**Cab Fare Prediction Project**

**Arnav Ashank**

24 December 2019

## Contents

## Introduction----------------------------------------------------------------------------------------3

Problem Statement------------------------------------------------------------------------------------------------3

Data Overview------------------------------------------------------------------------------------------------------3

Variables---------------- ---------------------------------------------------------------------------------------------4

## Methodology---------------------------------------------------------------------------------------5

Cleaning the data----------------------------------------------------------------------------------------------------5

Missing Value Analysis---------------------------------------------------------------------------------------------5

Exploratory Data Analysis ----------------------------------------------------------------------------------------6

Outlier Analysis -----------------------------------------------------------------------------------------------------9

Feature Engineering------------------------------------------------------------------------------------------------11

## Assessing the data--------------------------------------------------------------------------------13

Data Wrangling-------------------------------------------------------------------------------------------------------13

Feature Scaling-------------------------------------------------------------------------------------------------------14

## Modeling the data--------------------------------------------------------------------------------15

Building the model--------------------------------------------------------------------------------------------------15

R-------------------------------------------------------------------------------------------------------------------------15

Decision Tree---------------------------------------------------------------------------------------------------------15

Random Forest-------------------------------------------------------------------------------------------------------16

Linear Regression----------------------------------------------------------------------------------------------------16

XGBoost Regression-------------------------------------------------------------------------------------------------17

Support Vector Machines------------------------------------------------------------------------------------------17

Python------------------------------------------------------------------------------------------------------------------19

## Conclusion------------------------------------------------------------------------------------------22

Model Evaluation----------------------------------------------------------------------------------------------------22

Running scripts from DOS prompt-------------------------------------------------------------------------------22

## 

**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 Overview

Our task is to build a regression model which will predict the fare for the given test dataset. Given below is a sample of the data set that we are using to predict the losses:

Table 1.1: Cab Fare Prediction Sample Data (Columns: 1-5)

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

|  |  |
| --- | --- |
| dropoff\_latitude | passenger\_count |
| 40.712278 | 1 |
| 40.782004 | 1 |
| 40.750562 | 2 |
| 40.758092 | 1 |
| 40.783762 | 1 |

Table 1.2: Cab Fare Prediction Sample Data (Columns: 6-7)

As we can see that there are 6 predictor variables using which we have to predict the fare amount. They are:

|  |  |
| --- | --- |
| Predictor Variables | |
| S.no | Variable |
| 1 | fare\_amount |
| 2 | pickup\_datetime |
| 3 | pickup\_longitude |
| 4 | pickup\_latitude |
| 5 | dropoff\_longitude |
| 6 | dropoff\_latitude |
| 7 | passenger\_count |

As the dependent variable is continuous type so this is a regression type problem.

**1.3 Variables**

There are 7 variables in our data in which 6 are independent variables and 1 (fare amount) is dependent variable. Variable Information:

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.
7. fare\_amount – variable indicating fare amount for that journey.

**Chapter 2**

## Methodology

## 2.1 Pre Processing of the data

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 as **Exploratory Data Analysis**. In this project we look at the continuous and categorical variables in the dataset. Also we will go through the missing value and outlier analysis.

## 2.1 Cleaning the Data

Here we cleaned the data. First here we have converted the pickup date time into date time format removed and removed any value which has absurd as it is not possible to impute date time for a given record. Then we have converted the fare amount column to numeric type removing all characters from it if any. We have removed all those latitudes having value greater than 90 and less than -90,removed all those records having zero passengers or having passengers greater than 6.

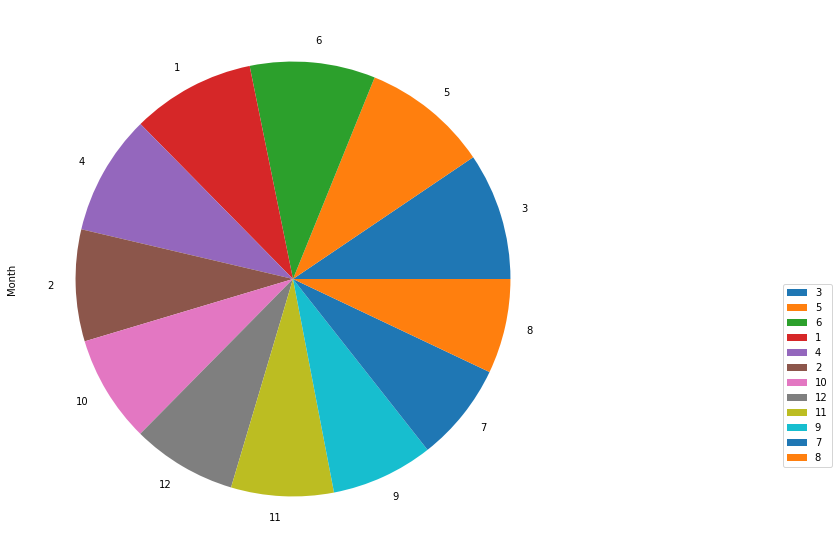
## 2.2 Missing Value Analysis

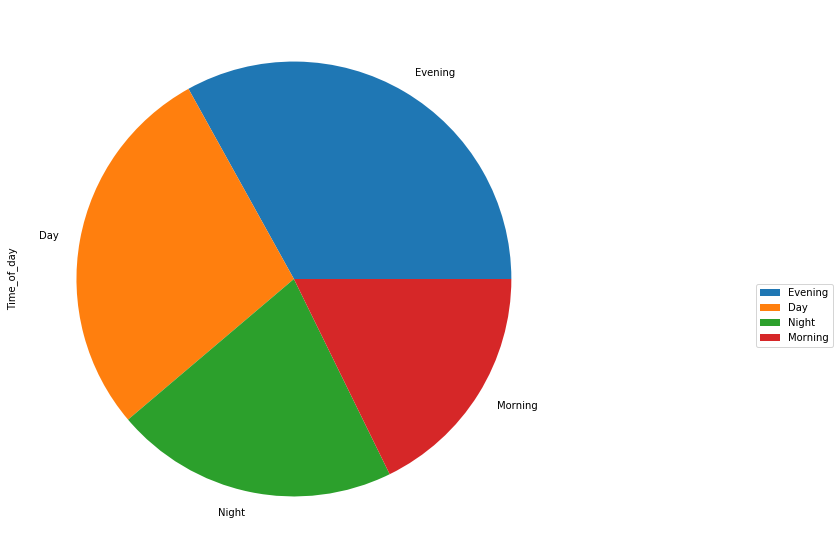
Here we first count the null values present in any variable including the dependent variable. If the null values are more than 30 percent then it won’t be wise to impute or drop it. Otherwise we can either impute the null values or simply drop that record from the dataset. Here we have imputed the missing values using the mean, median and KNN method.

Here we have first replaced a known value with null in the dataset and stored it in a variable. Next we have imputed the value using all the methods mean, median and KNN. After imputing the missing values using all the methods each of the imputed value is compared with the actual value stored originally and the method which imputes the value closest to actual value is used for imputing missing values of that particular variable. In python this procedure has been performed using user defined function and loops, and in R it showing memory error so only user defined function has been used and no loops.

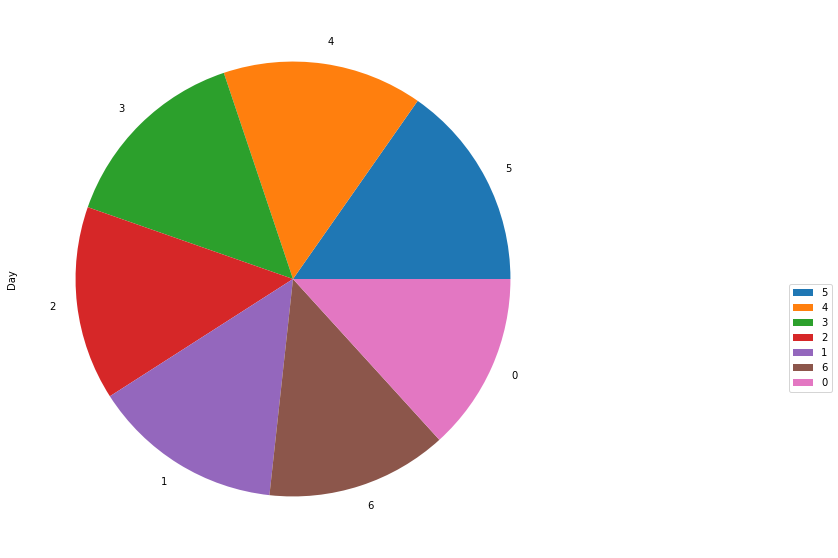
After performing the missing value analysis the data is cross checked so that no null values remain leftover. Also we checked the data for any absurd value and removed them accordingly.

## 2.3 Exploratory Analysis

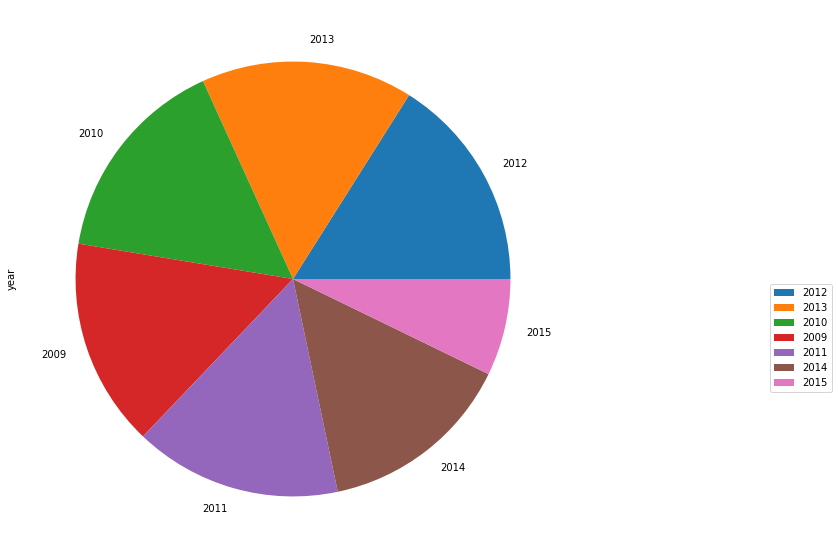


From the above pie chart each of the month appear to have equal distribution in the dataset.

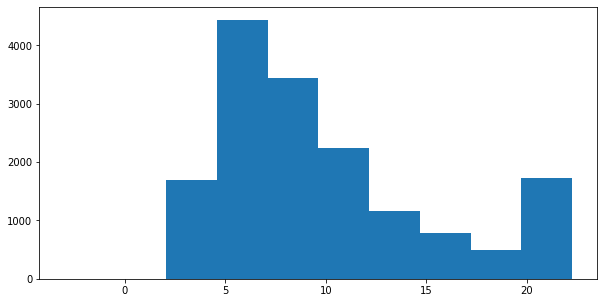
From the above pie chart it is clear that majority of rides are in Evening followed by Day and Night.



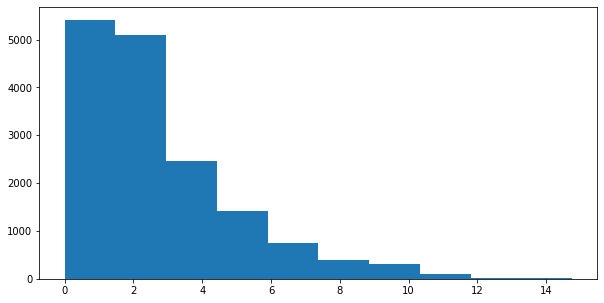
From the above pie chart for day of the week each day appears to have almost equal in the dataset.



From the above pie chart the data seems to contain equal proportion from each year.



From the histogram above of fare amount it varies between 2 to 23 approximately.

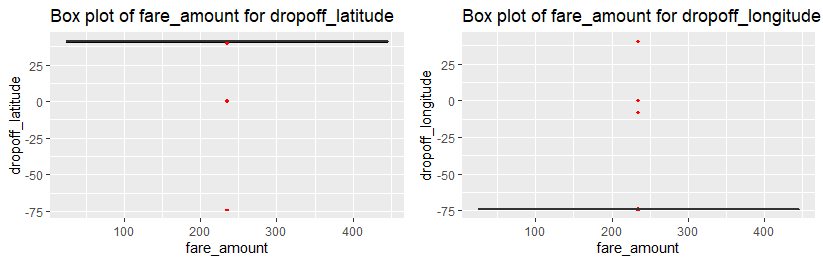


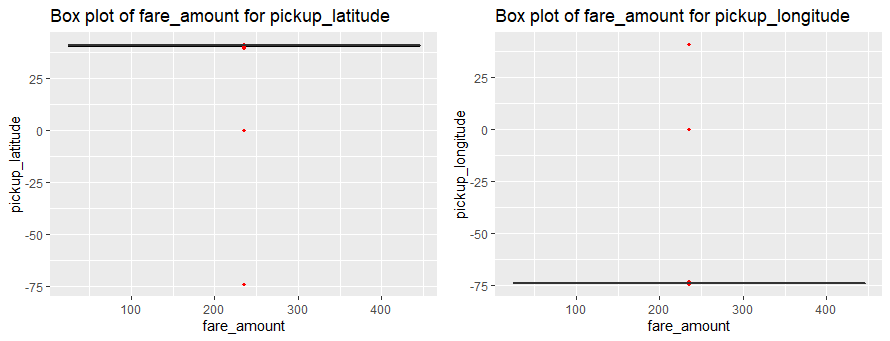
From above histogram it is clear that the distance is right skewed and varies from 0 to 15 approximately.

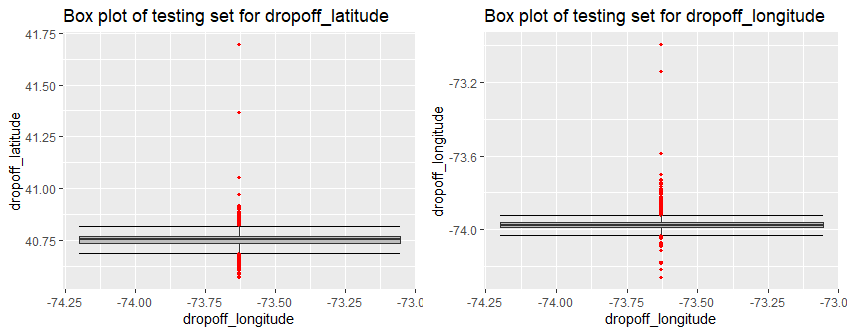
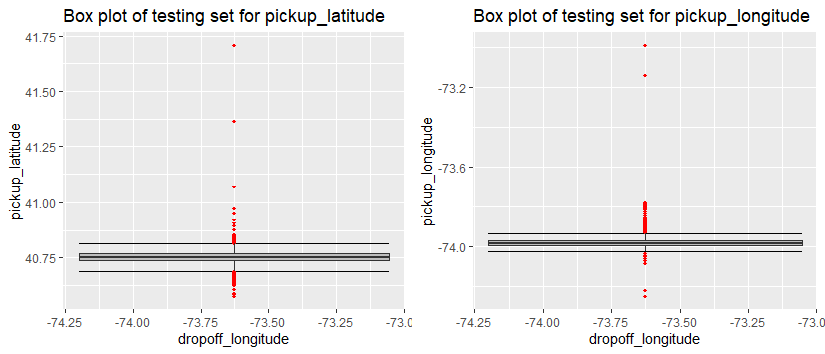
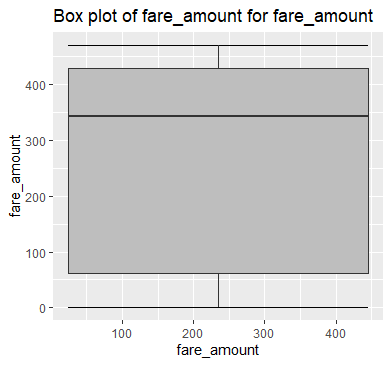
## 2.4 Outlier Analysis

Pie chart-2

From the histograms obtained for continuous variables it has been found that distance has skewed distribution. To get a clear confirmation for outliers in other continuous variables box plot analysis has been done for each continuous variable. Here the variables who have data points outside the whiskers are considered to have outliers. Below are shown box plot for each continuous variables.







From the above box plots it is clear that outliers are present in pickup latitude, pickup longitude, drop off latitude, drop off longitude and fare amount of training set. In the testing set also outliers are present in pickup latitude, pickup longitude, drop off latitude, drop off longitude.

So we need to remove these outliers. Here flooring and capping method has been used to remove the outliers. In this method we compare each value of a variable with two values

Min = q25-1.5\*iqr

Max = q75+1.5\*iqr

Where q25 = 25th percentile of that variable

q75= 75th percentile of that variable

iqr = inter-quartile range of that variable.

If a value exceeds Max then it is replaced with Max similarly when a value is less than Min it is replaced with min. Thus outliers are removed from every variable.

## 2.5 Feature Engineering

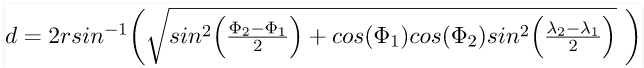
Here we have added new columns in the dataset. From the pickup\_datetime column we have extracted date, year, month, hour and day and created a column containing information regarding each of them. From the hour column we have created a column called time of day. Here we clubbed the hours in the following way

* 3am to 9am as Morning
* 9am to 3pm as Day
* 3pm to 9pm as Evening
* 9pm to 3am as Night

Thereafter we have added a new column called distance travelled by the cab for a ride sung the haversine formula.

Haversine formula:-

The **haversine formula** determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general **formula** in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.



Where is the longitude and is latitude of the two points.

R = radius of the earth

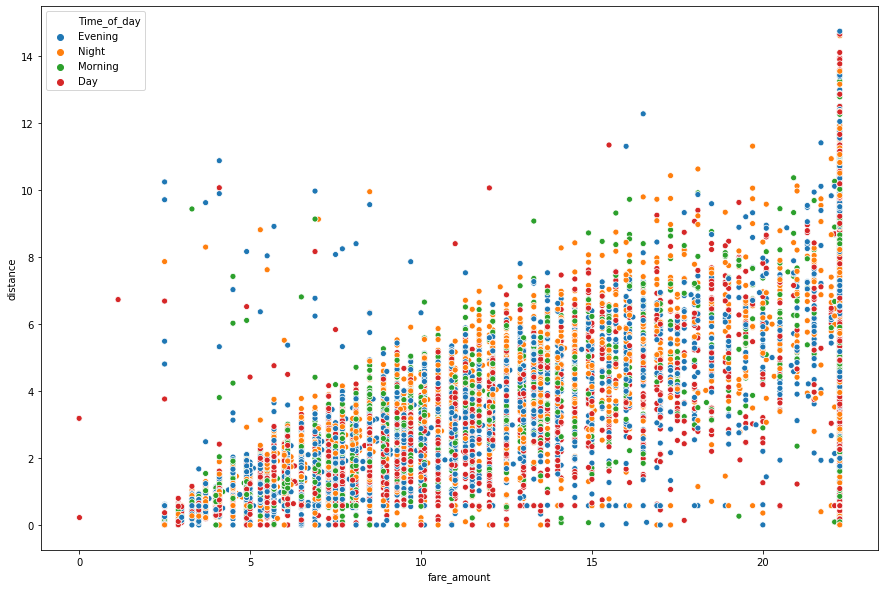
There after we have removed all the latitude and longitude from both test and training set.

**Chapter 3**

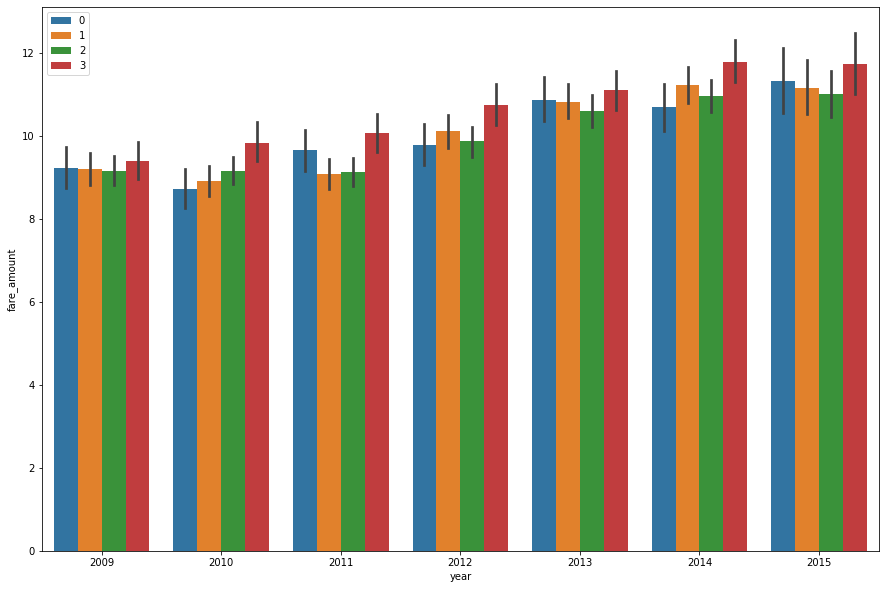
**Assessing the Data**

## 3.1 Data Wrangling

Here we have used multivariate plot to analyze how and how distance affects the cab fare.



Here in the above figure a scatter has been drawn between distance and fare amount with hue as time of day. The scatter plot appears to be increasing linearly. Majority of cab rides occur in the evening followed by day and night.



In the figure above a bar plot is drawn between fare amount and year. Time of day has been encoded here as 0-Morning, 1-Day, 2-Evening, 3-Night. The figure clearly shows that cab fare has increased from 2009 to 2015. Also cab fare in the night is higher than the rest of the day, which usually the case is.

## 3.2 Feature Scaling

From the histogram of distance shown in the previous chapter it is clear distance is right skewed, so it is required to be normalized such that the execution time of various algorithms can also be reduced. Thus to overcome this issue we have normalized the distance continuous variable.

After normalization all the values in the distance are scaled in range from 0 to 1.

**Chapter 4**

## Modeling the data

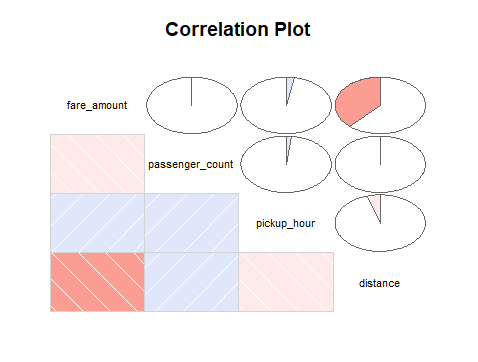
## 4.1 Building the Model

Here we have drawn correlation plot in both R and Python among all continuous variables to find any correlation among different continuous variables. A threshold of 0.8 is taken for assuming correlation in the given two variables.

Then we have calculated p-values for all categorical variables using anova in both R and Python. The non-significant categorical variables having p-values greater than 0.05 have been dropped in both R and Python.

## 4.2 R

In R we have plotted the correlation plot first for the continuous variables then removed those variables which have correlation coefficient greater than 0.8 with any of the other variables. In the same way we have removed those categorical variables who have got p-values greater than 0.05.



It has been found that distance has correlation coefficient 0.79 with fare amount. None of the categorical variables have got p-values greater than 0.05. Since regression is numerical analysis so we need numeric type data rather than factor type data. So we have added dummies for each categorical variable. After adding dummies one dummy variable from category has been dropped because the dummies of a categorical variable are linearly dependent and add up to one. So one dummy variable is dropped from each category to avoid any multi collinearity.

After this we split the entire dataset into two parts training and testing set in 80:20 ratio i.e. 80 percent is training dataset and 20 percent is testing set using random sampling as this is a regression problem.

## 4.2.1 Decision Tree

It is a predictive model based on a branching series of Boolean tests. It can be used for both regression and classification. There are different types of decision trees that can be used in machine learning algorithms.

It uses a model to predict a variable. It is like a flowchart structure. Each leaf/node represents an attribute or a class label. Decision tree is a rule. Each branch connects nodes with AND and multiple branches are connected by OR. It first selects a node and depending upon different categories present in that variable it will split. It will continue splitting until it covers the entire dataset. The table shows the Error metric RMSE of the Decision tree and r-squared.

|  |  |
| --- | --- |
| Error Metric | DT |
| RMSE(TRAIN) | 140.175 |
| RMSE(Test) | 141.625 |
| R2 | 0.324 |

R-squared is basically calculation of goodness of fit of the model i.e. how much the developed model fits the dataset.

## 4.2.2 Random Forest

Random forest is an ensemble that consists of many decision trees. It builds different decision trees using different observations from the same dataset to improve accuracy and reduce weak learners hence it is called an ensemble technique.

It combines Bruimann’s bagging idea and the random selection of features. It feeds the error of one decision tree to another so as to improve accuracy. It randomly selects features to build a tree. It can be used for both regression and classification purpose.

For prediction a new sample is pushed down the tree. It is assigned the label of training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble and the average of all trees is reported as random forest prediction. In case of regression it uses mean of all trees to predict an output. The table below shows the error metric RMSE of the Random Forest and r-squared.

|  |  |
| --- | --- |
| Error Metric | RF |
| RMSE(TRAIN) | 96.716 |
| RMSE(Test) | 140.1306 |
| R2 | 0.339 |

## 4.2.3 Linear Regression

Linear regression model is basically a statistical model unlike decision tree and random forest which are machine learning models. In machine learning the model stores patterns of every variable (in the form of rules) whereas in statistical models there are coefficients of variables which are used to calculate the test data. It can also be used for imputation.

Here we first used variance inflation factor to detect multi collinearity but none of the variables were correlated. After developing the model we found that very few of the variables appear to be significant, so we kept them and removed all other variables and again performed linear regression. It was found that the RMSE of the model doesn’t change much and r-squared also didn’t vary. So we decided to stick to the previous set of variables. The table below shows error metrics and r-squared of both linear regression models with original dataset and adjusted dataset.

|  |  |  |
| --- | --- | --- |
| Error Metric | LR | Adj LR |
| RMSE(TRAIN) | 158.812 | 159.053 |
| RMSE(Test) | 158.168 | 157.326 |
| R2 | 0.1486 | 0.1478 |

## 4.2.4 XGBoost Regression

Gradient boosting is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(machine_learning)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function). Like other boosting methods, gradient boosting combines weak "learners" into a single strong learner in an iterative fashion. It is easiest to explain in the least-squares [regression](https://en.wikipedia.org/wiki/Regression_analysis) setting, where the goal is to "teach" a model {\displaystyle F}to predict values{\displaystyle {\hat {y}}=F(x)} by minimizing the [mean squared error](https://en.wikipedia.org/wiki/Mean_squared_error). The table below shows the error metric RMSE of the XG boost and r-squared.

|  |  |
| --- | --- |
| Error Metric | XGBoost |
| RMSE(TRAIN) | 132.17 |
| RMSE(Test) | 135.43 |
| R2 | 0.383 |

## 4.2.5 Support Vector Machines

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), support-vector machines (SVMs, also support-vector networks[[1]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-CorinnaCortes-1)) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. In simple regression we try to minimize the error rate. While in SVR we try to fit the error within a certain threshold.

Support Vector Regression (SVR) works on similar principles as Support Vector Machine (SVM) classification. One can say that SVR is the adapted form of SVM when the dependent variable is numerical rather than categorical. A major benefit of using SVR is that it is a non-parametric technique. Unlike SLR, whose results depend on Gauss-Markov assumptions, the output model from SVR does not depend on distributions of the underlying dependent and independent variables. Instead the SVR technique depends on kernel functions. Another advantage of SVR is that it permits for construction of a non-linear model without changing the explanatory variables, helping in better interpretation of the resultant model. The basic idea behind SVR is not to care about the prediction as long as the error (ϵi) is less than certain value. This is known as the principle of maximal margin. This idea of maximal margin allows viewing SVR as a convex optimization problem. The regression can also be penalized using a cost parameter, which becomes handy to avoid over-fit. SVR is a useful technique provides the user with high flexibility in terms of distribution of underlying variables, relationship between independent and dependent variables and the control on the penalty term. The table below shows error metric RMSE and r-squared on training and testing set.

|  |  |
| --- | --- |
| Error Metric | SVR |
| RMSE(TRAIN) | 146.79 |
| RMSE(Test) | 152.47 |
| R2 | 0.2136 |

Summarizing the results into a single table:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Metric | DT | RF | LR | SVR | XGBoost |
| RMSE(TRAIN) | 140.175 | 96.716 | 158.812 | 146.79 | 132.17 |
| RMSE(Test) | 141.625 | 140.13 | 158.168 | 152.947 | 135.43 |
| R2 | 0.324 | 0.339 | 0.1486 | 0.2136 | 0.383 |

From the table above it is clear that of the given algorithms XGBoost algorithm has least RMSE and maximum r-squared of about 0.383.

## 4.3 Python

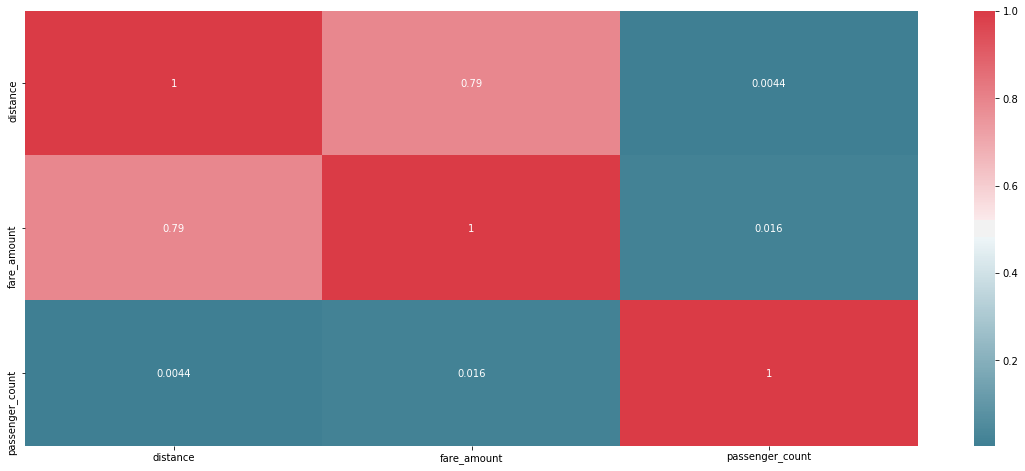
Here we have plotted the correlation plot first for the continuous variables then removed those variables which have correlation coefficient greater than 0.8 with any of the other variables. In the same way we have removed those categorical variables who have got p-values greater than 0.05.It has been found that distance has correlation coefficient of 0.79 with fare amount. All the categorical variables have got p-values less than 0.05. Thus in the final list for independent variables all these variables have been included. Since regression is numerical analysis so we need numeric type data rather than factor type data. So we have added dummies for each categorical variable. After adding dummies one dummy variable from category has been dropped because the dummies of a categorical variable are linearly dependent and add up to one. So one dummy variable is dropped from each category to avoid any multi collinearity.

After this we split the entire dataset into two parts training and testing set in 80:20 ratio i.e. 80 percent is training dataset and 20 percent is testing set using random sampling as this is a regression problem.

Thereafter different algorithms viz Decision tree, Random forest, Linear regression, XGBoost and support vector machine, have been used to create machine learning model and evaluate it’ s error metrics and r-squared. R-squared is basically calculation of goodness of fit of the model i.e. how much the developed model fits the dataset.

We have first developed model using the above mentioned algorithms and calculated RMSE (root mean squared error) for each of the model on testing and training set. Also r-squared has been calculated for each of them.

In linear regression model, after developing the model we found that very few of the variables appear to be significant, so we kept them and removed all other variables and again performed linear regression. It was found that the RMSE of the model further decreased along with r-squared. So we decided to stick to the previous training and testing set.



The table below shows the RMSE for each of the machine learning models before tuning along with R-squared value.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Metric | DT | RF | LR | SVR | XGBoost |
| RMSE(TRAIN) | 3.374 | 1.159 | 3.267 | 2.89 | 2.919 |
| RMSE(Test) | 3.28 | 2.94 | 3.095 | 3.29 | 2.84 |
| R2 | 0.632 | 0.7032 | 0.673 | 0.630 | 0.723 |

From the table above it is clear that of the given algorithms XGBoost has least RMSE and maximum r-squared of about 0.723 before tuning it’s hyper parameters. Since we generally look for a value of r- square close to one so here we decided to tune the hyper parameters of every algorithm except SVR and XGBoost as both of them were taking too much time as compared to other models for tuning and session used to expire till then.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Error Metric | DT tuned | RF tuned | LR | SVR | XGBoost |
| RMSE(TRAIN) | 2.96 | 2.761 | 3.267 | 2.89 | 2.919 |
| RMSE(Test) | 2.97 | 2.88 | 3.095 | 3.29 | 2.84 |
| R2 | 0.6971 | 0.7161 | 0.673 | 0.630 | 0.723 |

From the table above after tuning of Decision tree model and Random forest model, their respective R squared has increased but is still less than that of XGBoost model.**Chapter 5**

## Conclusion

## 5.1 Model Evaluation

From comparison of error metrics of above machine learning models in R and Python it can be concluded that in R the XGBoost model is the best as it has got low RMSE and high r-squared among all the given models.

Similarly in Python the XGBoost model out performs all the given models after tuning it’s hyper parameters using Random Search CV.

A quite interesting fact is visible here that there’s huge difference in RMSE and R squared among same models in R and Python (untuned algorithms). It may be due to the fact that R was executed on my local machine and python on Google colab. It clearly reveals the hardware dependency of Machine learning models. In python Tuning of hyper parameters has been performed for decision tree and random forest. For SVR and XGBoost the execution time was so long that runtime session always expires. Final results from each R and Python have been exported in a csv file with prediction of each model in a different column.

## 5.2 Running Scripts from command line

Steps for running ipynb file from DOS prompt:-

1. The file provided has been executed in google colab, so for executing it on a local machine first open the ipynb file in a Jupyter notebook delete first three and last cell, save it and close the file.

2. Open command prompt

3. Change to the directory where the ipynb file by typing cd /d <filepath> press enter

4. Type jupyter nbconvert --execute <filename.ipynb>

The output of above file will be saved in a html file at the same location.

Steps for running R file from DOS prompt:-

1. Find the path to R.exe or Rscript.exe on the computer.

2. Find the path to R file.

3. Open Notepad and combine paths together like this

“C:\Program Files\R\R-3.4.3\bin\R.exe” CMD BATCH C:\Users\myusername\Documents\R\Send\_Outlook\_Email.R

4. Save as file with extension .bat

5. Run that batch file to execute R script.