, we have to decide which model have to be selected for our dataset. For model development, dependent variable may fall under one of the below categories

* Nominal
* Ordinal
* Interval
* Ratio

For our data, dependent variable falls under **interval** category and so, the predictive analysis that we can perform is **Regression** Analysis. There are various algorithms and statistical models for such regression problems. Here we will use the following models for predicting cab fare for a journey.

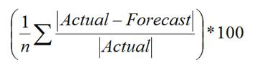
Let us look upon the Regression Algorithms in the order from simple to complex models as follows:

* Linear regression
* Decision Tree
* Random Forest
  + 1. **Model Evaluation**

Generally, for classification problems, we will be having the dependent variable as ‘Yes’ or ‘No’ or say, only two outcomes. In such cases, we can classify and evaluate the model based on the outcomes by forming the confusion matrix.

But, in regression problems, the dependent variable will be continuous and confusion matrix can’t be formed in such cases. So we have to employ some other error metrics to evaluate the model performance. Some of the common error metrics we use in Regression Problems are **Mean absolute Error (MAE), Mean Absolute Percentage Error (MAPE), Mean Square Error (MSE), Root Mean Square Error (RMSE), R-Squared and Adjusted R\_Squared.** Here, we will use two of these Error Metrics, **MAPE** and **RMSE** for evaluating our models.

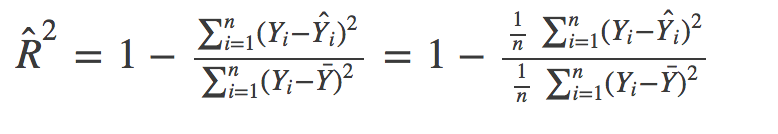
**Mean Absolute Percentage Error (MAPE)** is commonly used as a loss function for regression problems and in model evaluation, because of its very intuitive interpretation in terms of relative error. The MAPE measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error, as shown in the example below:



**Root Mean Square Error (RMSE)** is the standard deviation of the [residuals](https://www.statisticshowto.datasciencecentral.com/residual/) (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells us how concentrated the data is around the [line of best fit](https://www.statisticshowto.datasciencecentral.com/line-of-best-fit/).

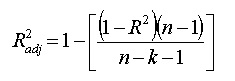


**R Squared & Adjusted R Squared** are often used for explanatory purposes and explains how well our selected independent variable(s) explain the variability in our dependent variable(s). Mathematically, R-Squared is given by:



The numerator is MSE (average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, **smaller the R\_squared and poorer is the model.**

Just like R², adjusted R² also shows how well terms fit a curve or line but adjusts for the number of terms in a model. It is given by below formula:



where n is the total number of observations and k is the number of predictors. Adjusted R² will always be less than or equal to R²

An adjusted R² will consider the marginal improvement added by an additional term in our model. So it will increase if we add the useful terms and it will decrease if we add less useful predictors. However, R² increases with increasing terms even though the model is not actually improving.

Basically the **MAE and MAPE won’t be affected by outliers**, but **MSE and RMSE will be affected by outliers**. Here in our dataset, we have some outliers, but they are **useful data** points. So, if we use RMSE, it will be better here when compared to MAPE or MAE as, RMSE will give more weightage to the outlier points as it is squared.

On the other hand, **R-Squared and Adjusted R-Squared** values shows us how a model is, by basics. Unlike other metrics, it describes how well the independent variables are capable of predicting the dependent variables. It alone gives the power of predictability of model. It ranges from **-infinity to 1**; where 1 is the best and close to 0 and negative values are worse. They can be used as **goodness of fit** metrics of models.

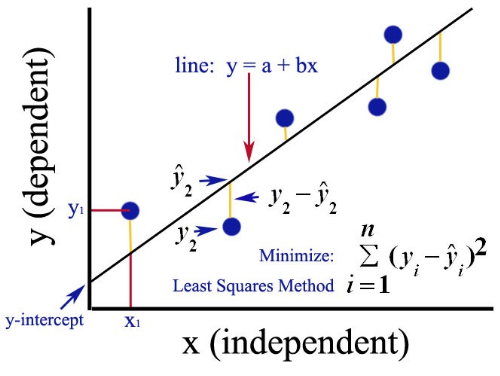
So, basically, in this project, we are going to use **RMSE, R-Squared and Adjusted R-Squared** metrics for evaluating and deciding the best model for this dataset based on **goodness of fit.**

* + 1. **Linear regression**

Linear regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression- Simple and Multiple. Simple Linear Regression means there will be one predictor and one response variable. Multiple Linear Regression is the current scenario which we face where there will be many predictors and a single target variable.

Linear Regression employs various methodologies to formulate the linear relationship and one among them is **the Least Squares Method**, where, there will be a linear line formed, which has the least squared error from the actual values, when compared to any other line formed for the data. A linear regression looks something similar to the below:

**Fig 2.6 Structure of Linear Regression based on Least Squares**



* + - 1. **Implementation**

Here we use **OLS(Optimum Least Square)** method in Python and R to develop the model. This method formulates the line with least squared error.

Using test\_train\_split method, we can sample the data into 80% for trining and 20% for testing in Python.There are various sampling methods such as Simple Random Sampling, Stratified Sampling which can also be used to generate sampled data. The test\_train\_split method will randomly shuffle the data and pick the data accordingly.

We sampled the data and built the model and the model made predictions. We can visualise the summary of the model using model method. For example, here we can see the model summary of reg\_model:

**Fig 2.7 (A) reg\_model Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | fare\_amount | **R-squared:** | 0.927 |
| **Model:** | OLS | **Adj. R-squared:** | 0.927 |
| **Method:** | Least Squares | **F-statistic:** | 5.421e+04 |
| **Date:** | Thu, 27 Jun 2019 | **Prob (F-statistic):** | 0.00 |
| **Time:** | 20:43:05 | **Log-Likelihood:** | -30463. |
| **No. Observations:** | 12852 | **AIC:** | 6.093e+04 |
| **Df Residuals:** | 12849 | **BIC:** | 6.095e+04 |
| **Df Model:** | 3 |  |  |
| **Covariance Type:** | nonrobust |  |  |

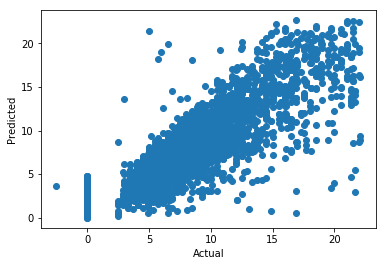
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **distance\_travelled** | 19.1451 | 0.090 | 212.015 | 0.000 | 18.968 | 19.322 |
| **Year** | 3.2769 | 0.060 | 55.031 | 0.000 | 3.160 | 3.394 |
| **Month** | 2.0063 | 0.058 | 34.708 | 0.000 | 1.893 | 2.120 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 2074.540 | **Durbin-Watson:** | 1.970 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 9156.902 |
| **Skew:** | 0.731 | **Prob(JB):** | 0.00 |
| **Kurtosis:** | 6.868 | **Cond. No.** | 3.30 |

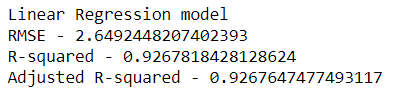
From the model summary, we can see that the **R-squared and Adjusted R-squared** values are close to 1. This shows that the model developed can determine 92-93% of the predicted values. So this model has a high **goodness of fit** index.

Similarly the column **‘coef’**explains 1 unit of a variable brings about how many units of change in the predicted variable. Based on this, the highest change (i.e., more correlation) is brought about by the **‘distance\_travelled’** column. The **p-value** column is also negligible for all variables and so none of the variables are unnecessary in predicting the target variable. The scatter plot between ‘actual’ and ‘predicted’ values of cnt\_model is shown below:

**Fig 2.7 (B) Actual vs Predicted (reg\_model)**



The linear relationship in the scatter plot shows that the predicted values are close to the actual values. If more data is fed in, the model performance can be improved. We evaluated the model based on the predictions made. The results are as follows:



Now let us look upon the performance of other models and decide on the best model for this dataset. The next complex dataset is the decision tree.

* + 1. **Decision Tree**

The first ML algorithm, that we are going to use here is the **Decision Tree Algorithm.** Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A **decision node** has two or more branches. **Leaf node** represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node.** Decision trees can handle both categorical and numerical data.

**Fig 2.8 Structure of a Decision Tree**

 **Root Node**

So here we can use the Decision Tree for Regression and predict the cab fare based on the other data.

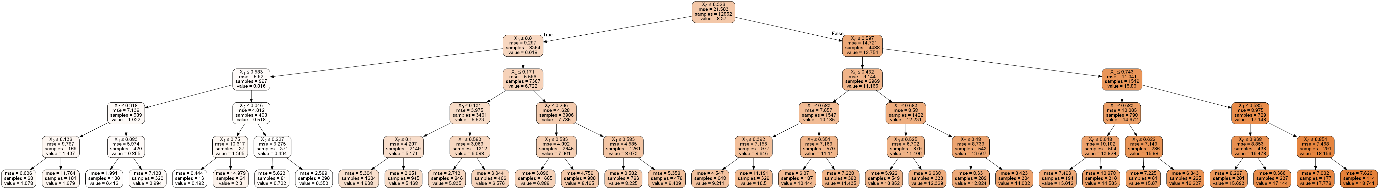
* + - 1. **Implementation**

Here we are going to use the **DecisionTreeRegressor** for building the model in Python and **RPart** for building model in R. They use **CART Algorithm for formation of trees** and using test\_train\_split method, we can sample the data into 80% for trining and 20% for testing in Python

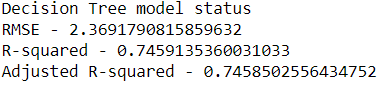
After the model is built, using the independent variables of test data, we will predict the dependent variables and then evaluate the error metrics for this model.

Here we can limit the depth of the tree to keep constraints in developing the model, otherwise the tree will develop until it reaches exact leaf node for all branches. This is called ‘**Pruning**’. When we tried with different depths for the decision tree, a **tree depth of 5** was optimum for this dataset as after that if the tree depth is increased, the output change was not phenominal. Below, we can see the **sample decision tree generated for the tree model.**

**Fig 2.9 Decision Tree For Tree model** (Python Code in Appendix B)

****

We sampled the data and built the model and the model made predictions and we evaluated the model based on the predictions made. The results are as follows:



Here, we can see, the RMSE is slightly lesser when compared to regression model, but the goodness of fit metrics of R-squared and Adjusted R-squared values have been considerably less when compared to Linear Regression Model.

This shows that the **decision tree model is not as effective as the Linear regression model** for this dataset.

Now let us look upon the next regression model – Random Forest.

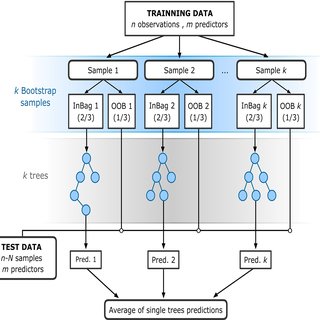
* + 1. **Random Forest**

The next regression model that we are going to use to predict the target variables is the Random Forest model. Random Forest is a supervised learning algorithm. From it’s name, we can sense that it creates a forest and makes it somehow random. The forest it builds, is an **ensemble** **of Decision Trees**, most of the time trained with the “**bagging**” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Random Forest basically is a collection of many decision trees and it combines the result of all the decision trees to get more accurate prediction of dependent variable.

Random Forest algorithm can be used for both Classification and Regression problems and it is a great advantage for this algorithm. Here, we can use this algorithm to predict the dependent variable by regression.

**Fig 2.10 Structure of a Random Forest**



Here we can use the Random Forest for Regression and predict the cab fare.

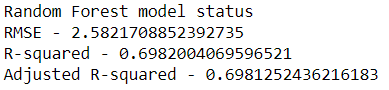
* + - 1. **Implementation**

Here we are going to use the **RandomForestRegressor** for building the model in Python and **RPart** for building model in R. We can use the same sampled data we used for decision trees, here too for making predictions.

As said previously, Random Forest is an ensemble of Decision Trees. We can specify and decide on the number of trees to be used in the algorithm. On trying out different number of trees for this dataset, **10 trees** is decided for building this model as above 10, the result of the predictions didn’t change much.

Usually, ensemble models perform better when compared to the normal ML algorithms. But it doesn’t guarantee that ensemble models are always the better ones.

We sampled the data and built the model and the model made predictions and we evaluated the model based on the predictions made. The results are as follows:



Here, we can see, the RMSE has increased and goodness of fit metrics; R-squared and Adjusted R-squared have even more reduced when compared to Decision Tree model.

This shows that the **random forest model is not as effective as the Linear regression model and the decision tree model** for this dataset.

**Chapter 3**

**Conclusion**

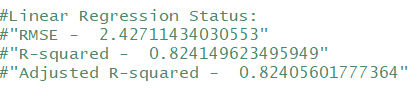
**3.1 Model Selection**

Here initially we performed various pre-processing techniques on the data as part of data cleaning and we made the data ready for Modeling phase. Then we build models using three different algorithms to decide which model is better in predicting for this data. So finally we have developed the models and we have evaluated all the three models.

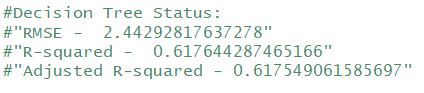
Based on the error metrics, though Decision Tree and Random Forest algorithms are able to predict the target variable, they are not better when compared to **Linear Regression model** in predicting the target variable for this dataset. This is because, though RMSE hasn’t considerably varied between the models, the **goodness of fit metrics R-squared and Adjusted R-squared are high for Linear Regression model** when compared to the other two models.

Even for the R-code, the error metrics are as follows when the models are run:

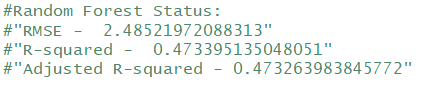
**Linear Regression:**



**Decision Tree:**



**Random Forest:**



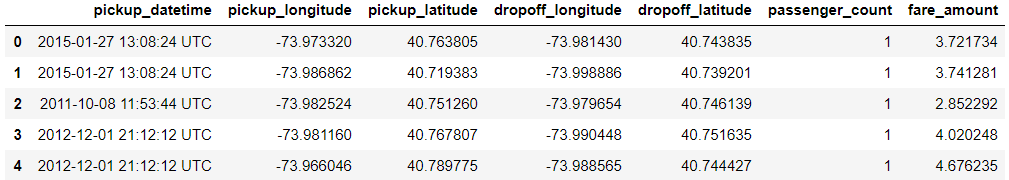
So, for this dataset, we use **Linear Regression model** to predict the cab fare for our launch across the country

**3.2 Output**

Using Linear Regression Model in R and Python, let us predict the cab fare based on the test data given in **test.csv.**

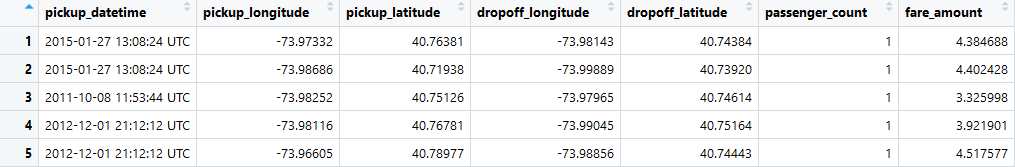
The output in Python is stored in **Python Test Output.csv** file and it is as follows:

**Fig 3.1 (A) Python Output** (Python Code in Appendix B)



The output in R is stored in **R Test Output.csv** file and it is as follows:

**Fig 3.1 (B) R Output** (R Code in Appendix B)

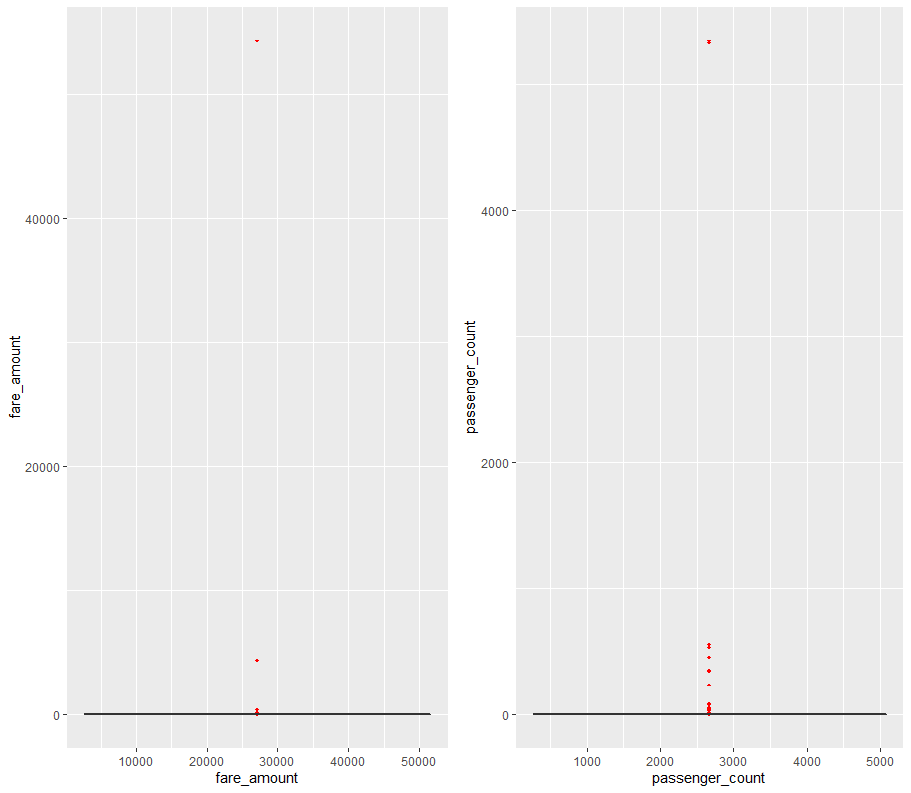


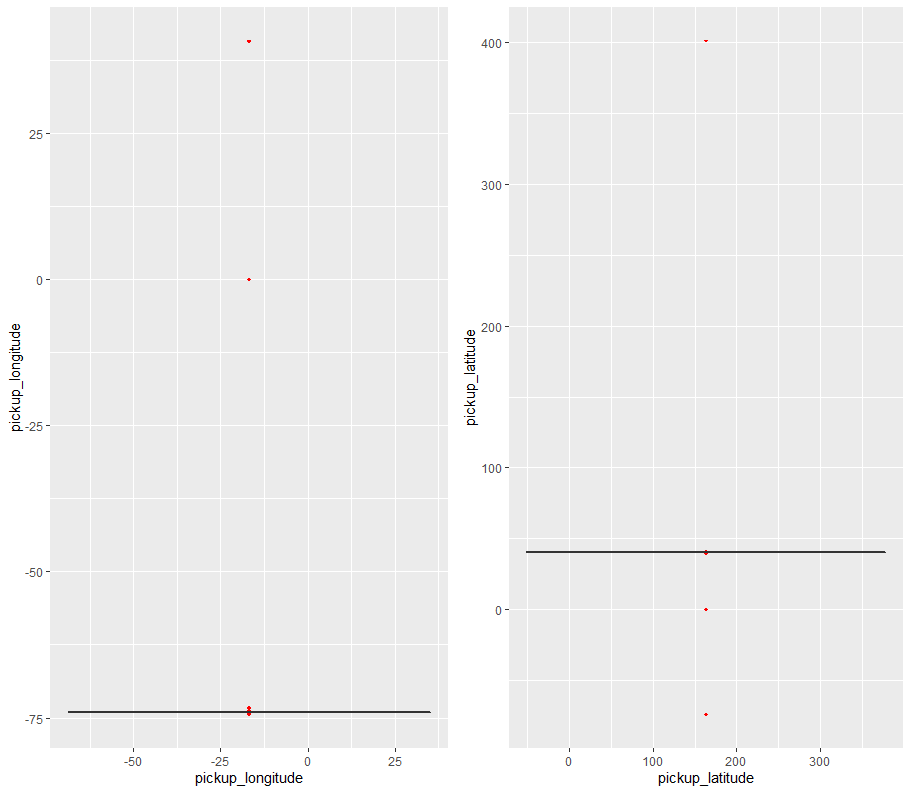
Thus the model is generated based on the data and the output is predicted by the model and stored in the csv files.

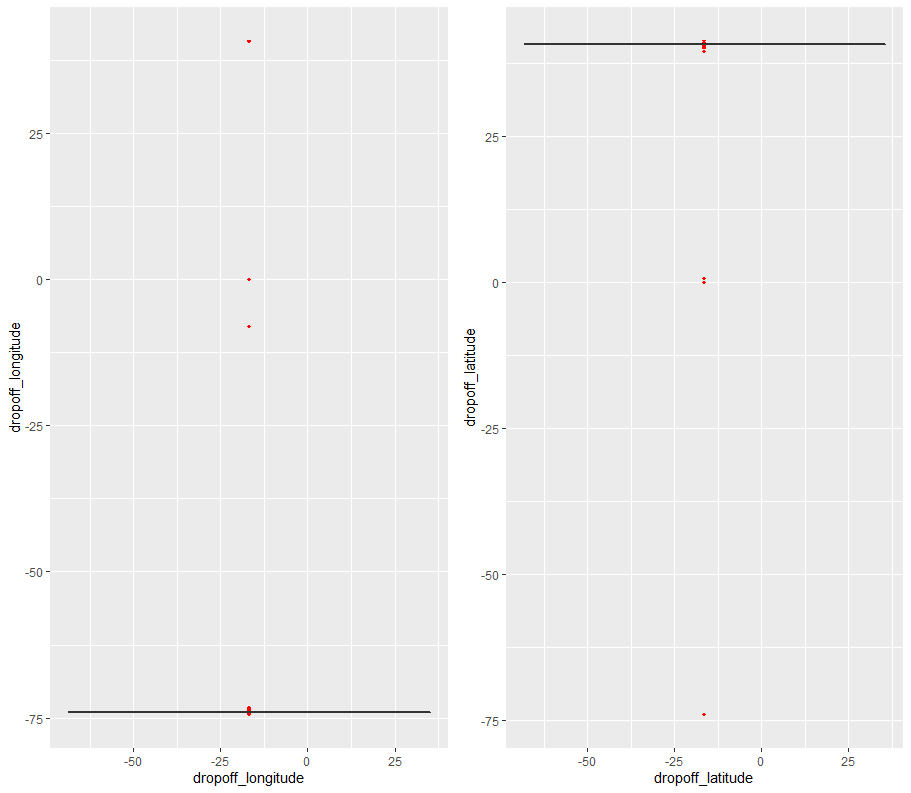
**Thus we have trained a model which can predict the cab fare based on latitude, longitude and timestamp data.**

**Appendix A - (Extra figures)**

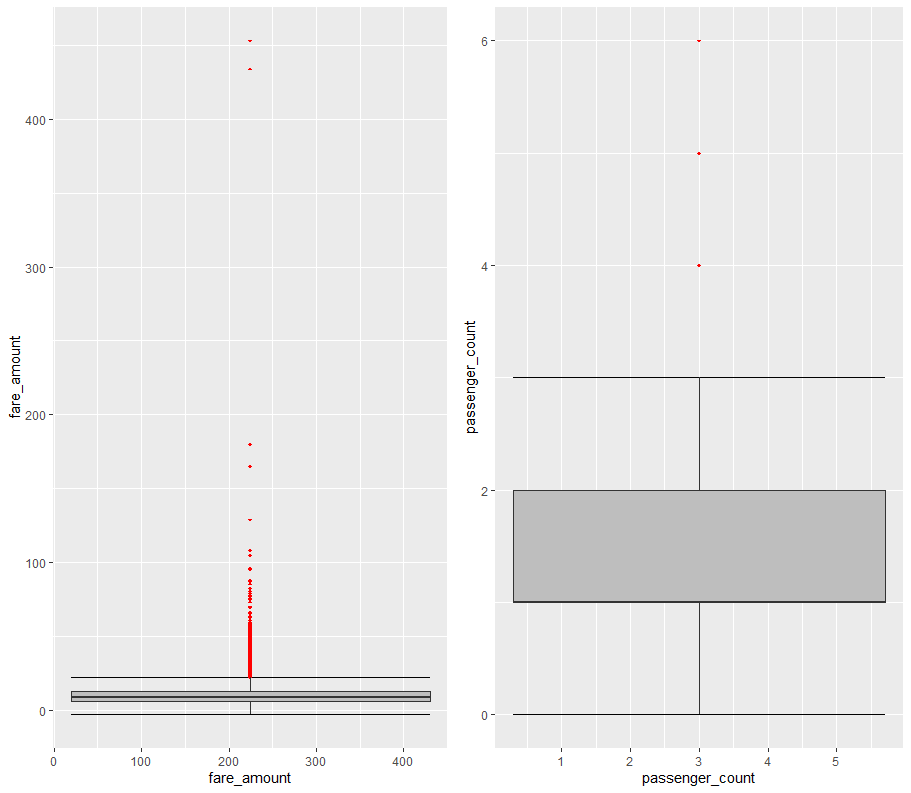
**Fig A.1 - Outlier Analysis - Box and Whisker Plots (R Plot)**

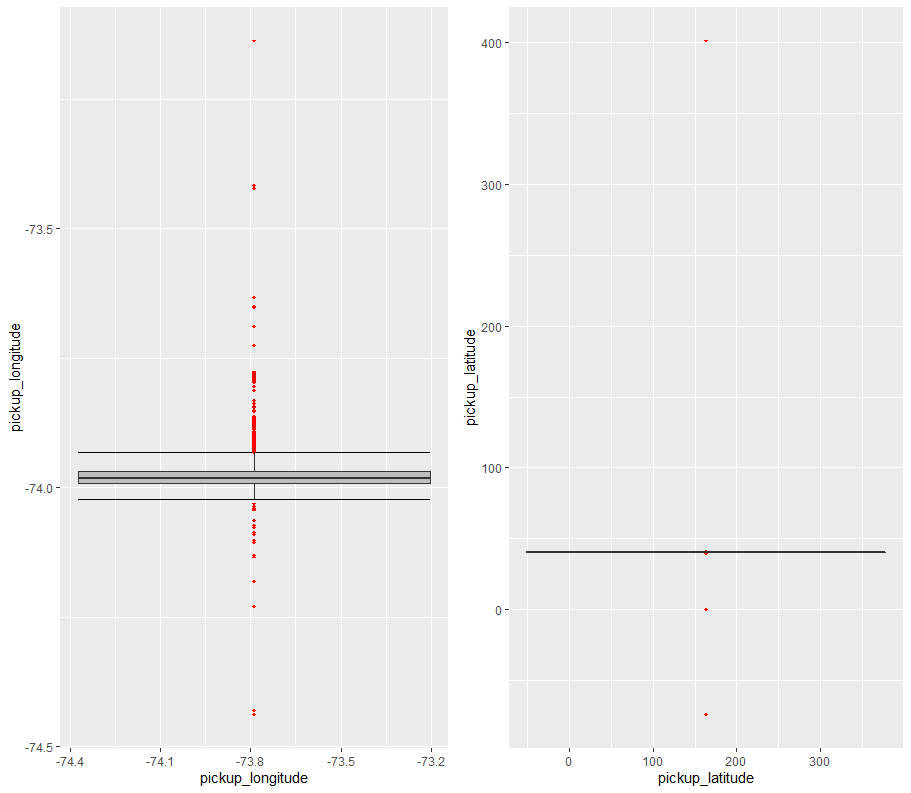


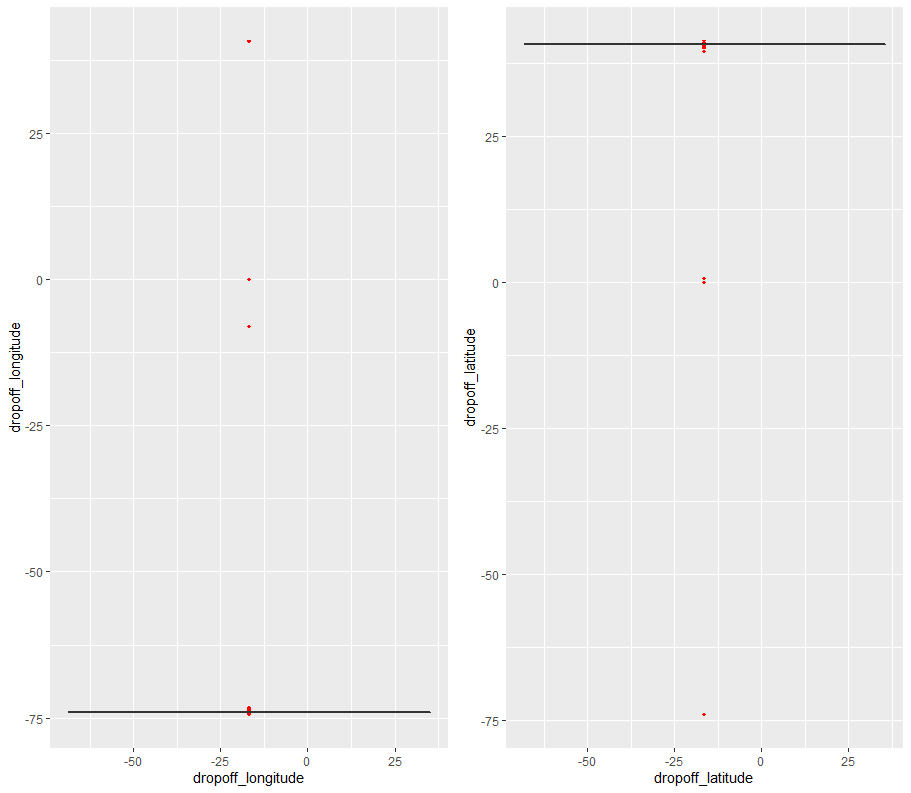




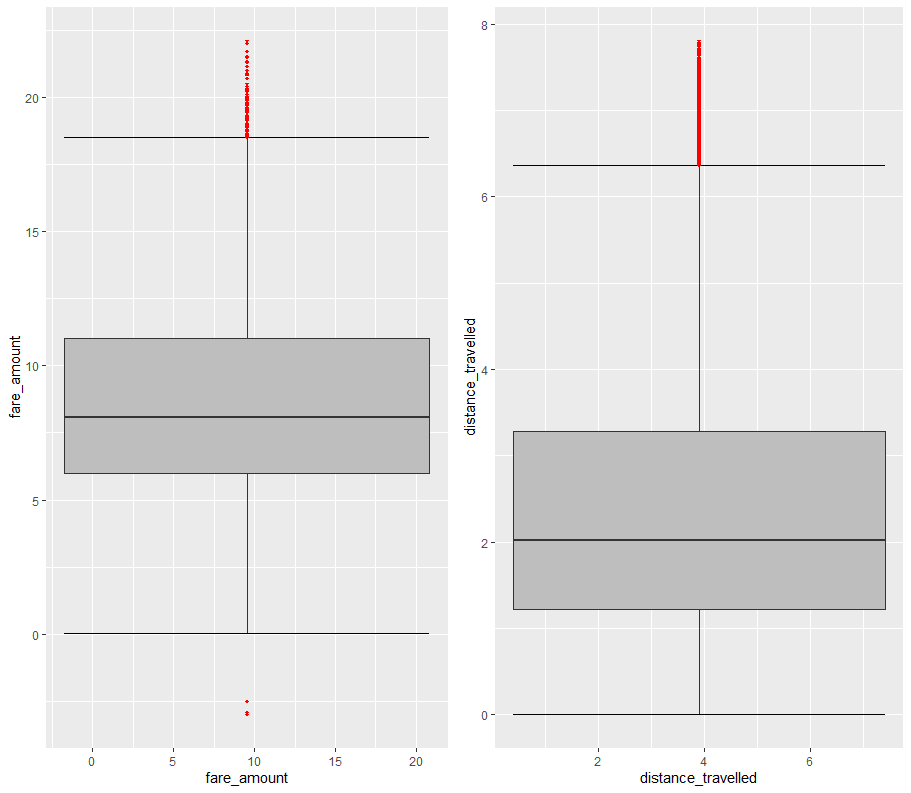
**Fig A.2 – Box and Whisker Plot(After Outlier analysis) (R Plot)**



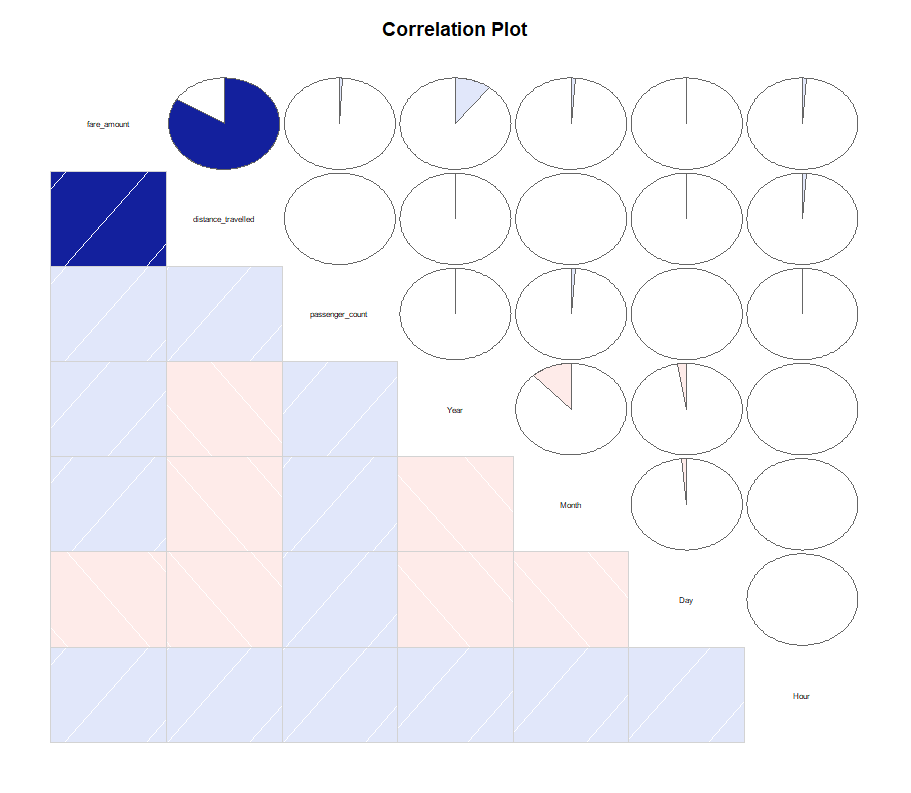




**Fig A.3 –Final Box and Whisker Plot(After Outlier analysis and data format) (R Plot)**

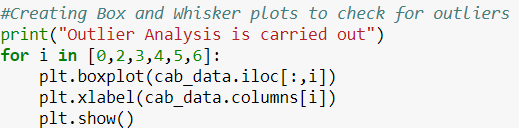


**Fig A.4 HeatMap (R Plot)**

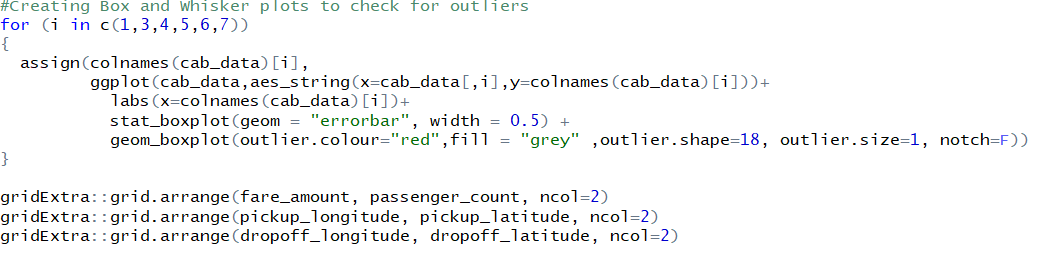


**Appendix B – (Code)**

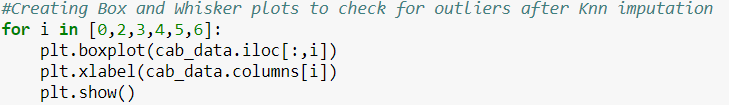
**Fig 2.1 Outlier Analysis – Box and Whisker Plots (Python Code)**



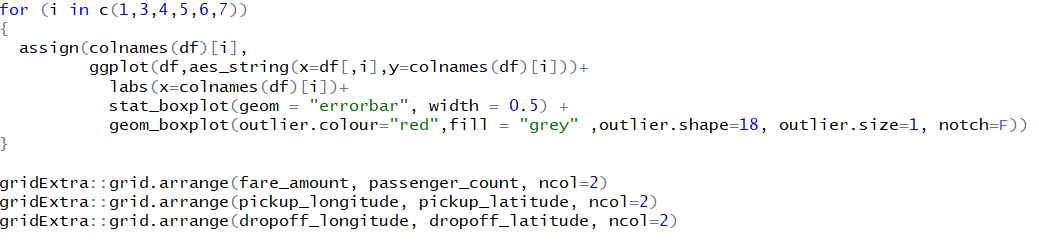
**Fig A.1 Outlier Analysis – Box and Whisker Plots (R Code)**



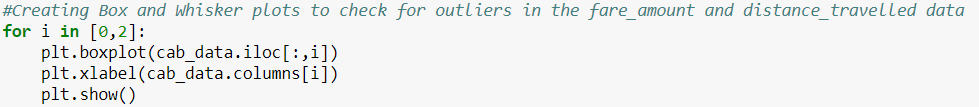
**Fig 2.2 Box and Whisker Plot(After Outlier analysis) (Python Code)**



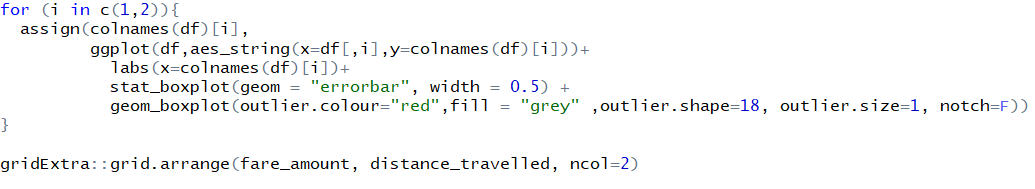
**Fig A.2 Box and Whisker Plot(After Outlier analysis) (R Code)**



**Fig 2.3 –Box and Whisker Plot(After Outlier analysis and data format) (Python Code)**



**Fig A.3 –Final Box and Whisker Plot(After Outlier analysis and data format) (R Code)**



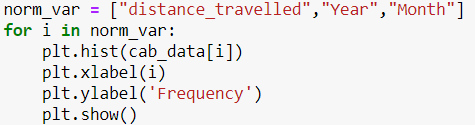
**Fig 2.4 Heat Map (Python Code)**



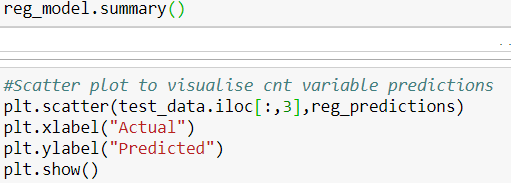
**Fig A.4 Heat Map (R Code)**



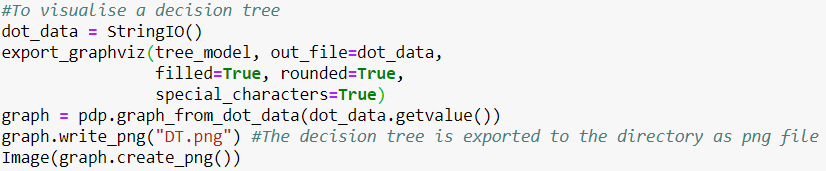
**Fig 2.5(A) Feature Scaling**



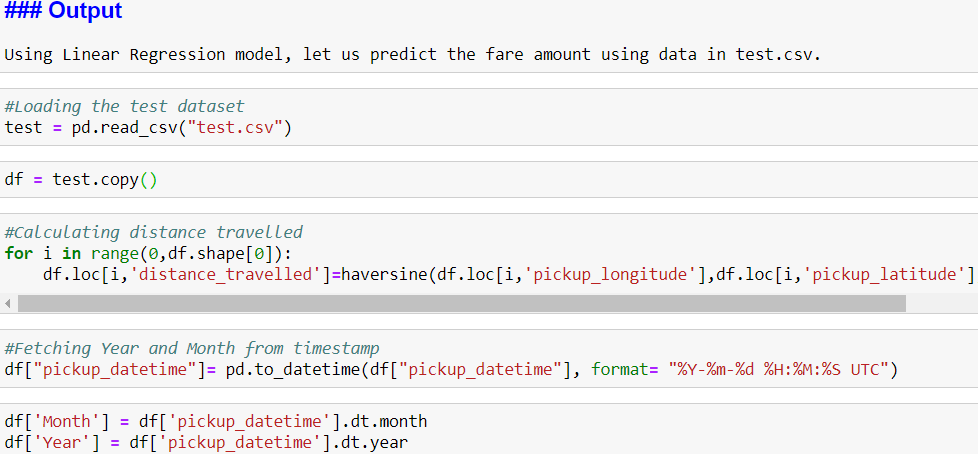
**Fig 2.7 (A) & (B) reg\_model Characteristics and Scatter Plot of Predicted Vs Actual**

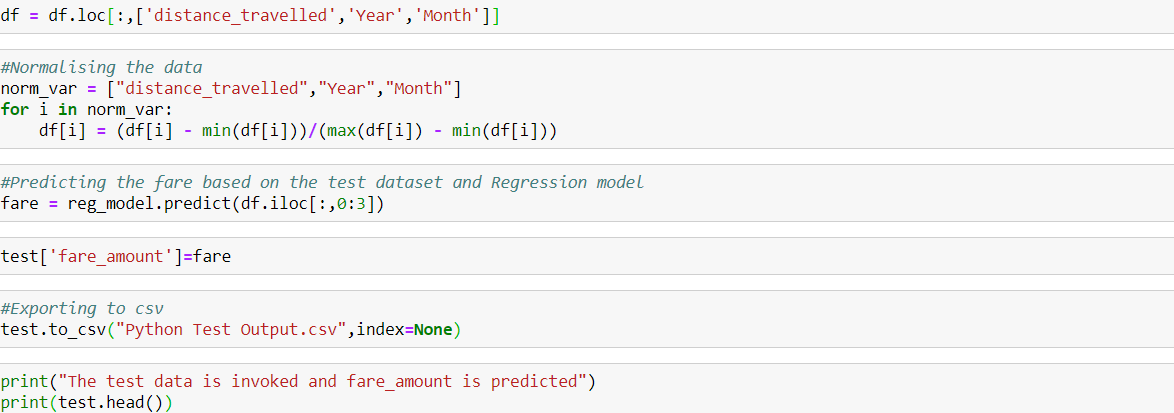


**Fig 2.9 Decision Tree structure (Python Code)**

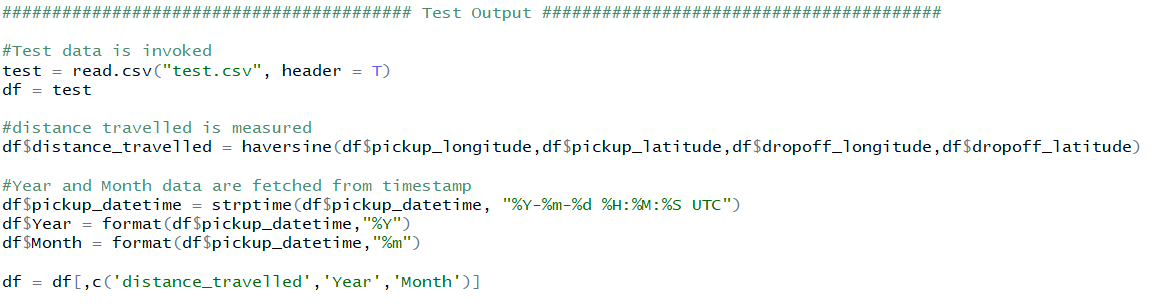


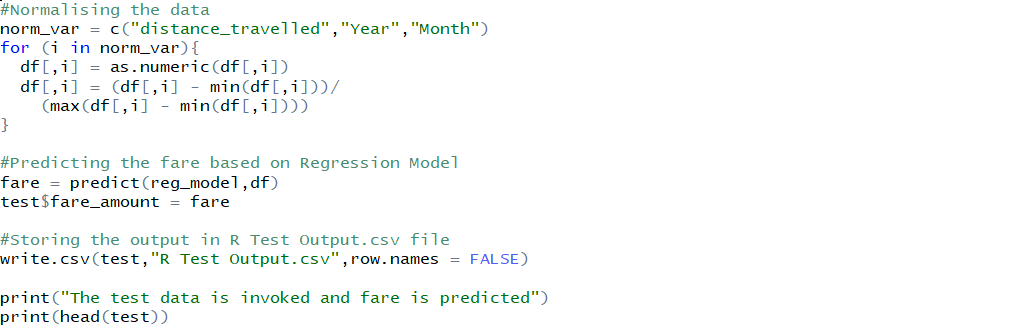
**Fig 3.1 (A) Python Output**



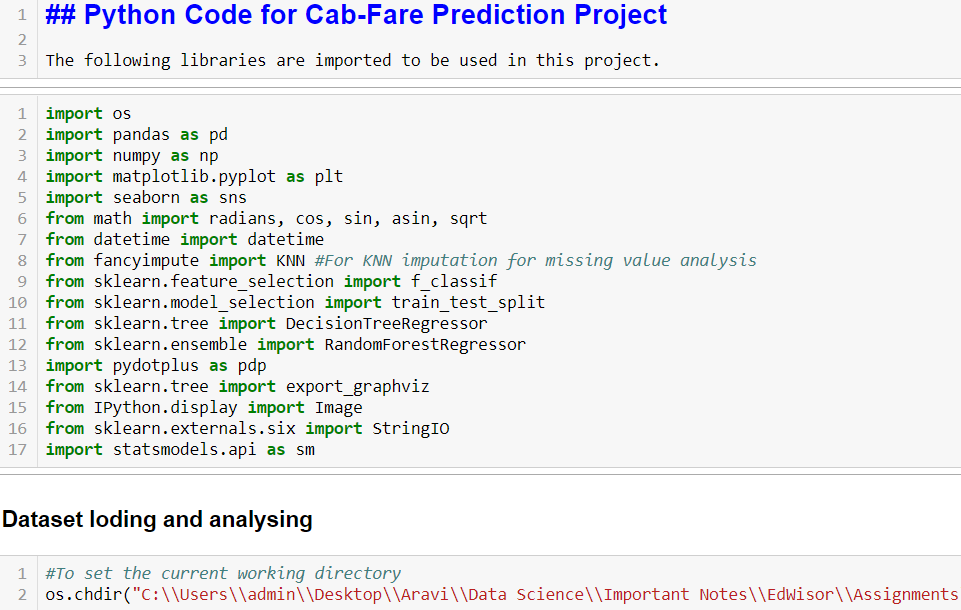


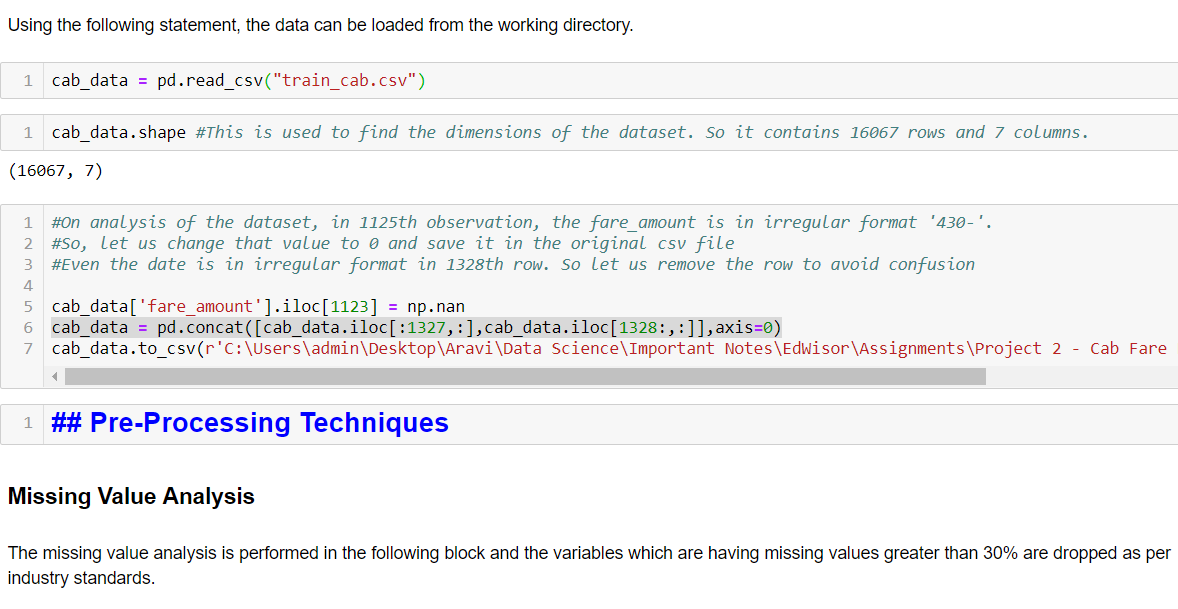
**Fig 3.1 (B) R Output**

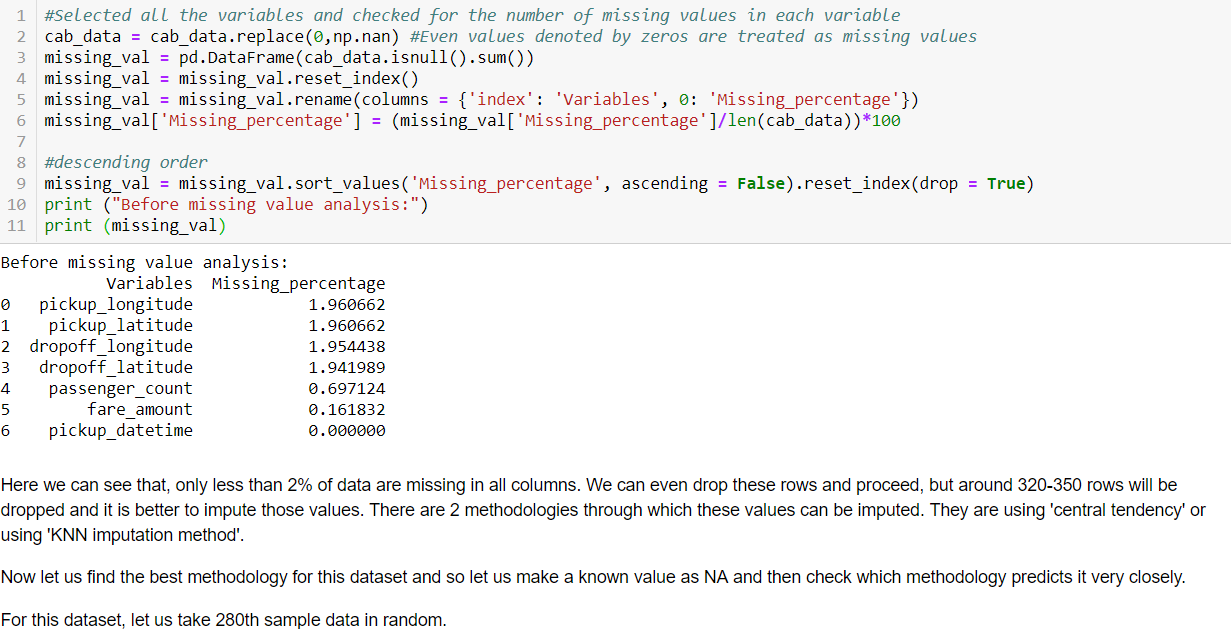


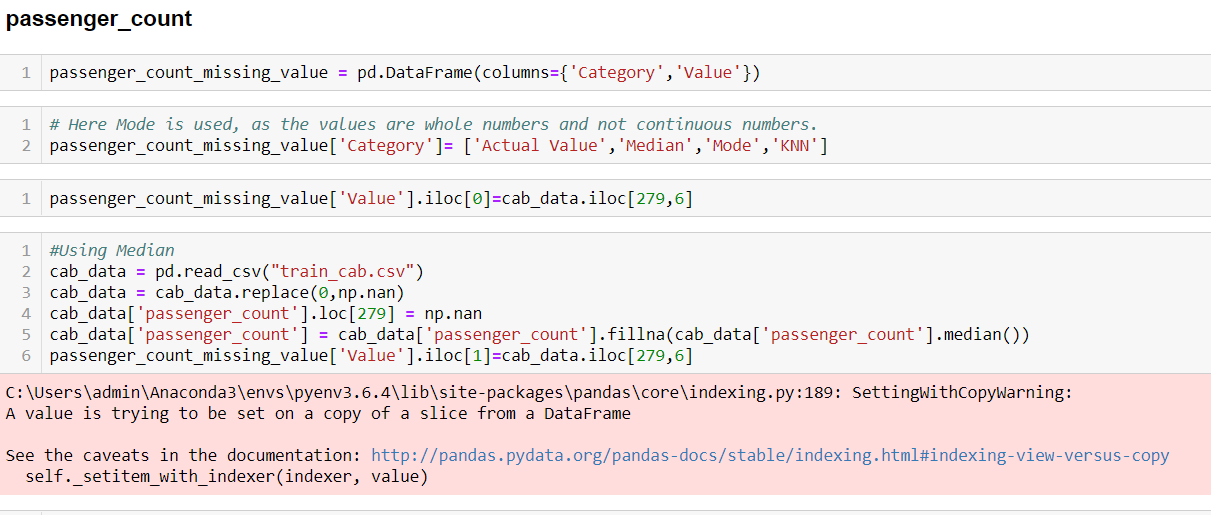


**Complete Python Code**

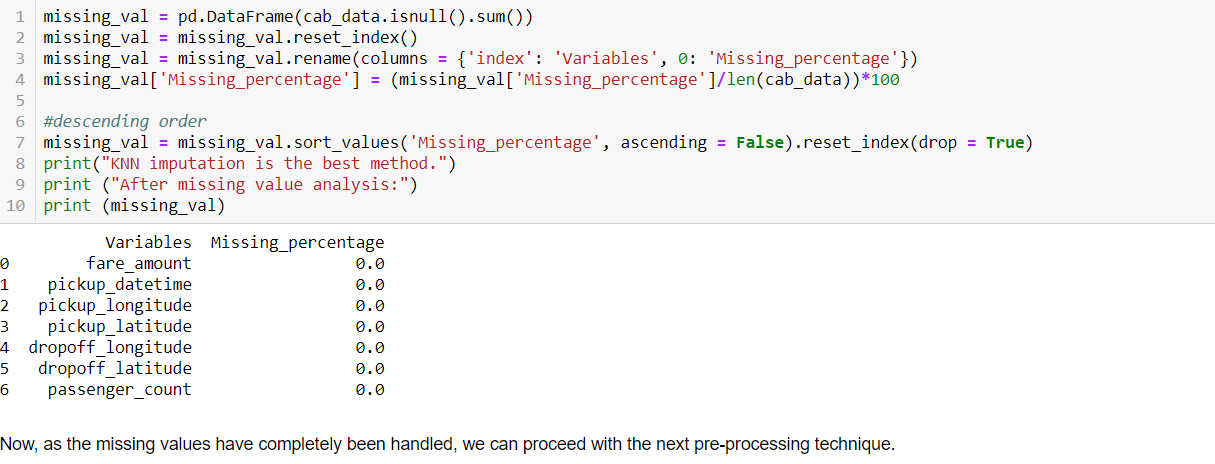


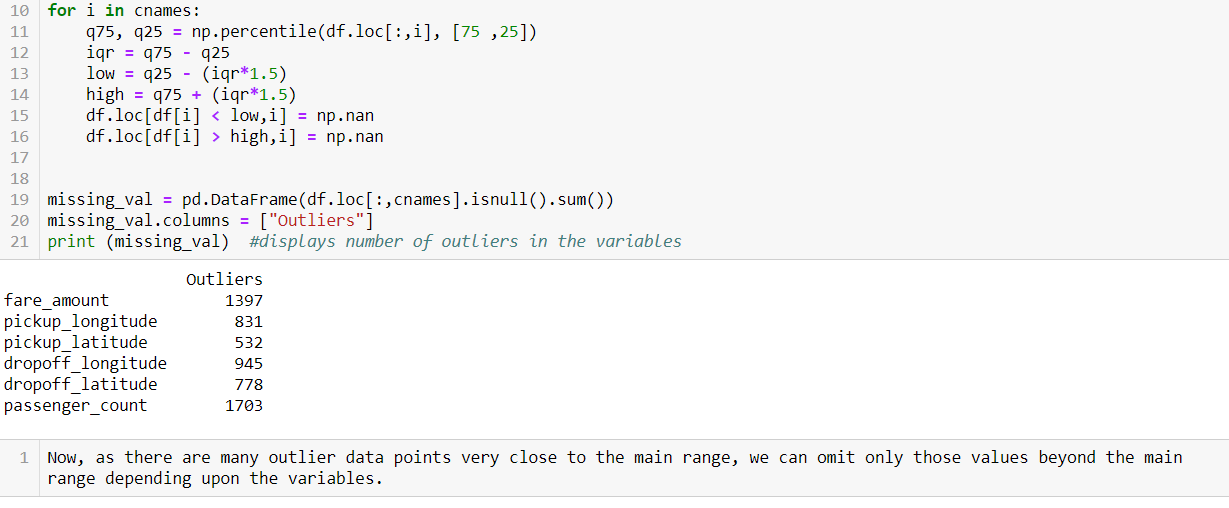
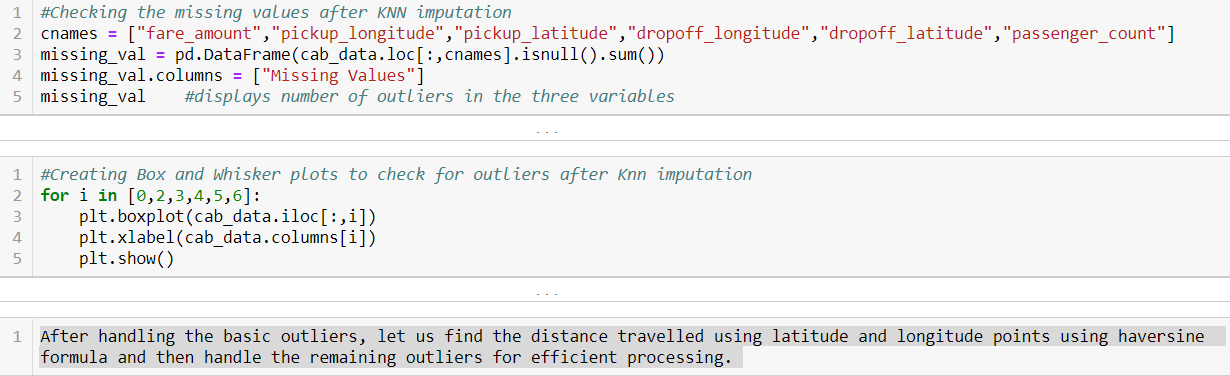
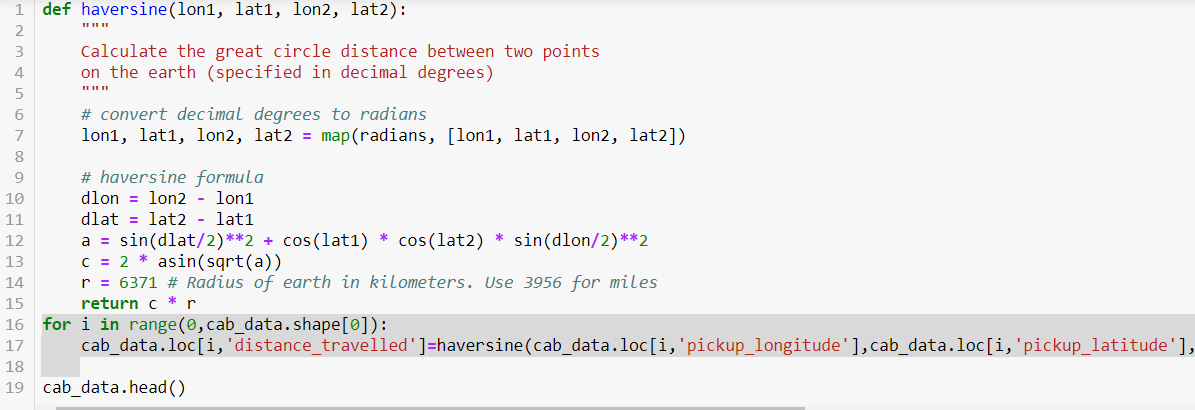


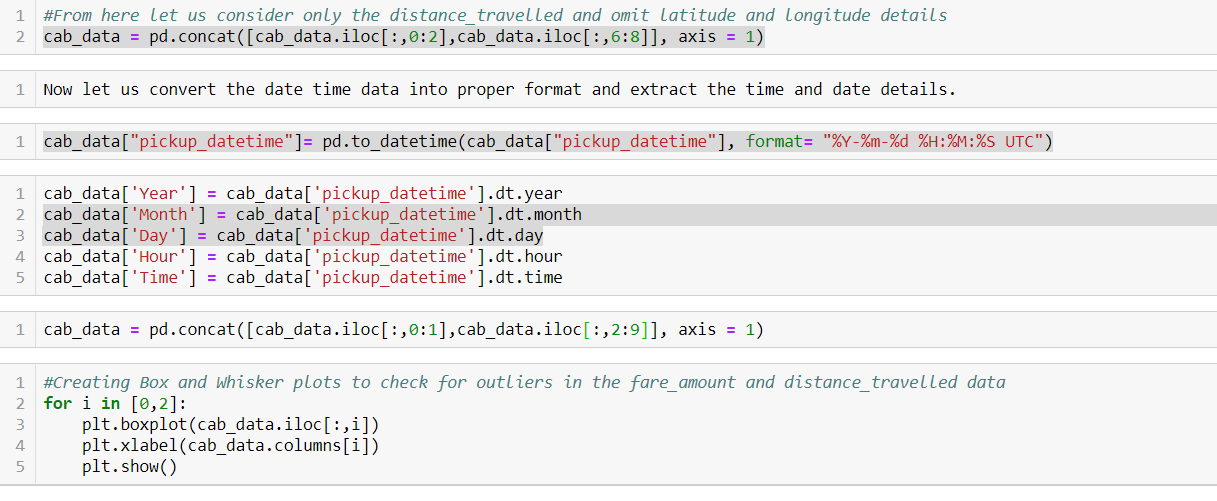
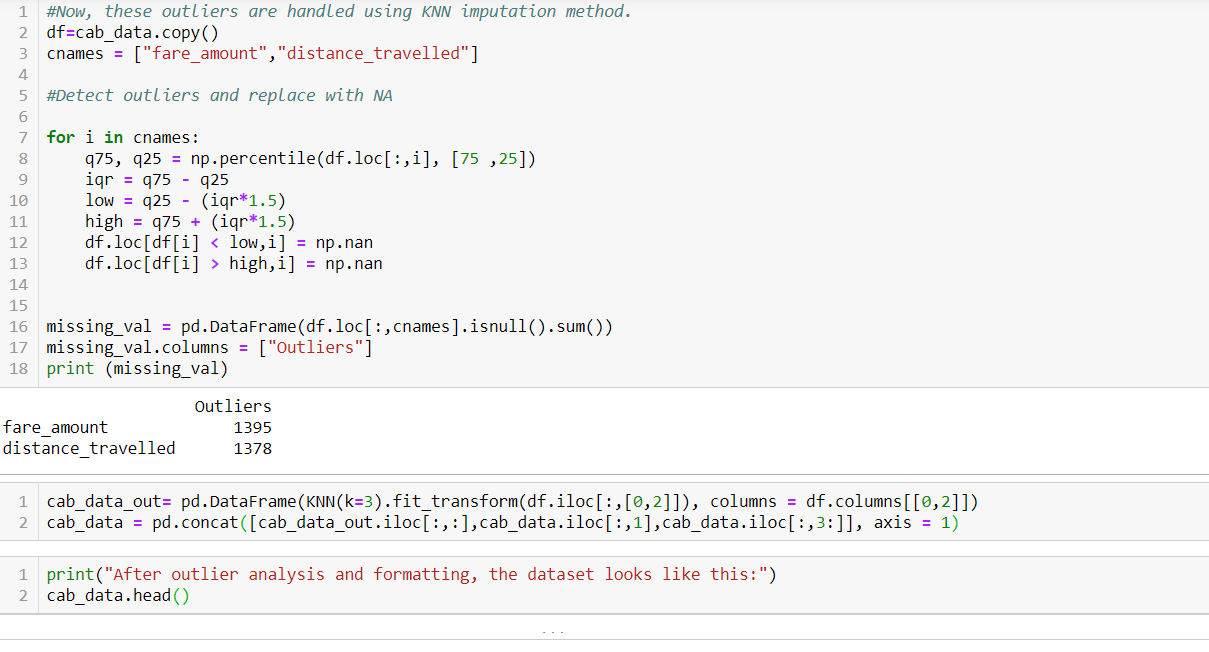
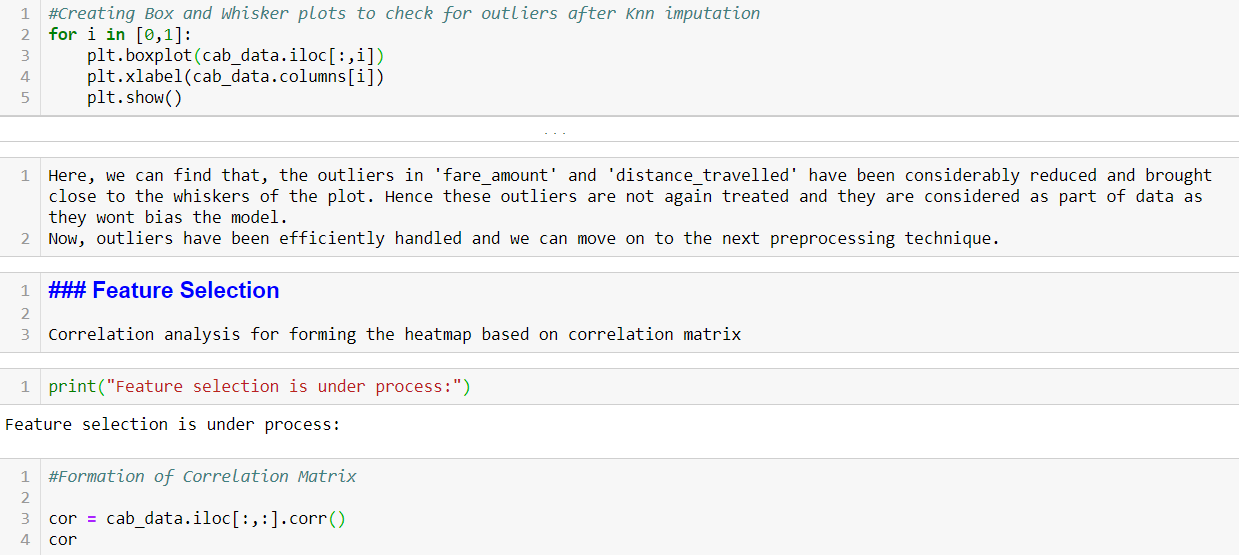


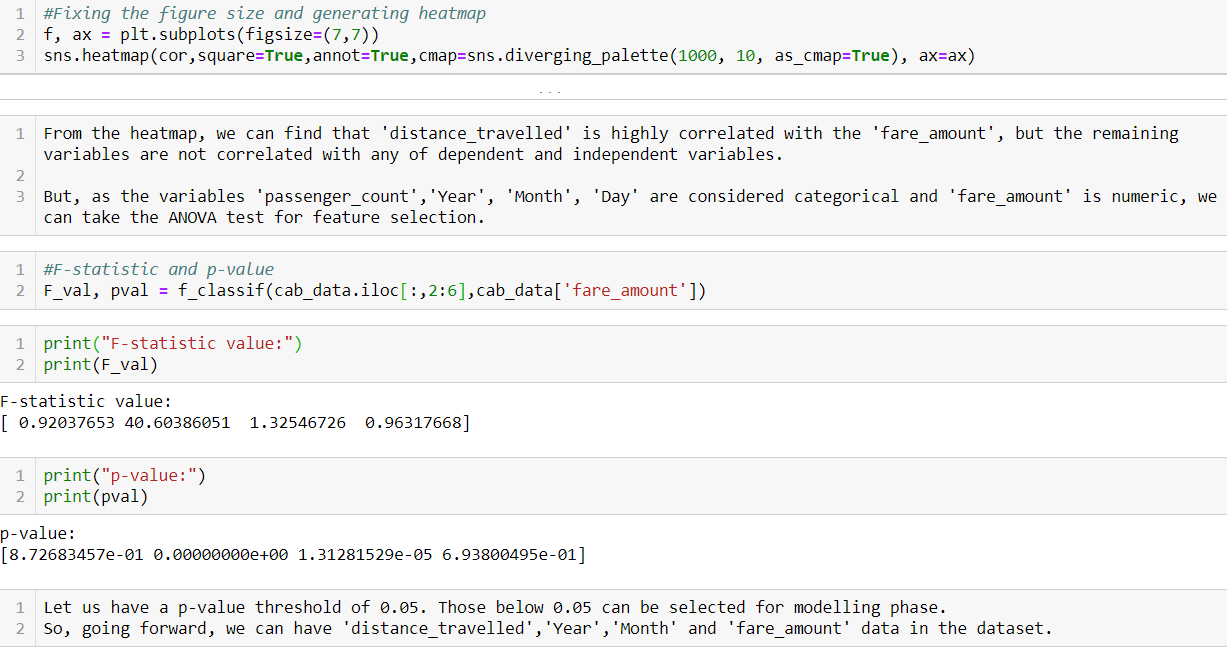
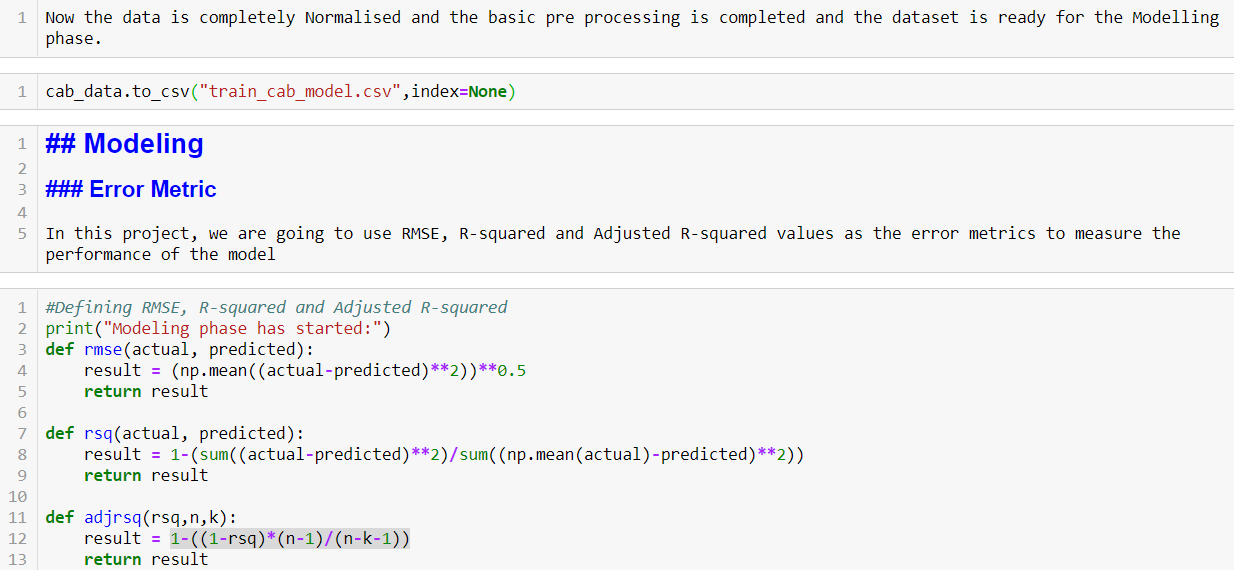


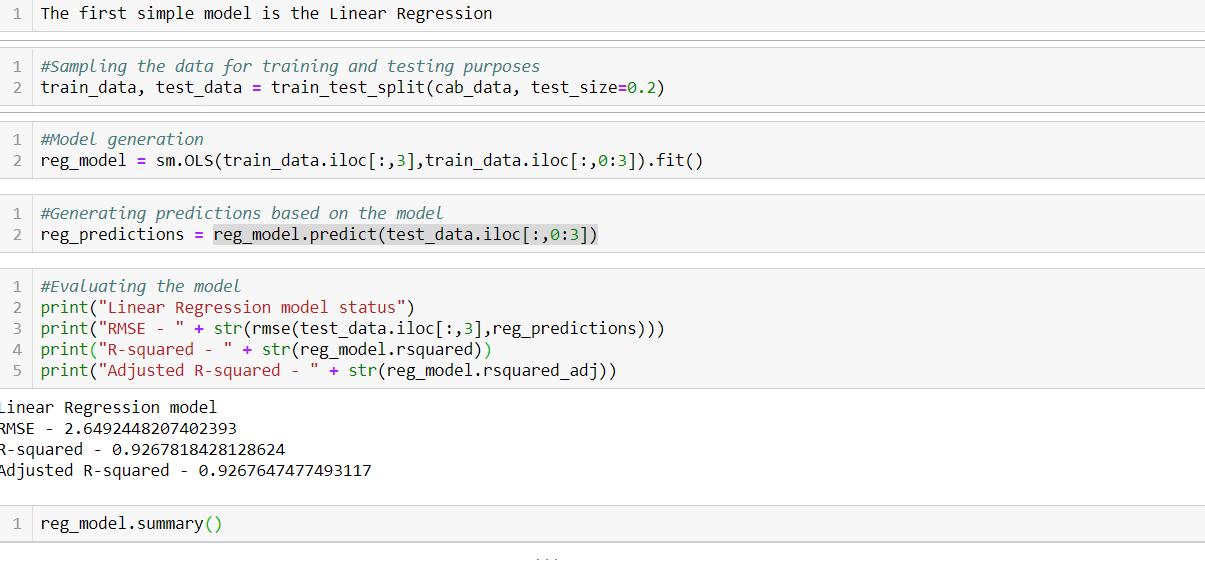
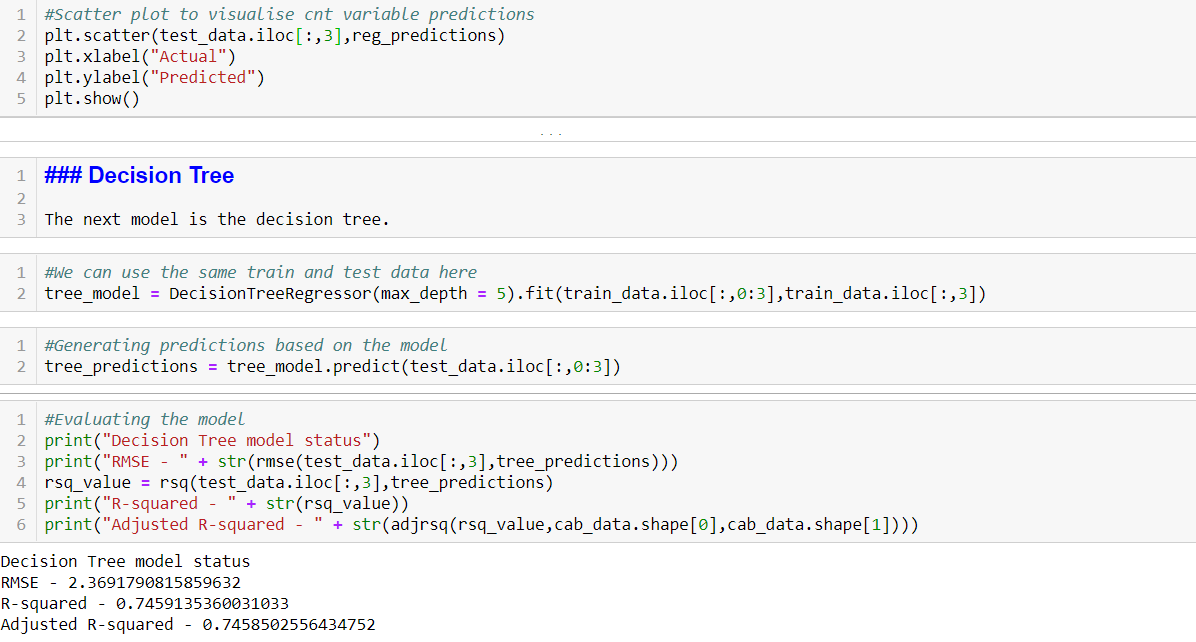
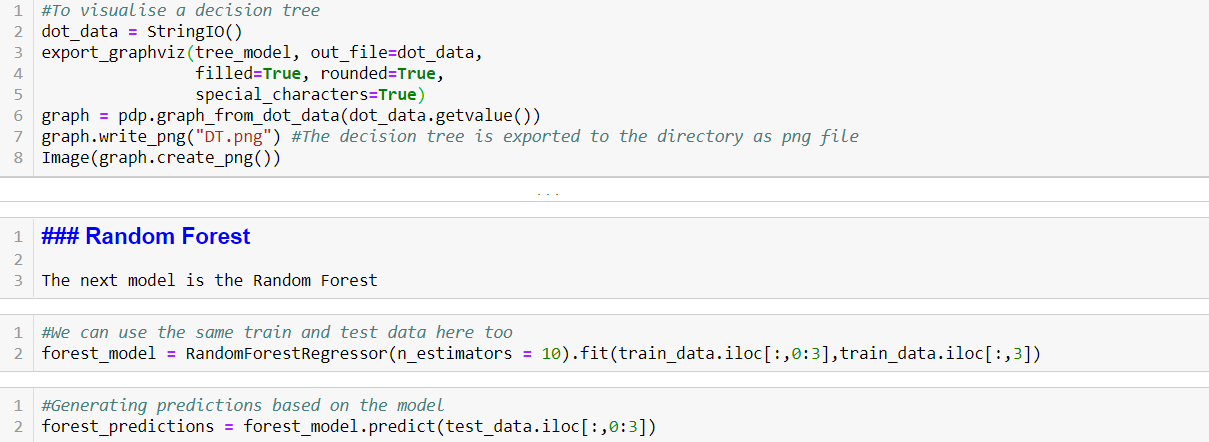


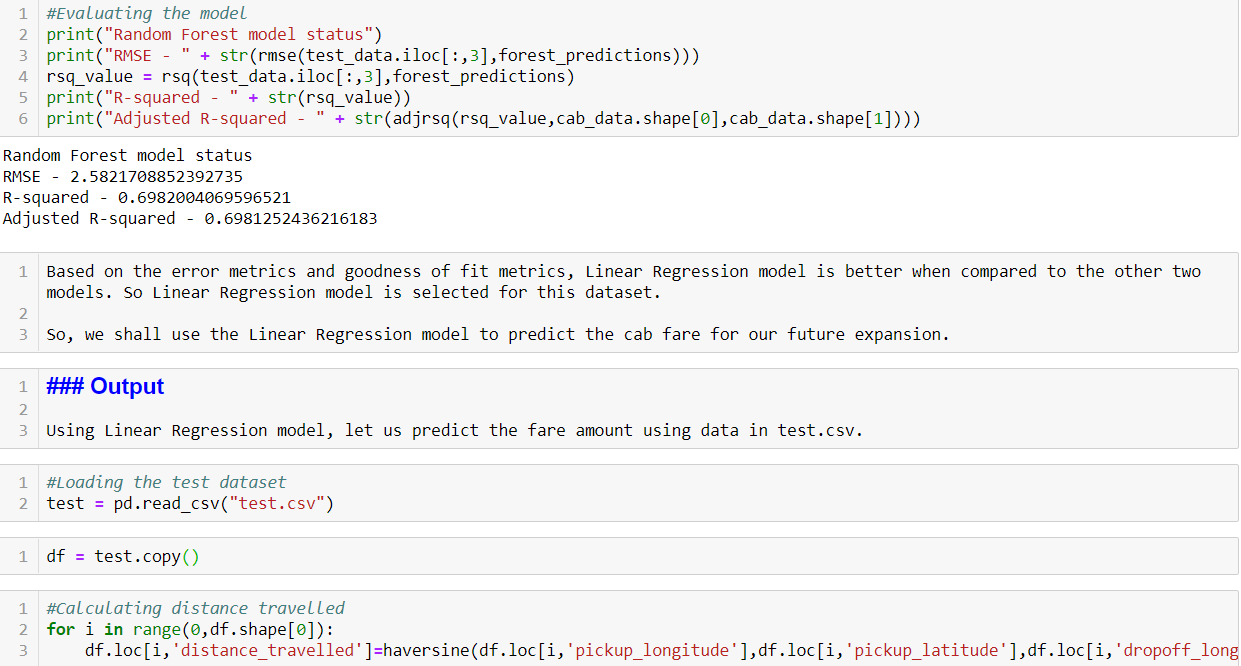
 

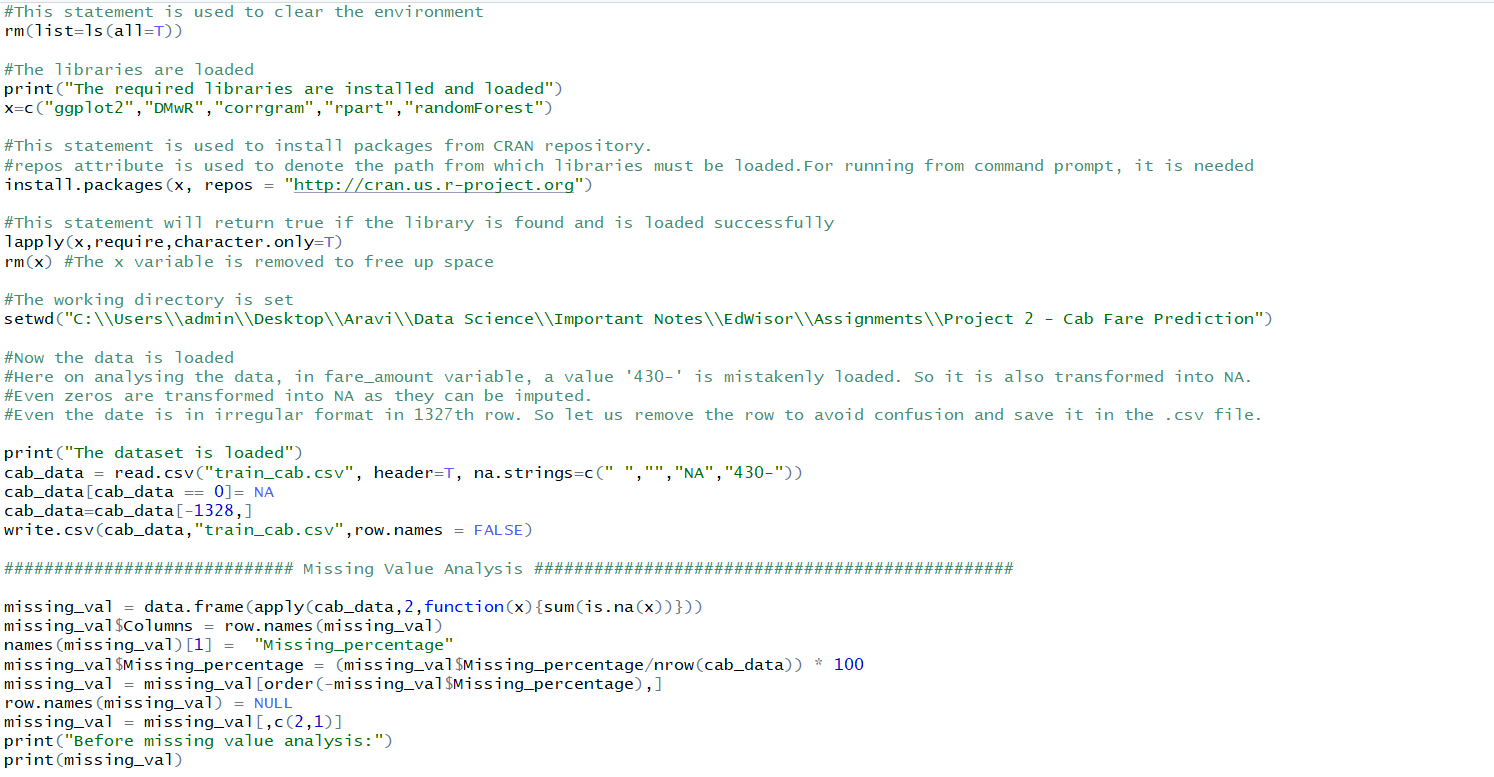
  

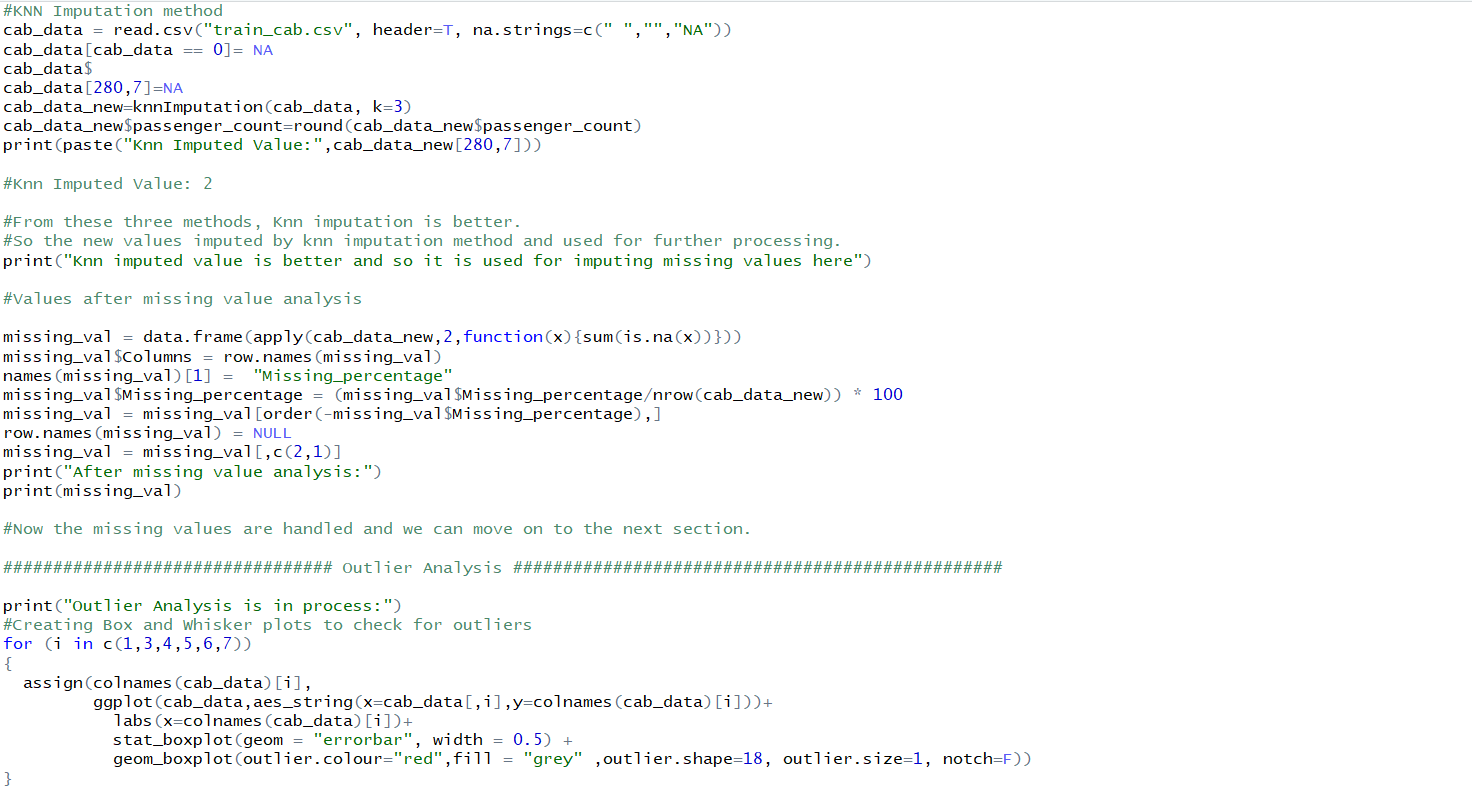
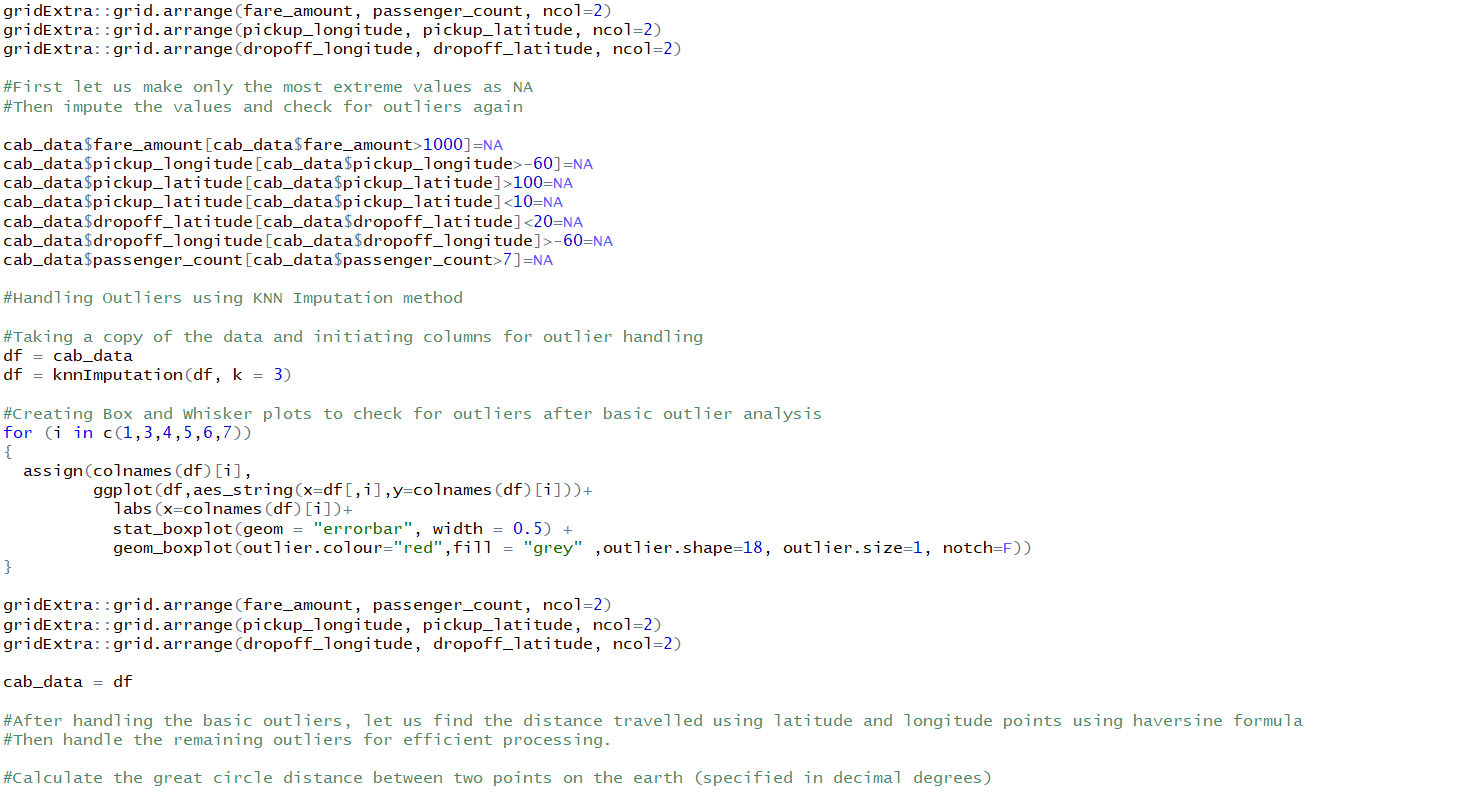
  

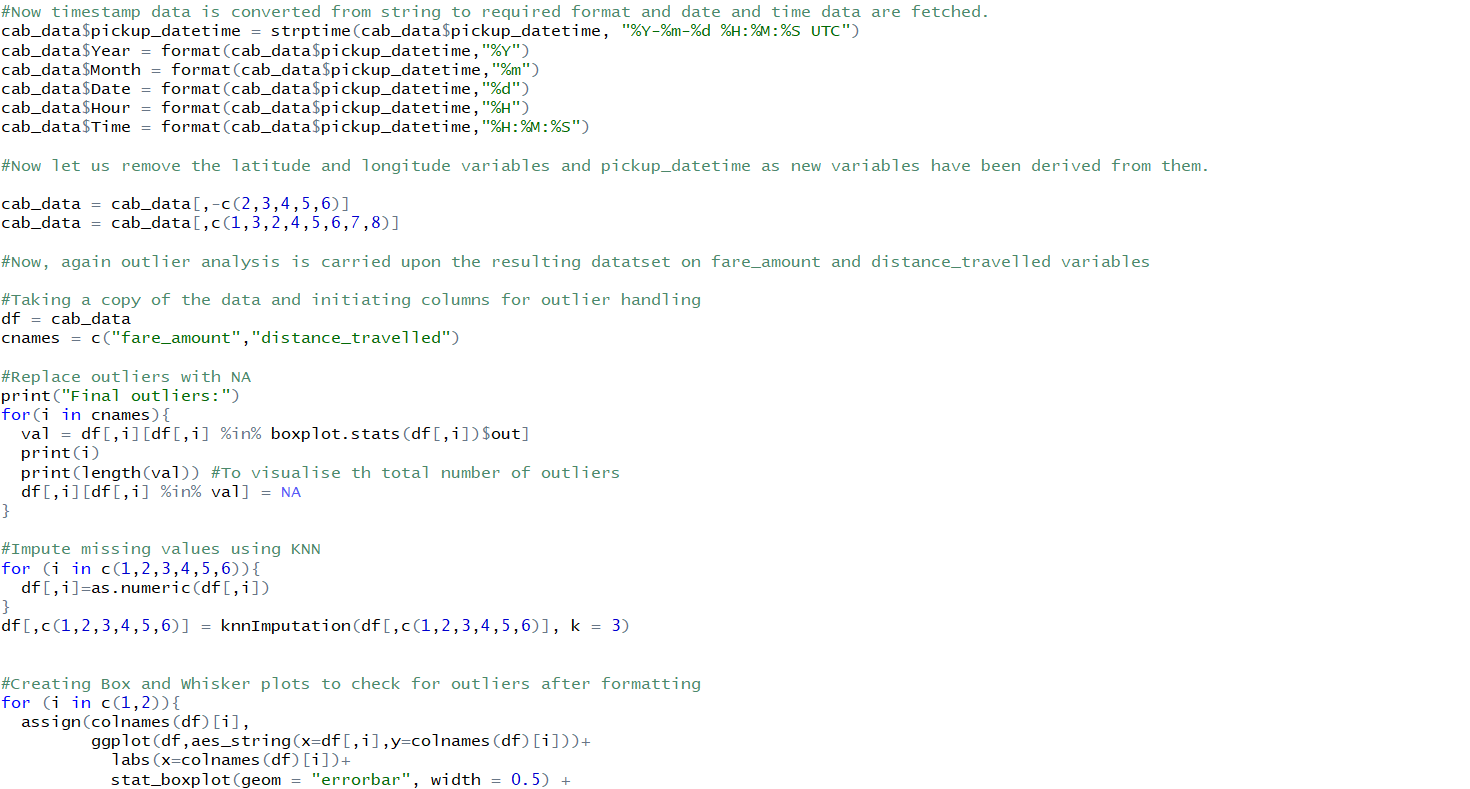
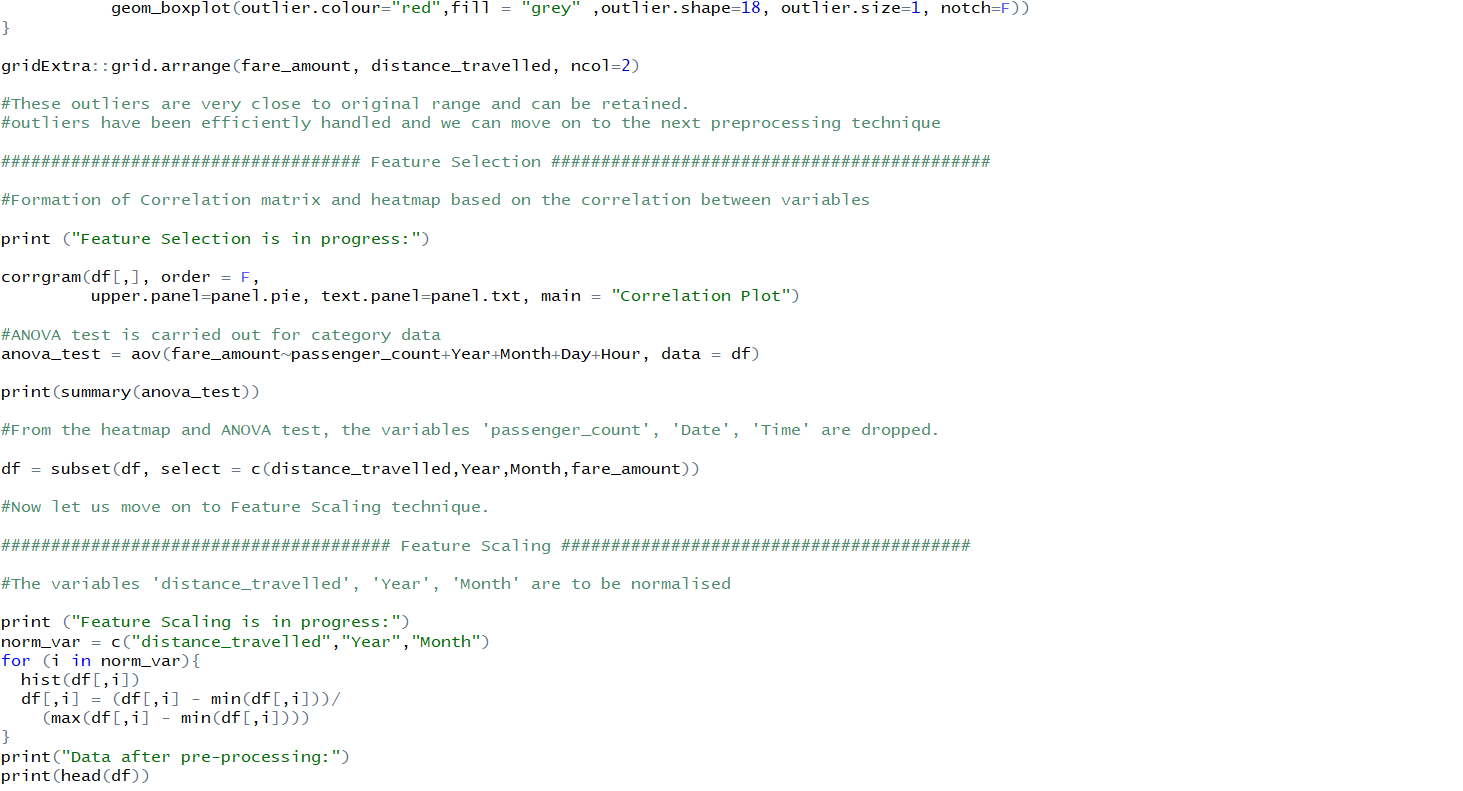
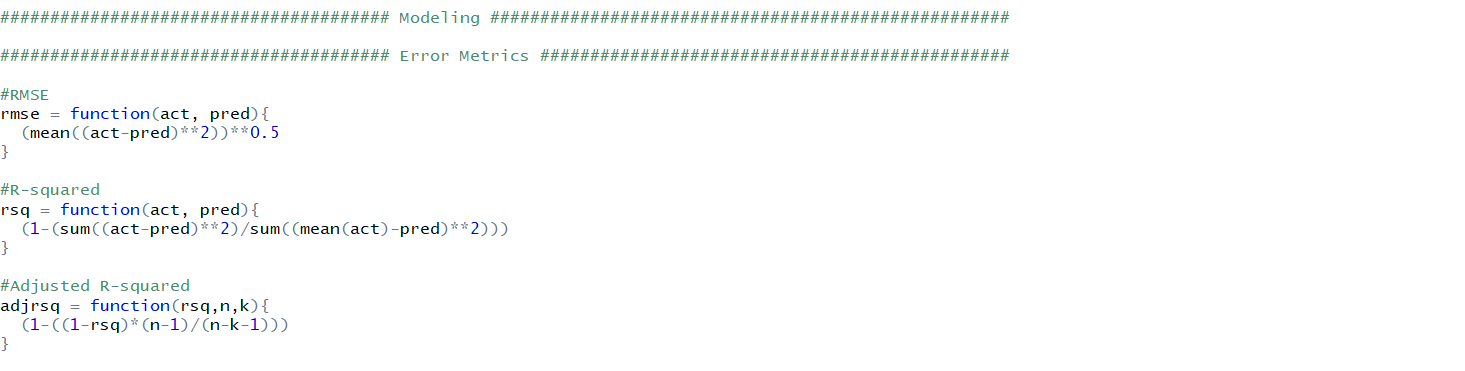
  

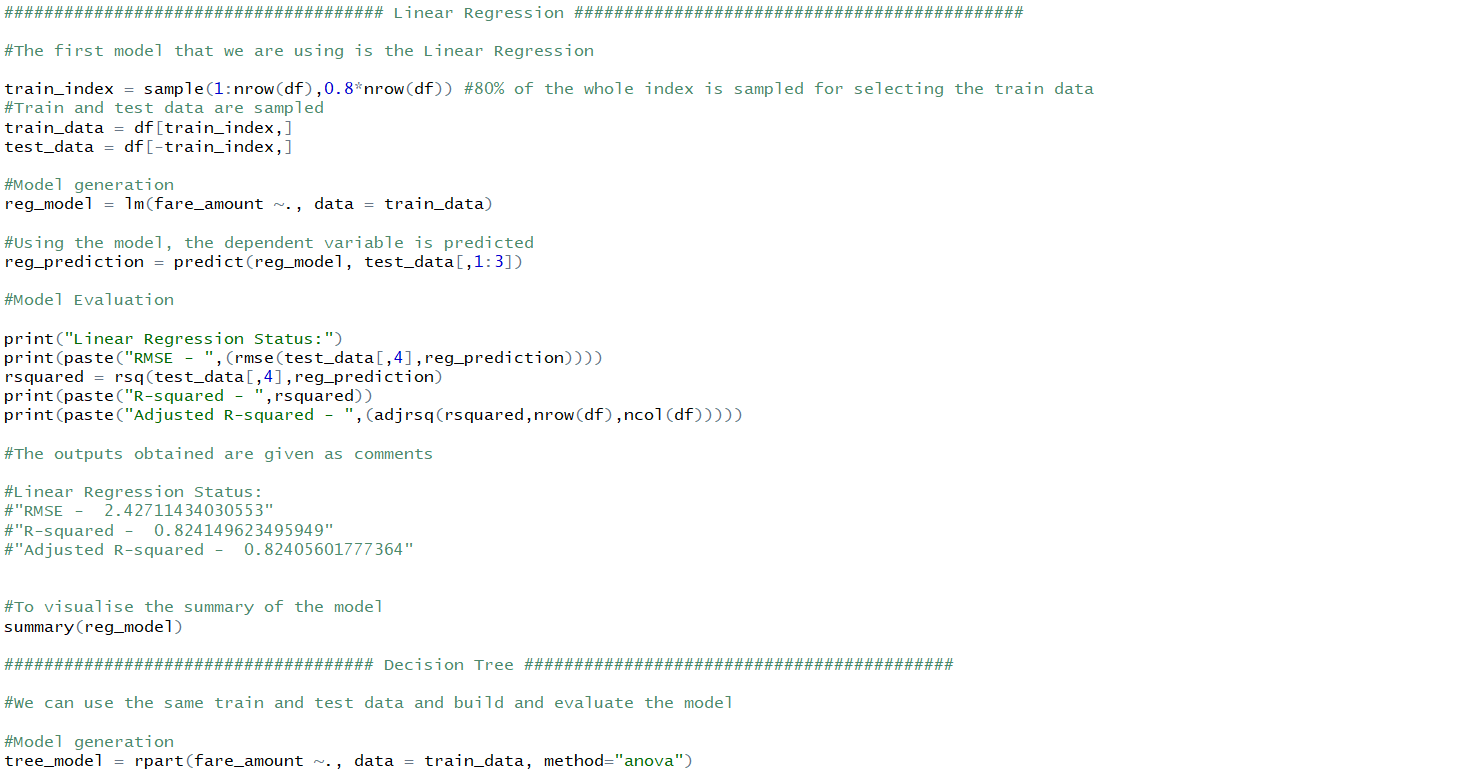
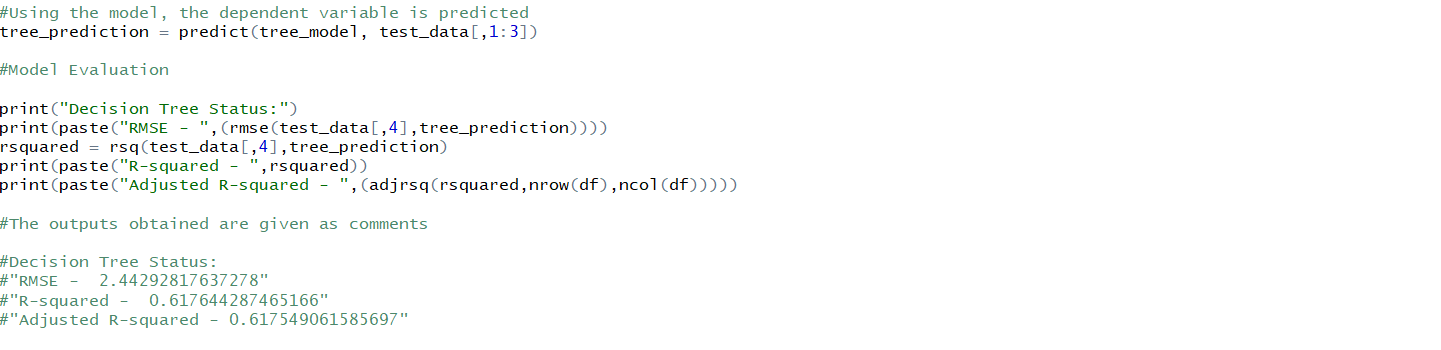
 

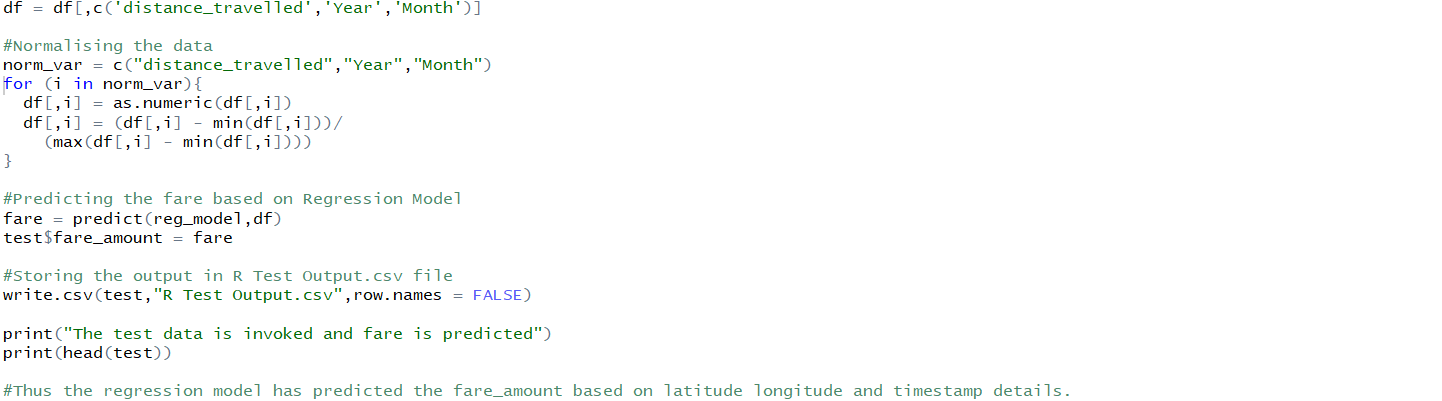
**Complete R Code**



**References**

Matplotlib library attributes – referred from [www.matplotlib.org](http://www.matplotlib.org)

Ggplot2 library attributes – referred from [www.tutorialspoint.com](http://www.tutorialspoint.com)

Pydotplot library and graphviz for visualising decisiontree – referred from blogs in [www.medium.com](http://www.medium.com)

R programming basic functionalities – referred from rprogramming.net

Small functionalities in R and Python – referred from [www.stackoverflow.com](http://www.stackoverflow.com)

Modeling and evaluation metrics – referred from [www.edwisor.com](http://www.edwisor.com)

Error metrics choosing techniques – referred from [www.medium.com](http://www.medium.com)

Importance of error metrics – referred from [www.analyticsvidhya.com](http://www.analyticsvidhya.com)