A Project On

Cab Fare Prediction

By

ABHISHEK KANKATI

on 18/11/2019

Table of Contents

Chapter I

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.

In this project, our task is to predict the fare for a cab ride in the city in a 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. After selecting the best model, we should find the *fare_amount* of the given test data.

1.2 Variables

Number of Attributes: 6
Missing Values: Yes
Attribute Information:

- 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
- fare_amount – float indicating the fare amount for the ride

We have a total of 6 predictor variables and 1 target variable. After extracting the new variables and dropping the older ones, we have 7 predictor variables and 1 target variable.

1.3 Sample Data

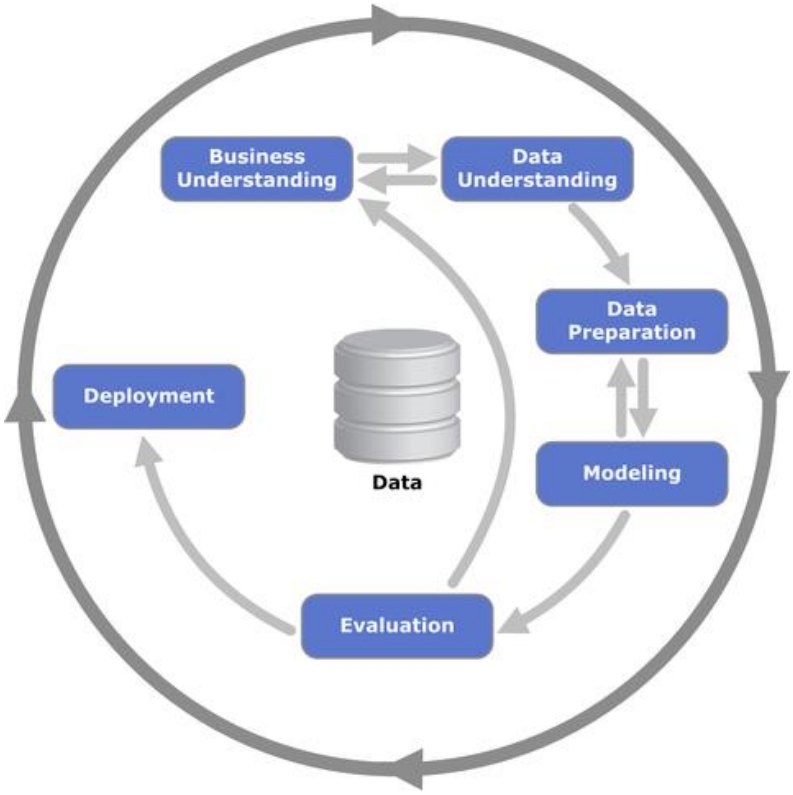
	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1.0
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1.0
2	5.7	2011-08-18 00:35:00 UTC	-73.982738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

By using the latitudes and longitudes, we can find the distance travelled by the passengers and by using the timestamp, we can extract date, day, month, year etc. After extracting the new variables and dropping the old variables, the sample data looks like below.

	fare_amount	passenger_count	year	Month	Date	Day	Hour	distance_travelled
0	4.5	1.0	2009.0	6.0	15.0	0.0	17.0	1.029601
1	16.9	1.0	2010.0	1.0	5.0	1.0	16.0	8.443441
2	5.7	2.0	2011.0	8.0	18.0	3.0	0.0	1.389132
3	7.7	1.0	2012.0	4.0	21.0	5.0	4.0	2.795790
4	5.3	1.0	2010.0	3.0	9.0	1.0	7.0	1.998338

1.4 CRISP DM Process (Cross Industry Standard Process Data Mining)

CRISP-DM is a cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining project. It is a robust and well-proven methodology. It can be explained by the below figure.



1. Business Understanding

This step mostly focuses on understanding the Business in all the different aspects. It follows the below different steps.

a. Identify the goal and frame the business problem.

b. Gather information on resource, constraints, assumptions, risks etc.

c. Prepare Analytical Goal

d. Flow Chart

2. Data Understanding

Data Understanding phase of CRISP DM Framework focus on collecting the data, describing and exploring the data.

Exploring the data involves analyzing the data in hand for

· Dependent and Independent Variable Identification.

· Uni-variate Analysis – Exploring each independent variable

· Bi-variate Analysis – Exploring the different combination of two or more variables using Correlation, Chi-Square Test, T-Test, Z-Test etc. This step also involves subcategory analysis of each independent variable on the dependent variable.

· Aggregated data exploration

· Data Quality Check is also performed at the step.

In a ML Implementation, we may get numerous independent variables that may or may not contribute to predict the dependent variable. This step of data understanding at times even gives us a gist of attributes which may be important for predicting the dependent variable.

3. Data preparation

In this step, we prepare and clean the provided data. There are many steps that involves in this step as mentioned below.

a. The first and foremost step being the NA (null values) treatment. Normally the data at hand is not clean and at most of the times our data will have NA. We must identify such values and appropriately fill or impute them. There are many different techniques of NA treatment and there are packages in R and Python which automatically treat such variables based on some default logic. However, it is always good to do it manually, as this way we get to understand the data even further and can replace these NAs with our understanding of business requirement.

Imputing Missing Values

b. The next step would be to treat NAs. This step is equally important as NA treatment and as per my experience, I have below steps for the Null treatment.

i. If the variable is continuous in nature (numeric variable), we can use Mean/Median/Mode/KNN imputation technique for the missing value treatment.

ii. If the variable is categorical in nature, we can impute the NAs with "Unavailable" as these null values or Unavailable values may contribute significantly to the Model creation and we do not want to lose any important attribute.

Outlier Treatment

c. Outliers are the values in continuous variables that are inconsistent (may be very far from the mean) with data. These outliers will drastically impact our mean value. So, treating outlier will be our next step post imputing missing values. We have a few packages in R and python to remove the outliers or try to impute them with again mean/median/mode/KNN methods. We can display such variables using boxplot of the attribute.

Feature Scaling

d. Scaling of the data – This step is used to scale up the values of the attribute so that they lie between 0 to 1. With scaling, the range of the variables get reduced and result in a better predicting variable. This could be done on Continuous Variables. If the data is normally distributed, we will use Standardization technique. Otherwise we will go with Normalization technique. If the data is right skewed, we should go for log transformation.

Feature Engineering

e. Feature Engineering: One of the most important steps and can be clubbed as the combination of Feature Transformation and Feature Extraction. In this step, we try to create or extract more attributes from the available attributes with some business sense. The more we explore, the better we can extract with different relations from the existing variables. The examples are below,

i. Create Value Transformation like Square or Cube or even Square-root or Cube root or Log of certain columns, as it has been seen that such derived columns contribute in algorithm then the deriving column.

ii. Variable Creation like creating dummy variables from the categorical variables, Data Split etc. also contribute to Feature Engineering.

Feature Engineering step is one such step which can be explored more and can contribute significantly to the outcome.

Dimensionality Reduction (Feature Selection)

f. Dimensionality reduction is a series of techniques in machine learning and statistics to reduce the number of random variables to consider. It involves feature selection and feature extraction. Dimensionality reduction makes analyzing data much easier and faster for machine learning algorithms without extraneous variables to process, making machine learning algorithms faster and simpler in turn. Assume we have 1000+ variables in our dataset. Methods like Principal Component Analysis and Factor Analysis will help us to find the most important variables that explain our target variable to the best in these 1000

variables and we can use those variables to train our data. Thereby reducing the complexity of our data and increase the model's speed and the performance.

4. Modeling

Once the above steps are done, we have implemented the necessity of machine learning and now we can proceed with the implementation of different ML algorithms. The algorithm to be selected depends completely on the business requirement, available data and the desired outcome. In an ideal situation, we should try different algorithm or combination of algorithm (ensembles) to arrive at our final best algorithm. For our problem statement, I have used linear regression, decision tree, random forest and Gradient Boosting algorithms.

5. Evaluation of the Model

There are many model evaluation techniques like Accuracy, Sensitivity, Specificity, F-Score, R-Squared, Adj R-SQ, RMSE (Root Mean Square Error), MAPE etc.

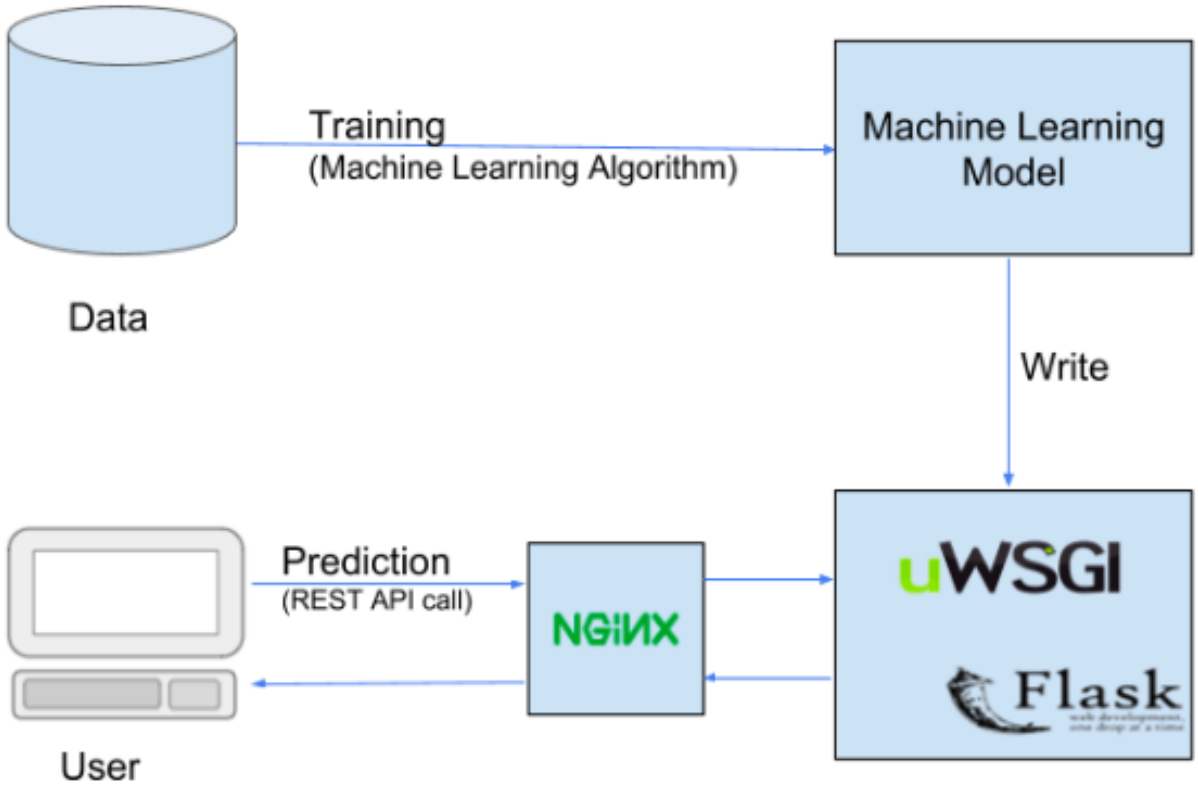
6. Deployment

Finally, once the model is created, tested and evaluated on the Test and Validation data, this is presented to the business (with PPT). The model the undergoes different real time evaluation and testing like A/B Testing and after all the approval process, the code is pushed to the PROD/Live data.

Deployment of machine learning models or putting models into production means making our models available to the end users or systems. However, there is complexity in the deployment of machine learning model, and we have different ways to deploy our model based upon the client requirement or our problem statement. Let us go through one of the easiest ways of deploying a model, by using Flask. Flask is a Python-based microframework used for developing small scale websites. Flask is very easy to make Restful API's using python by putting our machine learning models into production using Flask API.

This method of deployment using Flask has four parts:

1. model.py — This contains code for the machine learning model, let's say to predict cab fare based on the history we have collected from the pilot project.
2. app.py — This contains Flask APIs that receives sales details through GUI or API calls, computes the predicted value based on our model and returns it.
3. request.py — This uses requests module to call APIs defined in app.py and displays the returned value.
4. HTML/CSS — This contains the HTML template and CSS styling to allow user to enter sales detail and displays the predicted sales in the third month.


Training
(Machine Learning Algorithm)
Machine Learning Model
Data
Write
Prediction
(REST API call)
NGINX
uWSGI
Flask
User

Chapter II

METHODOLOGY

2.1 Pre – processing

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

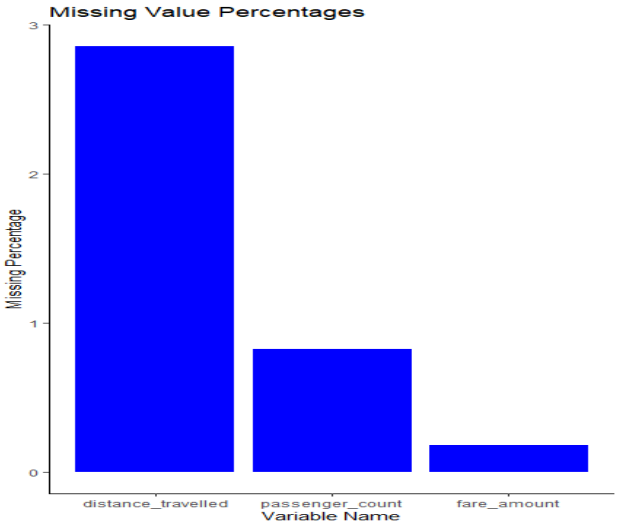
For achieving better results from the applied model in Machine Learning projects the format of the data must be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values, therefore, to execute random forest algorithm null values must be managed from the original raw data set. Another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in one data set, and best out of them is chosen. So, as part of pre – processing techniques, we should explore the data and analyze it, impute the missing values, treat the outliers, feature engineering, feature selection and reduce the dimensions of the dataset by selecting the important features of all the features.

Below are a few observations while exploring the datasets both in train and test datasets. Before proceeding for the pre-processing techniques, the below steps are resolved.

- pickup_datetime should be converted to date type using pandas
- passenger_count should be an int type and any data point less than 1 and greater than 6 can be removed/imputed
- fare_amount should be a float type and any data point less than 0 can be removed/imputed
- pickup_latitude and dropoff_latitude should have values in between -90 to +90 degrees and data point beyond these values can be removed
- pickup_longitude and dropoff_longitude should have values in between -180 to +180 degrees and data point beyond these values can be removed
- By using the co-ordinates of latitude and longitude, we can find the distance between pickup and drop locations
- After the above steps, we'll try to drop a few variables and ensure that all the variables are have appropriate data types

2.2 Missing Value Analysis

Missing values occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions or insights that can be drawn from the data. So, if a variable in a dataset has more than 30% of the missing values, then we can drop that variable. Otherwise these values can be filled by using central statistics or KNN imputation or prediction methods. The missing values for the given data are plotted and is mentioned below.

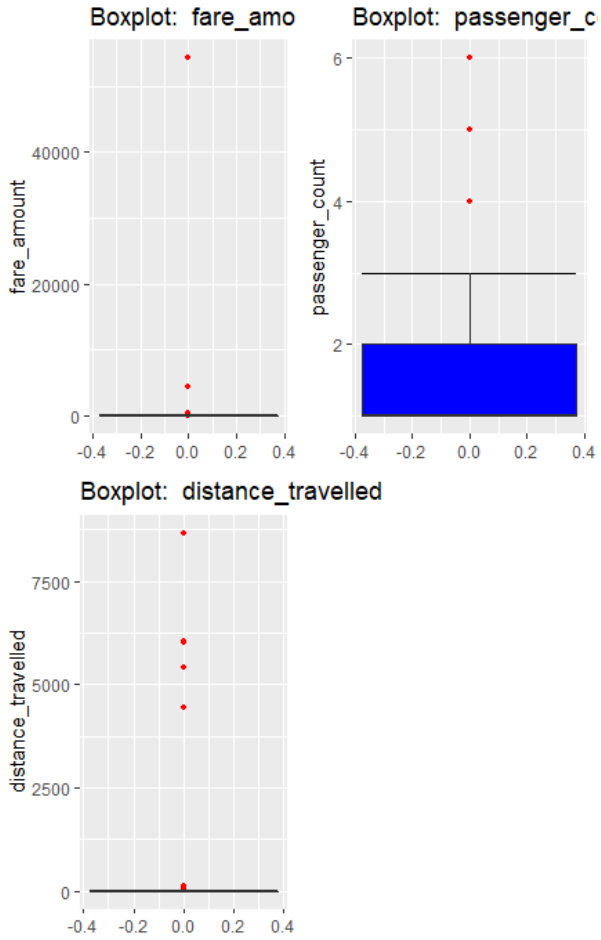


	Variables	Missing_percentage
7	distance_travelled	2.8571429
2	passenger_count	0.8278867
1	fare_amount	0.1805167
3	Year	0.0000000
4	Month	0.0000000
5	Date	0.0000000
6	Hour	0.0000000

From the above plots, we can observe that variable, *distance_travelled* has the highest percentage of the missing values with 2.85%. The missing values are imputed by using KNN method as it is giving the nearest value in our experiment.

2.3 Outlier Analysis

Outliers are the numeric values that are inconsistent with our data. Mean will have a drastic impact because of outliers. Outliers can explain the skewness in the data. So, these outliers are to be detected and should be treated. I have used *boxplot* method to detect the outliers. Then I have replaced them by NA and used KNN imputation technique to impute those missing values. Below are the boxplots plotted for each continuous variable.



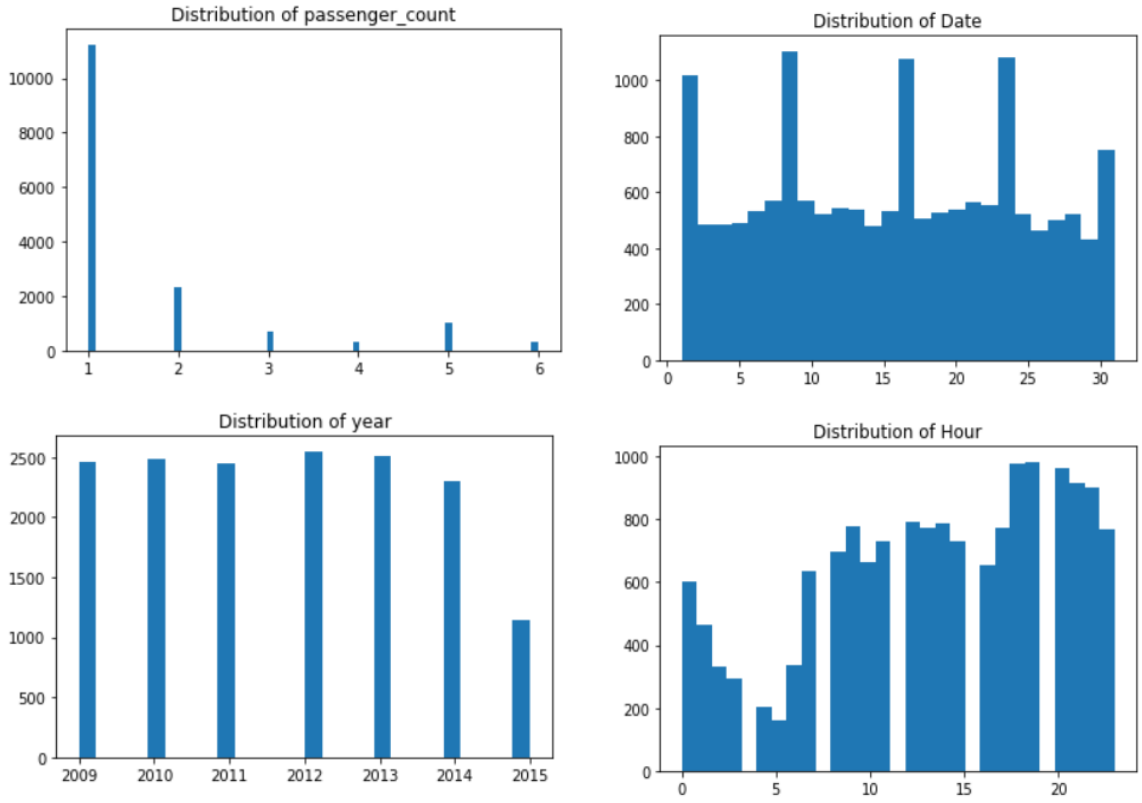
From the above plots it can be concluded that *fare_amount,distance_travelled and passenger_count* has outliers. These outliers are treated and imputed as mentioned above.

2.4 Distribution of the variables

By using various plots, we find the distribution of the variables. This is exploratory data analysis, from which we'll get a basic idea on how the variables are distributed, basic statistical parameters like mean of the variable or if the variable is uniformly or non-uniformly distributed etc.

2.4.1 Distribution of Continuous variables

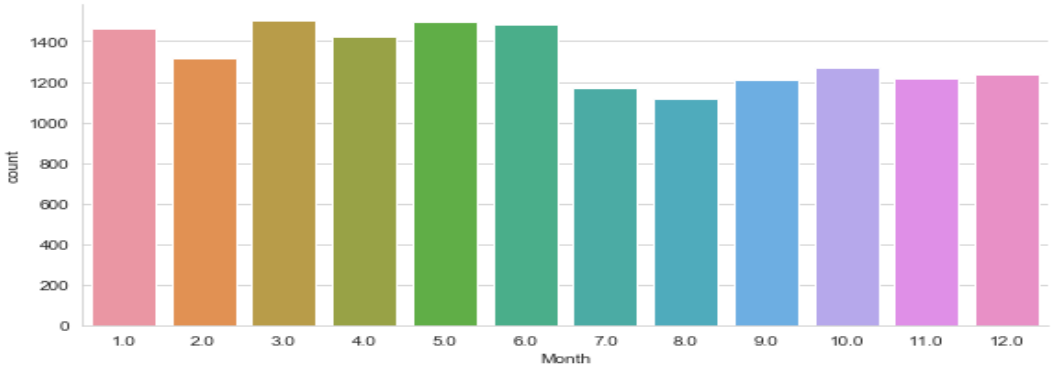
To check the distribution for continuous variables, I have used histograms. The distribution plots are as below,

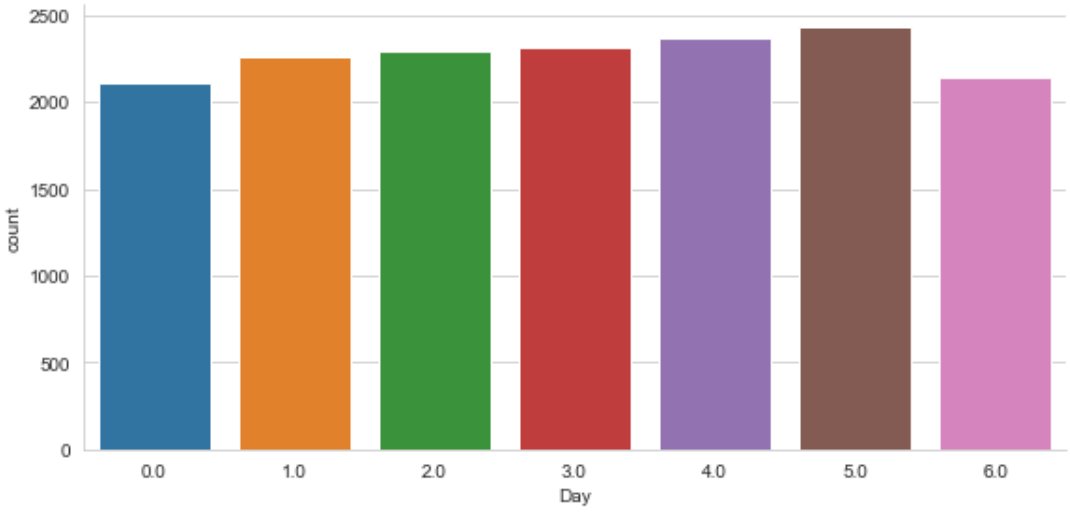


By observing the above plots, it is found that no variable has a normal distribution.

2.4.2 Distribution of Categorical Variables

To check the distribution for categorical variables, I have used bar plots. The distribution plots are as below,

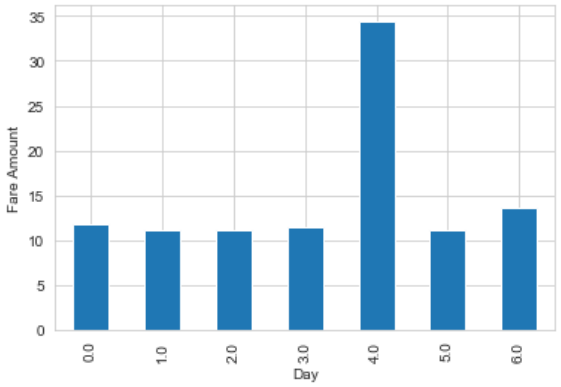


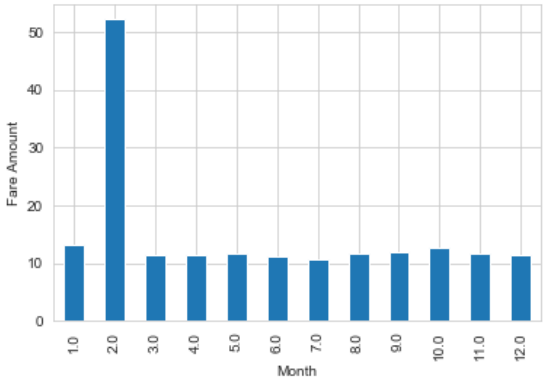


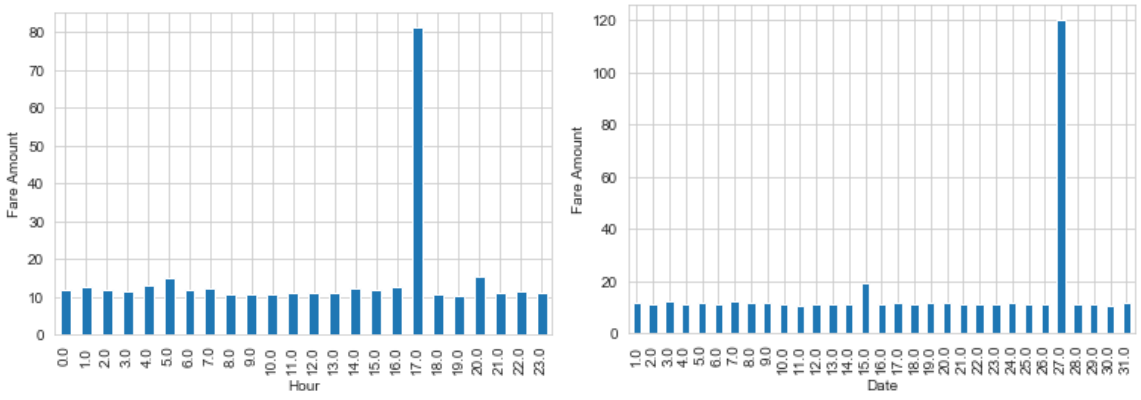
Below are a few observations from the distribution of all the variables.

- Demand of cabs is high on 6th and 5th days in a week and least on 1st day of the week
- Demand of cabs is high in the month of May, March and June respectively and least demand in August
- Cabs are high in demand during evening hours and least demand during early hours of the day
- Single travelled passenger's prefer cabs than with a group of 4/5

Let’s also check the distribution of these variables against our target variable. I have grouped the data with most of the variables and found the *mean* of ***fare_amount*** for each variable. The plots are as below.







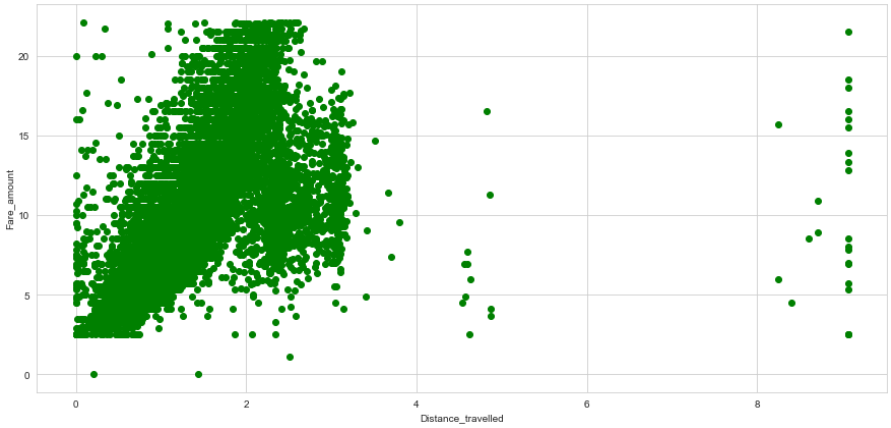
The observations are as below from the above plots.

- The average fare amount is higher on the 5th day of the week
- The average fare amount is higher in the month of February
- Fair amounts are higher between 6 P.M.- 7 P.M. and least at 5 A.M.
- Average fare price is highest on 27th of every month
- The average fare amount is higher at 5 P.M.

2.5 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. By using latitudes and longitudes of both pickup and drop locations, we have found the new variable *distance_travelled* (in kilometers)by using **Haversine** formula in R and by importing ***geodesic*** from the library ***geopy.distance*** and also extracted date, month, day, year and hour from the timestamp variable, *pickup_datetime*.

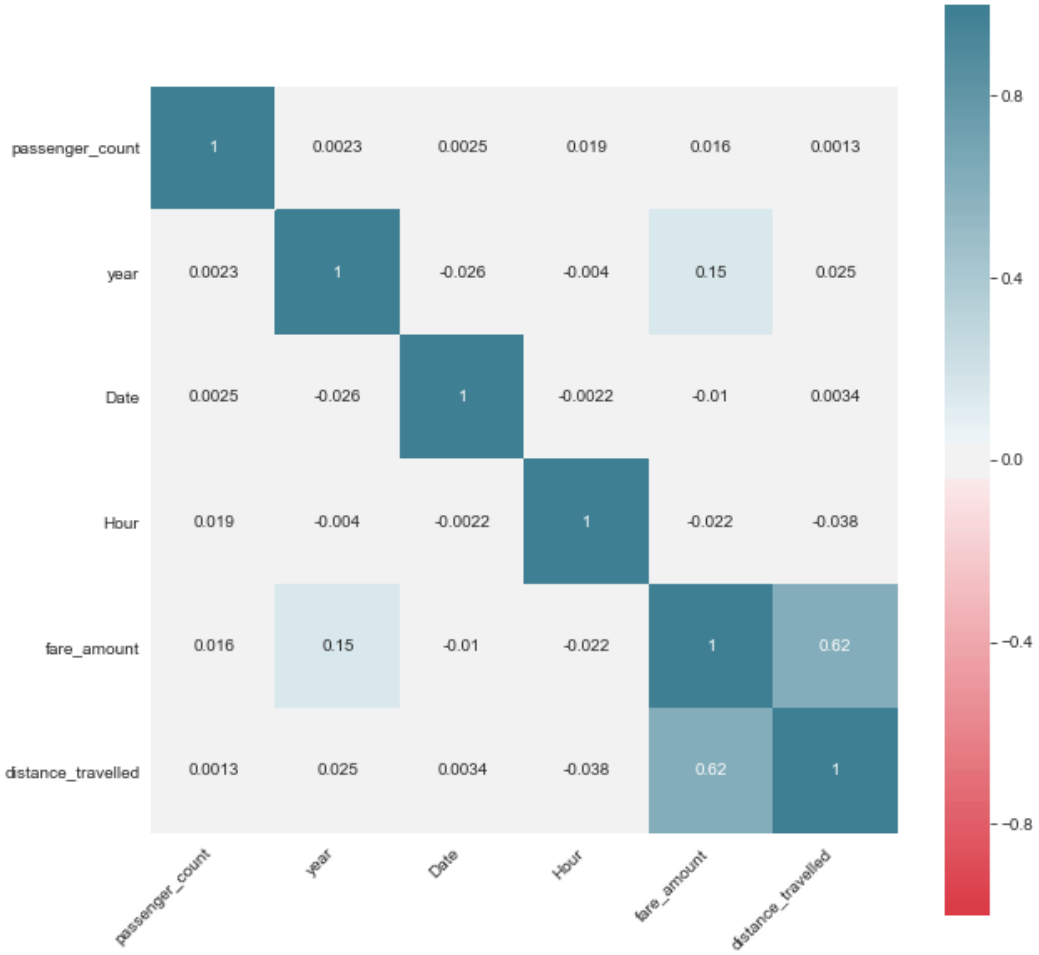
Below is the distribution of new variable *distance_travelled* against our target variable.



2.6 Feature Selection

Feature selection reduces the complexity of our model. The lesser the variables, the higher the performance. It also reduces the over fitting of the model. Multicollinear variables are to be removed if correlation value, r > 0.7.

This can be found by using correlation plot for continuous variables and chi-square test for categorical variables. I have used one-way ANOVA for categorical variables. Below is the correlation plot



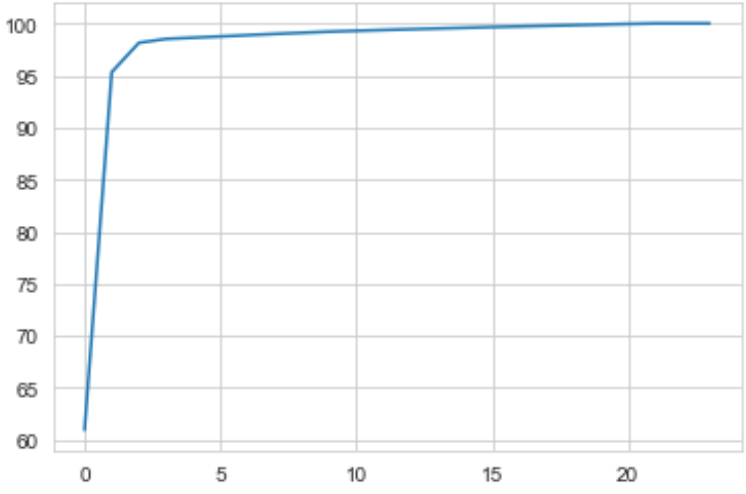
From the plot, it is evident that, no two variables are highly correlated to each other. Hence, we can pass all our variables to our model. Chi-square distribution for the categorical variables is below and conclude that no variable is dependent on another variable.

```
[1] "Year"

        Pearson's Chi-squared test

data:  table(cab$fare_amount, factor_data[, i])
X-squared = 11662, df = 834, p-value < 2.2e-16

[1] "Month"

        Pearson's Chi-squared test

data:  table(cab$fare_amount, factor_data[, i])
X-squared = 1921.4, df = 1529, p-value = 2.38e-11

[1] "Date"

        Pearson's Chi-squared test

data:  table(cab$fare_amount, factor_data[, i])
X-squared = 4477, df = 4170, p-value = 0.000501

[1] "Hour"

        Pearson's Chi-squared test

data:  table(cab$fare_amount, factor_data[, i])
X-squared = 3679, df = 3197, p-value = 4.282e-09
```

2.7 Principal Component Analysis

Principal Component Analysis (PCA) tool that can be used to reduce a large set of variables to a small set that still contains most of the information in the large set. It is used to explain the variance-covariance structure of a set of variables through linear combinations. It is often used as a dimensionality-reduction technique. We've used this technique while programming in python to find the most important variables (as components) which carries the maximum information to drive our target variable.

After creating dummy variables for our categorical variables, the data have 24 components (independent variables) and 15905 observations. After applying PCA algorithm and by plotting a cumulative scree plot, we have found that approximately 7 variables explain almost 96% + of our data. Hence, we have selected those 7 variables and created a new model again.



The sample data after applying dummies to our categorical variables is as below:

	passenger_count	year	Date	Hour	distance_travelled	Month_1.0	Month_2.0	Month_3.0	Month_4.0	Month_5.0	...	Month_11.0	Month_12.0	Day_0.(
0	0.0	2015.0	27.0	13.0	1.200263	1	0	0	0	0	...	0	0	(
1	0.0	2015.0	27.0	13.0	1.230751	1	0	0	0	0	...	0	0	(
2	0.0	2011.0	8.0	11.0	0.481303	0	0	0	0	0	...	0	0	(
3	0.0	2012.0	1.0	21.0	1.085078	0	0	0	0	0	...	0	1	(
4	0.0	2012.0	1.0	21.0	1.853612	0	0	0	0	0	...	0	1	(

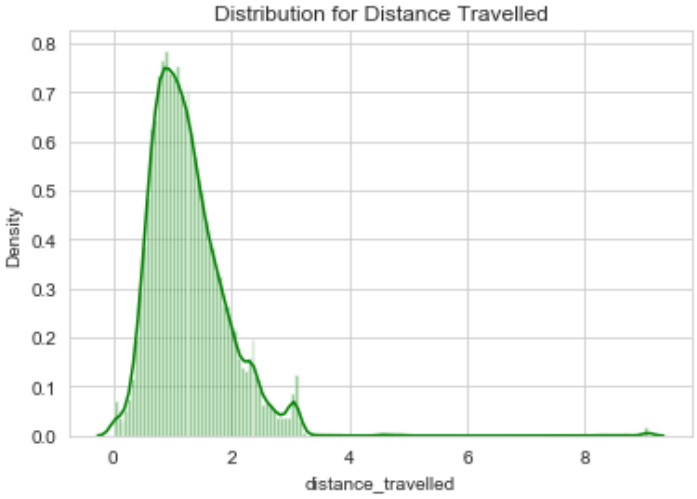
5 rows × 25 columns

The 25 columns include the target variable, the dummy variables and the other predictor variables.

2.8 Feature Scaling

Feature scaling is a method used to standardize the range of independent variables or features of data. It is also known as data normalization and is generally performed during the data pre-processing step. Since the raw data we receive from the client might have **n** number of variables with wide ranging values, our model will give high importance to the variable with high values. So, all the independent continuous variables are to be scaled by using Standardization or Normalization techniques. If our data is normally distributed, we will use Standardization technique. As our data is not normally distributed, we have used Normalization technique to scale our variables. If any variable is right skewed then, we should apply log and transform the variable. In some machine learning algorithms, objective functions may not work properly without normalization thereby leading to inaccurate results. Hence scaling of the variables is a mandatory.

Below is the plot for *distance_travelled* variable after applying the log. It turns to the normal distribution after applying the log.



Chapter III

Modelling

3.1 Modelling and Model Selection

After applying all the pre-processing techniques such as missing value analysis, outlier analysis, feature engineering, feature selection and feature scaling on our data, we should develop a regression model to predict our target variable. Our target variable is a continuous variable, hence the models that we chose are Linear regression, Decision tree, Random forest and Gradient Boosting methods.

The whole data is divided into test and train data. We will train our model using train data and then apply and validate that model on our test data. Also, we have various metrics to validate our model like RMSE, MAPE, R-Squared, Adj R-Squared etc.

3.2 Linear Regression

Linear regression is one of the statistical models that is used for prediction (regression only). It was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

Linear regression is a linear model, i.e., a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

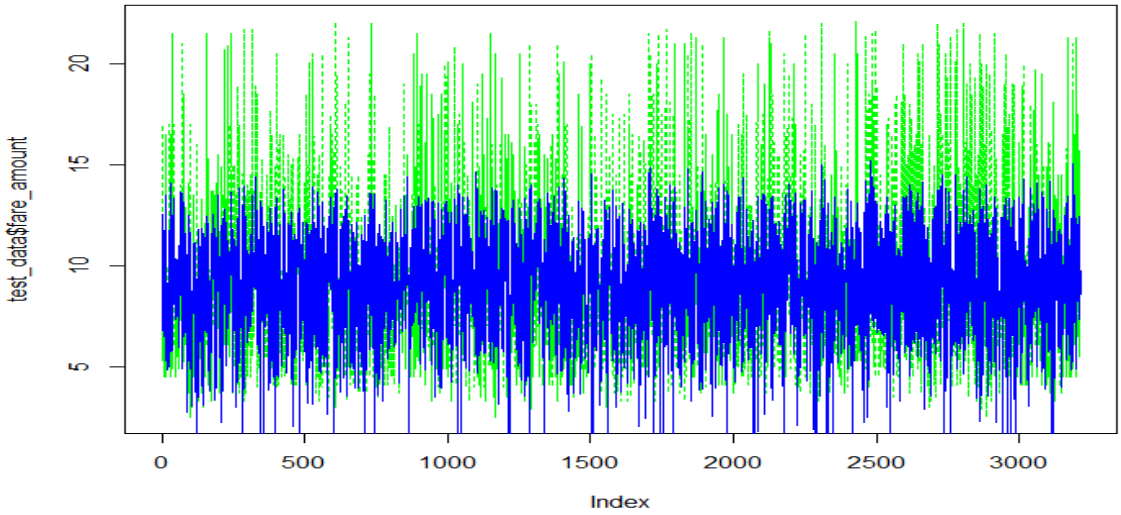
When there is a single input variable (x), the method is referred to as *simple linear regression*. When there are multiple input variables, it refers to the method as *multiple linear regression*.

Parameter	R	Python
RMSE	2.242702	3.18132
R-Squared	0.7173	0.788309

Below is the line plot between actual and the predicted values.

Green line is the Actual values

Blue line is the Predicted values



3.3 Decision Tree

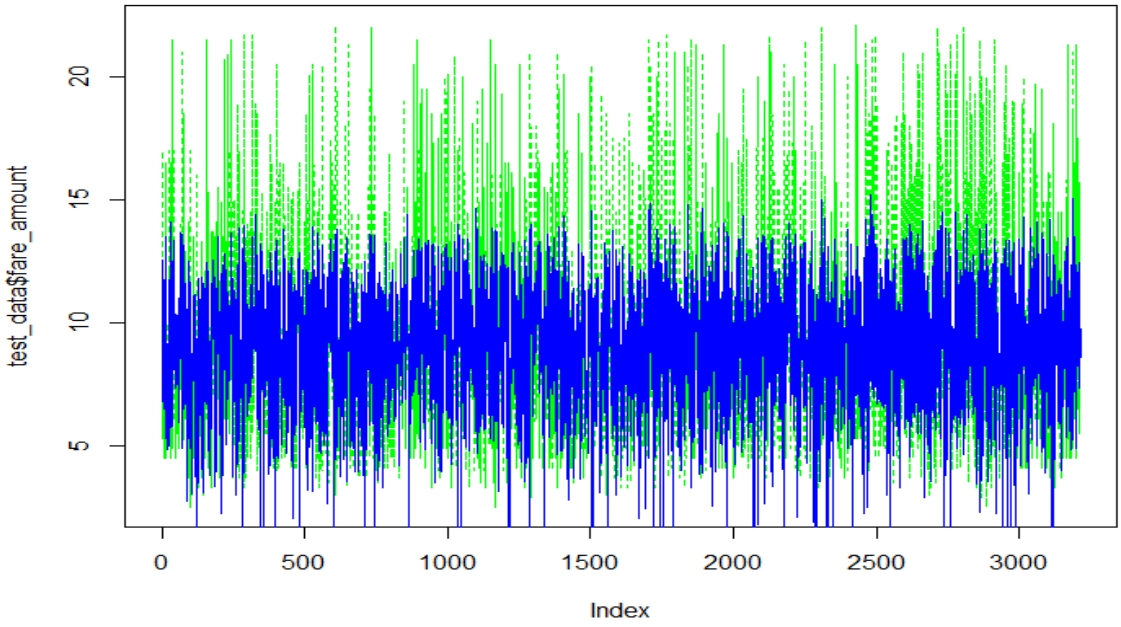
Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Each node in a decision tree represents a feature, each link represents a decision, or a rule and each leaf represent an outcome – *categorical value or continuous value*.

Parameter	R	Python
RMSE	2.470647	2.63070
R-Squared	0.44110	0.5821636

Below is the line plot between actual and the predicted values.

Green line is the Actual values

Blue line is the Predicted values



3.4 Random Forest

Random Forest is one of the most popular and powerful supervised machine learning algorithms. It is a type of ensemble machine learning algorithm called Bootstrap Aggregation or bagging. This ensemble technique consists of many decision trees to improve the accuracy and reduce the weak learners to produce a strong learner from the model.

Let's assume we have 50000 observations in a dataset with 1000 features. We can ensure that one DT will give us the best results or rules or decisions with the minimal errors. In random forest, we do concatenate all the decision trees that we form on a dataset with different observations with different features to obtain the best results.

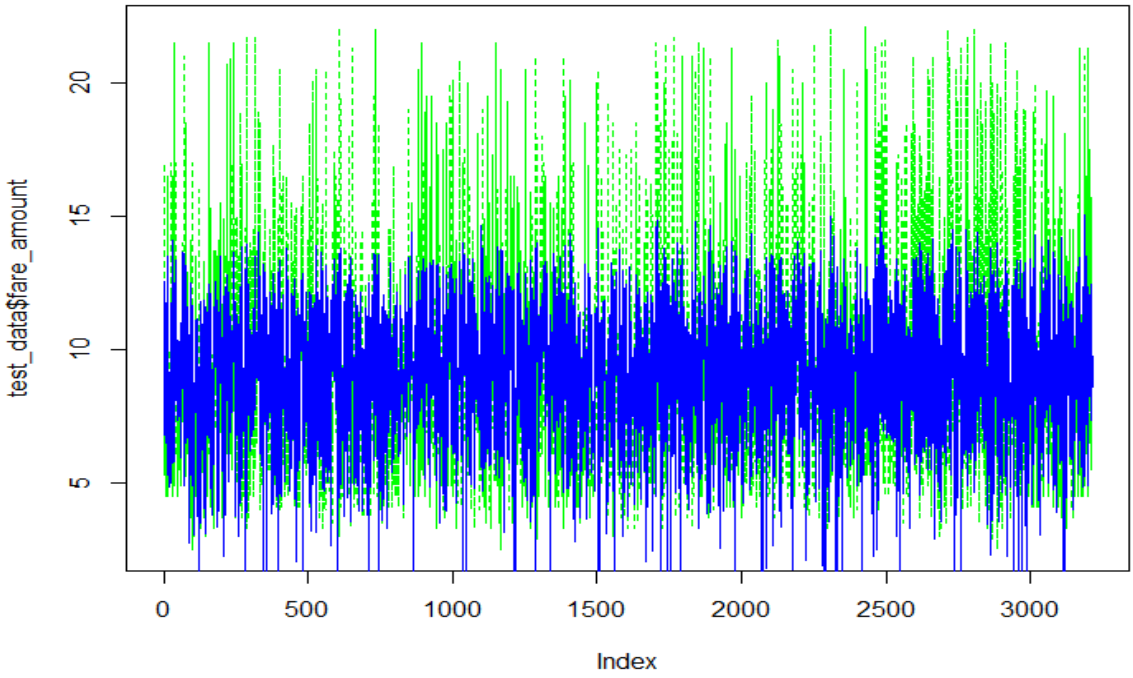
This method is the combination of *Bagging* idea and random selection of features. Bagging is feeding an error of one DT as an input to the next DT to improve the accuracy of the whole model. Likewise, we can improve the accuracy of the model.

I have used 200 trees in R and Python respectively to predict our target variable.

Parameter	R	Python
RMSE	2.470647	2.34786
R-Squared	0.441103	0.677965231

Below is the line plot between actual and the predicted values.

Green line is the Actual values and Blue line is the Predicted values.


test_data$fare_amount
5
10
15
20
0
500
1000
1500
2000
2500
3000
Index

Chapter IV

Conclusion

4.1 Evaluation of the Model

Model evaluation metrics are used to assess goodness of fit between model and data, to compare different models, in the context of model selection and to predict how predictions (associated with a specific model and data set) are expected to be accurate. I have considered the RMSE (Root Mean Square Error). The lower RMSE shows the best fit.

RMSE: Root Mean Square Error (RMSE) is the standard deviation of the residuals which are the prediction errors. *Residuals* are a measure of distance of the data points from the regression line. It is also a measure of how spread out these residuals are. RMSE is calculated by using the below formula,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

Predicted – The value our model predicted

Actual – Actual value

R-Squared and Adj. R-Squared: These 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:

$$\hat{R}^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \bar{Y})^2}$$

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 R^2, adjusted R^2 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:

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(n-1)}{n-k-1} \right]$$

where n is the total number of observations and k is the number of predictors. Adjusted R^2 will always be less than or equal to R^2

An adjusted R^2 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, R^2 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, which are **useful data** points. So, if we use RMSE, it will be better 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* measures the proportion of the variation in our dependent variable (Y) explained by our independent variables (X) for a linear regression model. *Adjusted R-squared* adjusts the statistic based on the number of independent variables in the model. Unlike other metrics, it describes how well the independent variables can predict 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 or negative values are worse. They can be used as goodness of fit metrics of models. For example, if R-Squared value is, say 0.8218, it means that our independent variables can explain 82% variance in the target variable.

So, 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.**

4.2 Selection of the Model

From all the RMSE values we obtained from different models, ***Random forest and Linear Regression*** gave us the least RMSE values. But Linear Regression could explain the variance in our target variable with a better R-Squared value. Hence, we can conclude that ***Linear Regression*** is the best model for this dataset.

4.3 Output

Let us predict the cab fare based on the test data given in **test.csv** by using ***Linear regression*** and see a few observations in the given test data.

Filter

	passenger_count	Year	Month	Date	Hour	distance_travelled	fare_amount
1	0	7	1	27	14	0.0232359618	5.374174
2	0	7	1	27	14	0.0242565185	5.390765
3	0	3	10	8	12	0.0061870742	3.541167
4	0	4	12	1	22	0.0196128152	4.069469
5	0	4	12	1	22	0.0538803208	4.626549
6	0	4	12	1	22	0.0322290481	4.274569
7	0	3	10	6	13	0.0092954374	4.105050
8	0	3	10	6	13	0.2154017801	7.455681
9	0	3	10	6	13	0.0387444288	4.583797
10	0	6	2	18	16	0.0109978037	4.789916

Instructions to run the code in DOS Prompt:

The following steps are to be followed to execute R code or R-Script.exe in DOS prompt.

I. Find the path to R.exe or Rscript.exe on the computer. When we try to run R.exe from the command line, we will get an R terminal (*"C:\Program Files\R\R-3.5.0\bin"*)
II. Find the path to R file that we want to run in cmd line.
III. Open cmd line and execute below statement
"C:\Program Files\R\R-3.5.0\bin>RScript.exe C:\R\CommandLine\Cab_Pred.R"

The following steps are to be followed to execute python code in DOS prompt.

I. Open cmd prompt
II. Get the path for *.py* file *("C:\Users\abhis\Desktop> ")*
III. Enter path> python filename.py

References:

For deployment, https://www.kdnuggets.com/2019/10/easily-deploy-machine-learning-models-using-flask.html

/**********THE END**********/