Project Report

Cab Fare Prediction using Machine Learning Algorithms in R

Instructions to run the code file: -

- Extract the 'Cab_Fare_Prediction.zip' file.
- There will be two folders "Python" and "R".
- Make sure R studio is installed in your PC.
- From the "R" folder open three files namely: "Clean_train+EDA.R", "Clean_test.R" and "Fare_predict.R".
- R Studio will open the above files.

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.

Solution: -

This project was done in three parts, and has 3 R Scripts (.R) files attached with it.
The first part deals with cleaning of the train data and exploratory data analysis (EDA).
The second part deals with cleaning of the testing data and third part deals with prediction of fares in the test data. Each part will be explained in detail in this report.

1.1 Cleaning of training data and Exploratory Data Analysis (Please refer to the file named "Clean_train_data+EDA.ipynb")

The training data contains 16062 observations and 7 variables which are:

- **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.

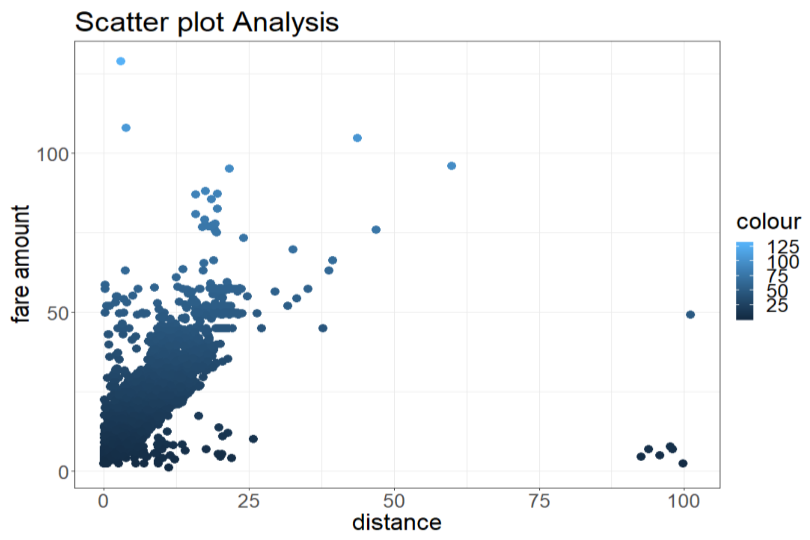
The entire procedure of cleaning the train data will be further explained step by step:

- First, "train_cab.csv" as opened in MS Excel to observe the data. The train data had many missing values, some values were 0 for number of passengers and some values were more than 6. Also, it was observed that some fare amount had negative values. Here, the missing values were replaced with 0 using special paste function in excel. The file was saved and rest of the data cleaning was done in R Studio.

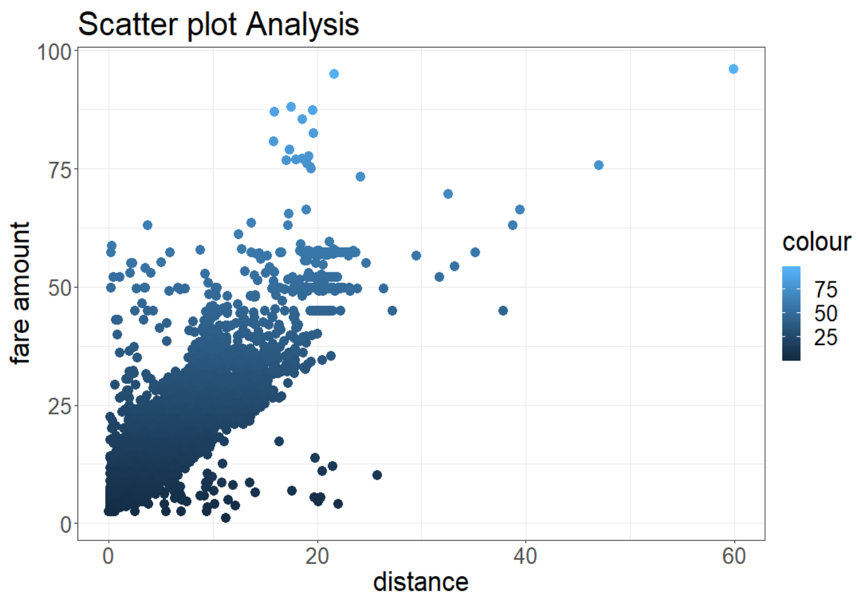
- In R Studio, we first clear all the objects from the environment using: *rm(list=ls())*
- We then set working directory of our project.
- Then, we load all the libraries that we would require for this project. Some of the important ones are:
    - dplyr : package for data manipulation.
    - ggplot2 : data visualization package.
    - tidyr: used for data cleaning.
    - lubridate: used for extracting year, month, date, day and time from timeseries data.
    - geosphere: spherical trigonometry package used for calculating Haversine distance.
    - DMwR: data mining tools and KNN imputation.

- After loading the libraries, we loaded the “train_cab.csv’ which is our train dataset. In this command note that *na.strings* command is used to convert the matching strings in the dataset with NA.
- Then we checked the summary of the dataset to get an overall idea of it. Here, it was observed that maximum fare value was 54343 which is not possible and clearly an outlier. Then pickup longitude had max value 40.766 as the longitude must be in the range of -70. The same applies for dropoff longitude. For pickup latitude max value was 401.08, which is again an outlier. Dropoff latitude seemed fine. Passenger count had max value of 5345 which is not possible.
- The outlier values for passenger count was removed followed by fare amount and GPS co-ordinate values. After these extreme datapoints were removed, it was time to impute the missing data.
- To impute missing data, KNN imputation was used. Imputations with different values for k were compared and it was found that for k=10, the result was close to the original as well as it performed better than other values of k.
- Then, features like year, month, date, day and hour were extracted from timeseries variable pickup_datetime. After extraction, pickup_datetime was removed from the dataframe.
- Next, the Haversine distance was calculated using pickup and dropoff longitude and latitude and was appended to the dataframe. Haversine distance is basically a vector distance that is calculated from the pickup point to the dropoff point on a spherical surface like our earth.
- We now deal with some more outliers. We see that some values for fare amount are $2.5 and distance travelled for those observations was 0. This means that the customer booked a cab and for some reason cancelled it without completing the ride, incurring him the base charge of $2.5. These observations are of no use to us and are thus, removed.

- We also observe that there are some values where fare amount was more than $2.5 but the distance was 0. This must indicate a round trip as pickup and dropoff locations were same. We calculate distance for such observations as follows:
  distance = (fare_amount - 2.5)/1.56
  Here, $2.5 is base charge and $1.56 is added per km.
- Also, there are some implausible combinations of fare amount and distance i.e. for very less distance (<0.1km) , fare amount was more than $2.5. Now, we know that for first 1km, we need to pay only $2.5. But, here it was observed that for distance <0.1, values were $5 to $10 range. Such values were removed.

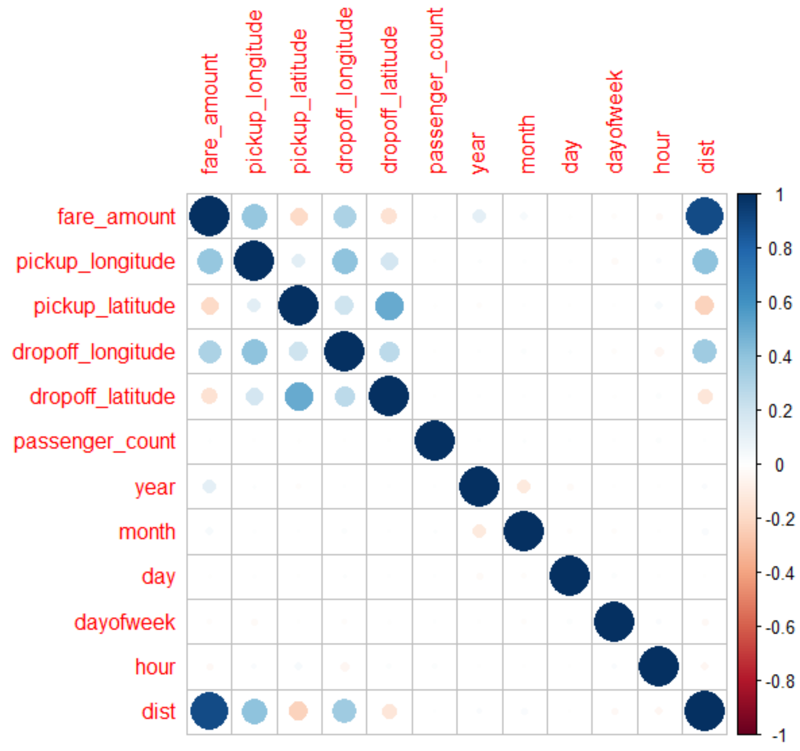


- As we can see from the scatter plot, there are still some values where distance is more and fare is very less and vice versa. Such outlier values were removed.



- As seen from the graph, there is a linear relationship between distance and fare amount. The cleaned dataset was saved as .csv file.

1.2 Exploratory Data Analysis



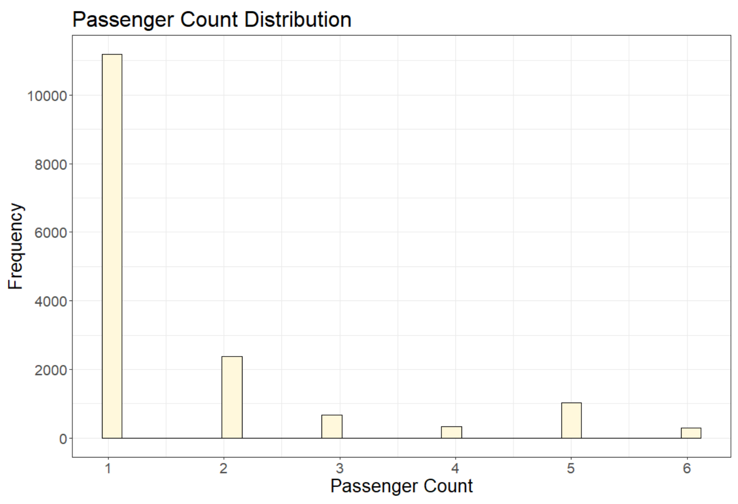
After cleaning the data, some exploratory data analysis (EDA) was done to figure out which features or variables were important and which ones created multicollinearity and were to be removed.

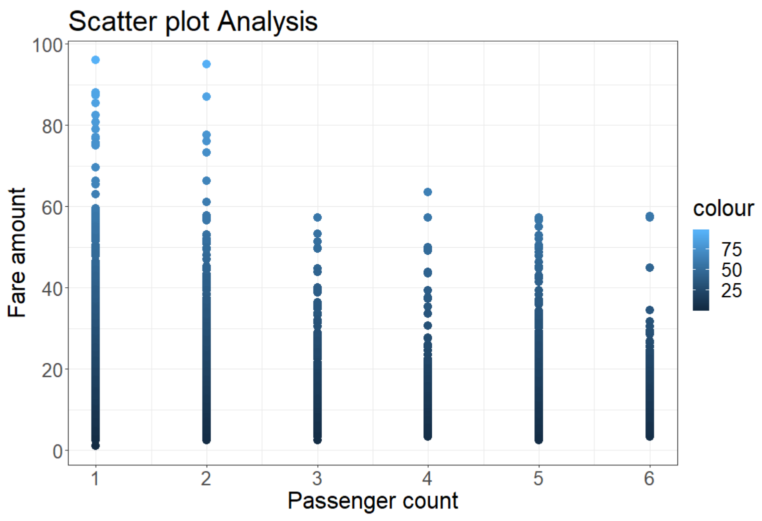
- The first step in EDA is to make a correlation plot to see the correlation among different variables.
- Here from the above correlation plot, it was observed that distance and fare amount variables have a strong positive correlation, which is a good thing since independent variables need to be somehow related with target variable. Only thing is that, independent variables should not be strongly correlated with each other as that would cause redundancy. Here, no independent variable was strongly correlated and thus, no variable was removed.

- The aim of further EDA was to address following questions:
    i. How no. of passengers affects the fare?
    ii. How year affect the fare?
    iii. How month affect the fare?
    iv. How day of the month affects the fare?
    v. How day of the week affects the fare?
    vi. How particular hour of day affects the fare?

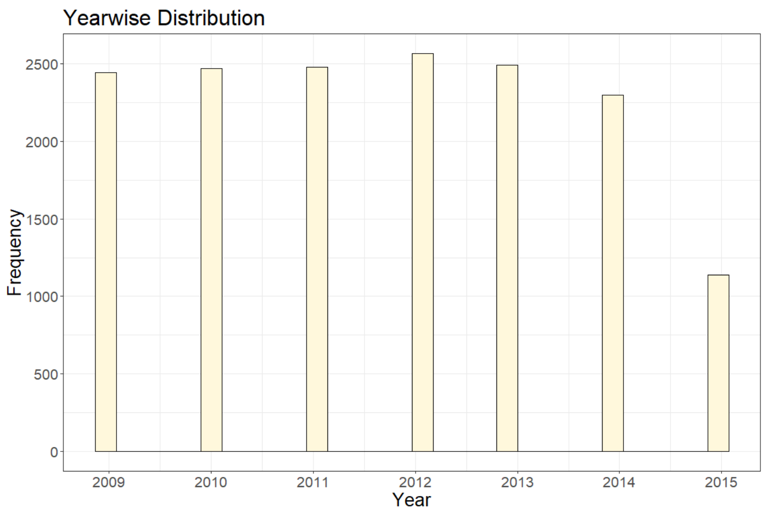
Following are their answers that were obtained through visualizations:

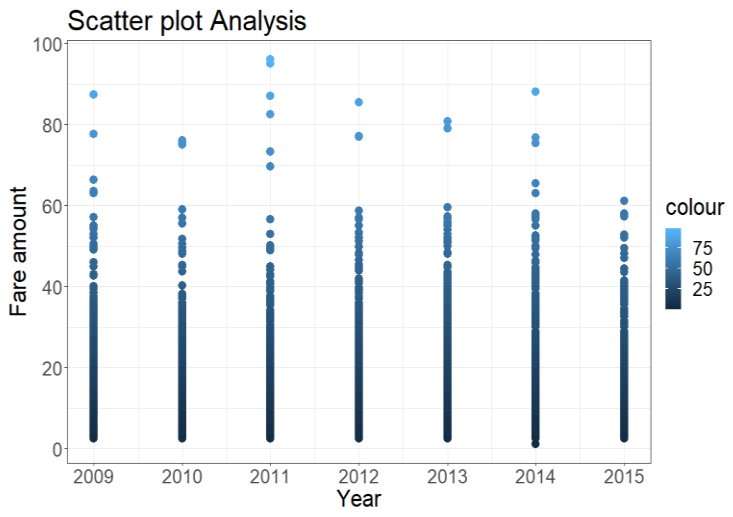
i. Single or two passenger rides were ones with maximum fare amount. It could be that such rides covered longer distances.



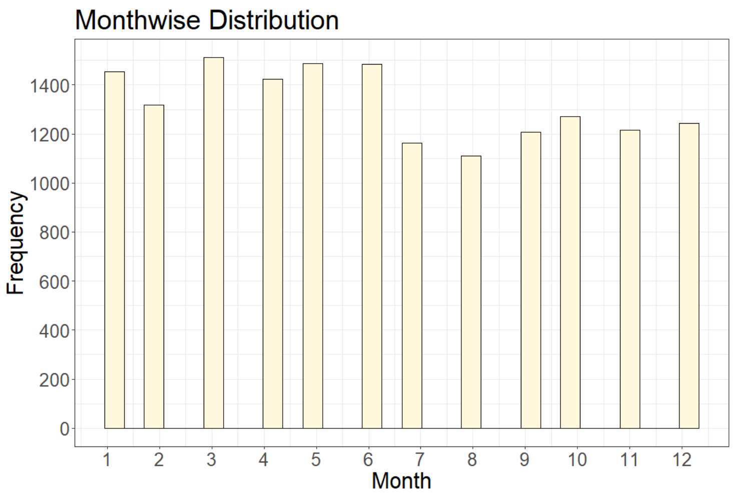


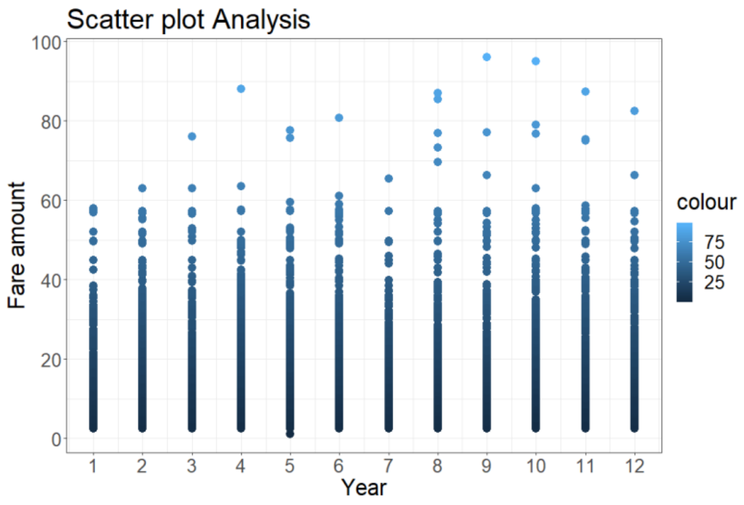
ii. The fares seem consistent over the years. Although there are some datapoints indicating high fares in 2011, they are very few and maybe they are long distance rides, hence the high fares



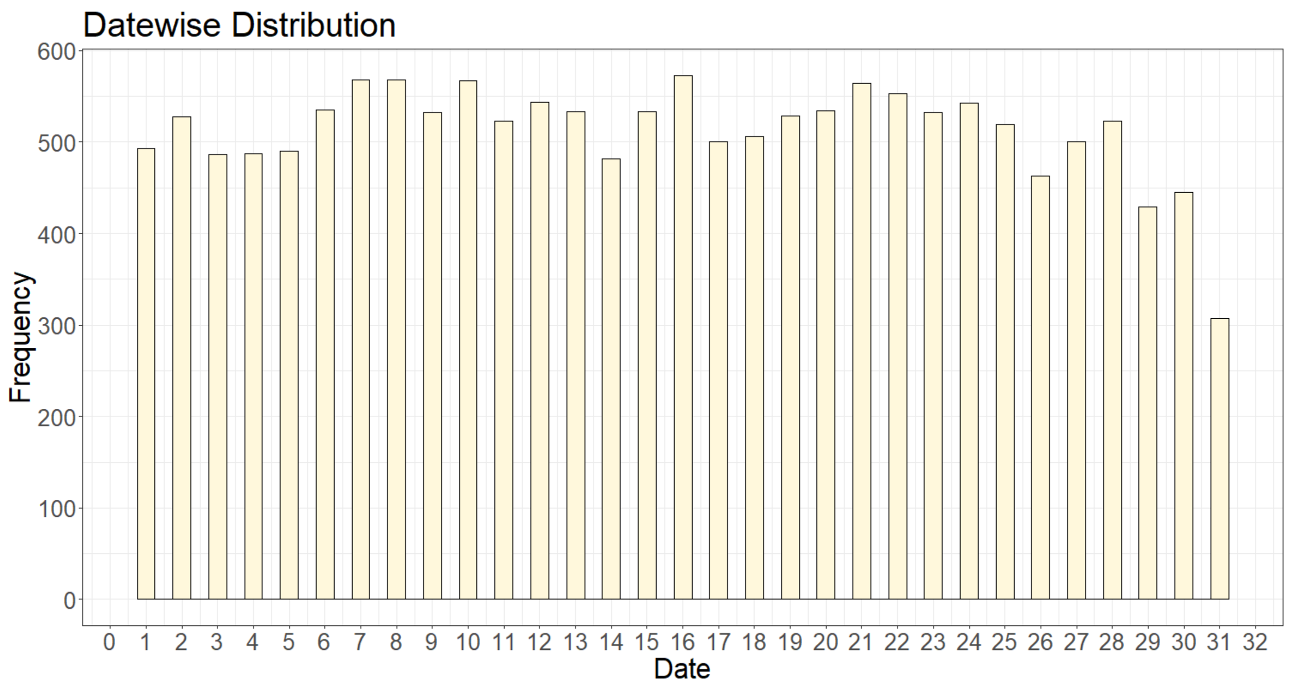


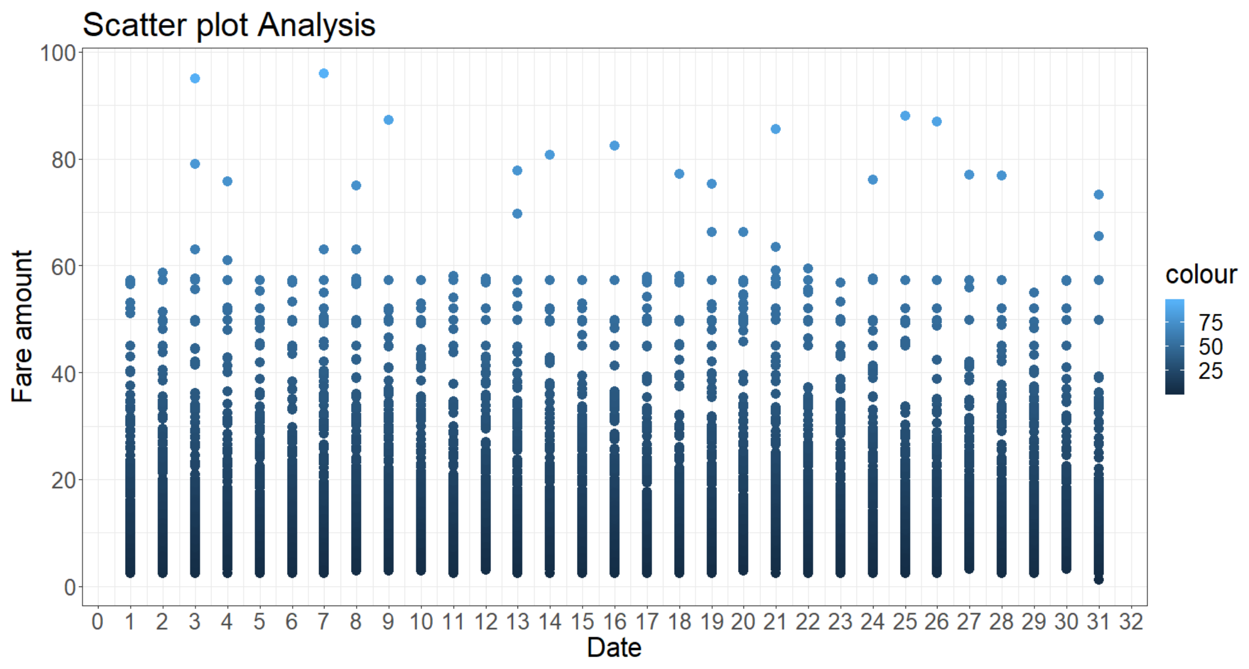
iii. Fare amounts from August to December seems to be higher than other months.



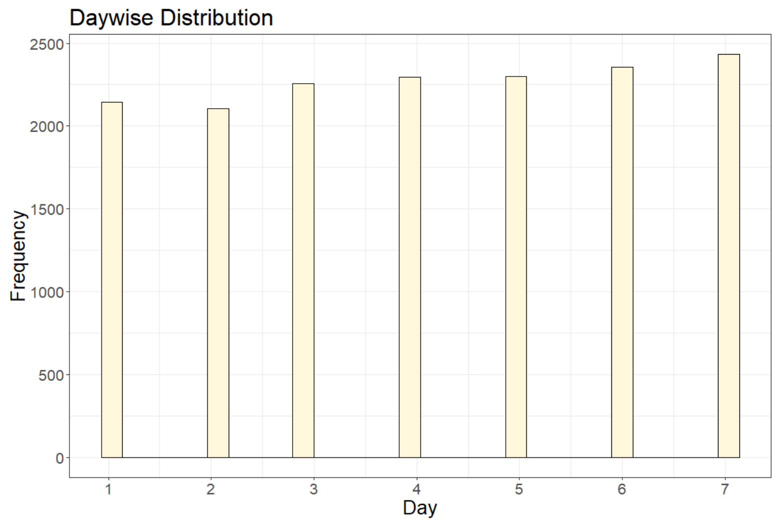


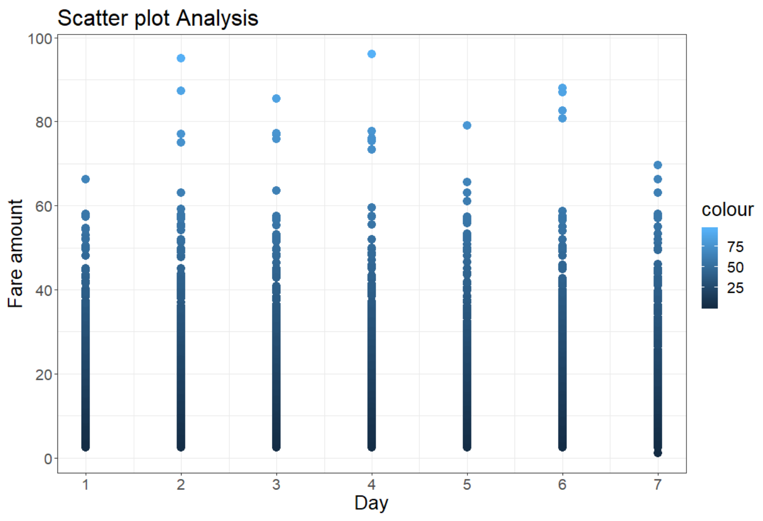
iv. The prices seem more or less consistent throughout the month.



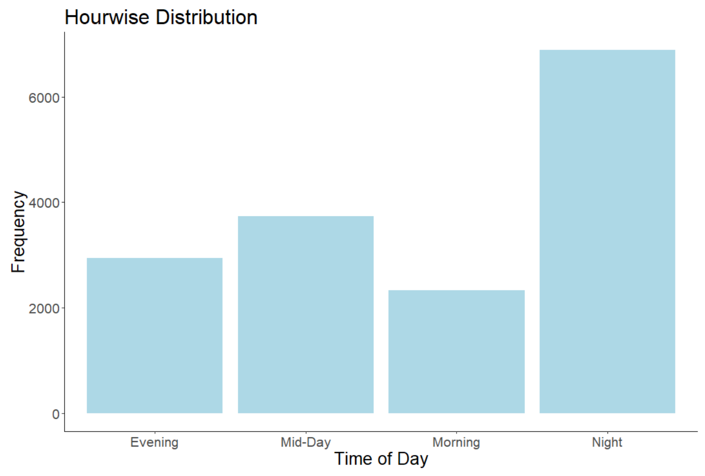


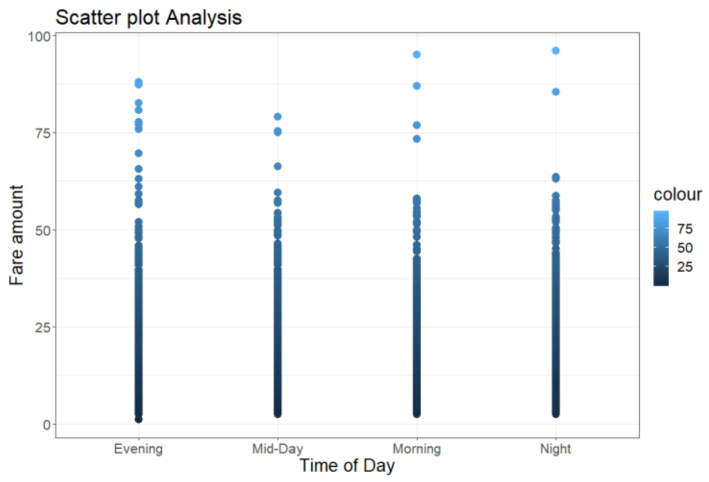
v. The highest frequency is on Friday and Saturday (weekends). While the prices aren't affected that much although there are some datapoints with high fares which maybe due to people travelling long distances during weekdays.





vi. Frequency during midnight and early morning form i.e. 3 am to 9 am (labelled "Morning") is observed as least. The frequency is most during 6 pm to 3 am (labelled "Night). However, the prices seem to be comparable and just with few high fare amounts exist during morning and night. One fact to notice is that even though frequency during morning is least, yet the prices are high. It could be that prices during this time of day are high.





2. Cleaning Test Data
(Please refer to the file named "Clean_test.R")

- Explore the text file in excel. It was observed that there were no missing values in the dataset. The dataset was further explored in the python notebook.
- Import important libraries just as in the previous section.
- Check the summary of numeric variables. It was observed that all the values were sensible and there were no outliers.
- Then we checked for missing values again and there were no missing values in the dataset.
- Next, features were extracted from pickup datetime variable and features like Year, Month, Date, Day and Hour were appended to the test dataset.
- Then, we drop pickup_datetime once extraction of necessary information is done.
- Then, Haversine Distance was calculated for the given co-ordinates in the test dataset, just like we did in the train dataset.
- Next, we looked for 0 values in the H_Distance variable and found that 86 observations had distance 0. This may indicate a round trip, but we cannot impute it appropriately as we don't have fare amount to calculate the best estimate of distance. So, we drop these observations.

3. Fare Prediction
(Please refer to the file named "Fare_Prediction.ipynb")

For this section, we first created a baseline model using multiple linear regression model and then went on to use Lasso Regression, Decision Tree, Random Forest, XGBoost and LightGBM. Detailed explanation for the process is given below.

- Import the important libraries.
- Load the cleaned train and test data.
- Now, we need to split our training data into train and validation data. First, we set the seed value to 1234 and then use random sampling to split the data. The train data was given 80% of the original training data and validation data was given 20% of the original training data.
- Independent variables and dependent variable were separated into different objects for both train and validation data.
- Next, for calculating error metrics for our models, a function was written wherein all the error metrics like mean absolute error (MAE), root mean square error (RMSE), Akaike information criteria (AIC) etc.

  Note: Even though many error metrics were calculated, special attention was given to RMSE values. The reason for choosing RMSE for determining error in this project is that RMSE penalizes highly varying predicted values from the original values. In other words, if the residue is higher, higher would be the RMSE. On the other hand, MAE will give us mean of all the residues i.e. it will give us exact mean of difference between original and predicted values without penalizing highly varying predicted values. Thus, if we use RMSE, we will get a good idea as how well our model is predicting the fares.

  Implementation of Models: -

  i. **Multiple Linear Regression: -**
  Created a baseline model using MLR. The baseline RMSE we got is 4.029 for train data and 4.132 for validation data. Our aim is to build a model that has better RMSE score than this. Also, the variance between these two RMSEs matter and should be minimum, because if these are far apart, it would indicate overfitting of the model which we must avoid.

  ii. **Lasso Regression: -**
  A Lasso Regression model was implemented and was iterated for different values of alpha, but it was found that using lasso regression is not feasible as alpha=0.015 gives us least RMSE of 4.133 which is almost as good as multiple linear regression model's RMSE according to above analysis. But if we use such a small value of alpha for lasso, it will reduce lasso to linear regression model (as alpha=0 for lasso gives linear regression). Thus, we won't use Lasso for our further analysis.

iii. **Decision Tree Model: -**

We also tried a non-linear model like a Decision Tree and Random Forest. For Decision Tree, we observed that the best RMSE value it could provide is 4.31 for train data and 4.37 for validation data. Both these values are more than our baseline MLR model RMSE. Thus, we cannot use this model.

iv. **Random Forest: -**

We tried decision trees and now we will try Random Forest. As it is based on ensemble method and may avoid overfitting problem as seen in decision tree. Also, we will see whether it gives better result than the decision tree.

Although, the RMSE for train and validation data (1.651 and 3.714) are the least of all above models. But it is observed that RMSEs of train and validation data are way off. The variance is 2.06. We will try other models and see if we can get a better fit.

v. **Light Gradient Boosting: -**

We tried LightGBM model to see whether it performs better. LGBM has trees of fixed length and are shorter than the trees in decision tree and random forest. It is also based on ensemble method like random forest, but the difference is that all the trees in LGBM do not have equal say on the output. The say (or vote) of each tree depends upon the error of previous tree and how that current tree reduces the error.

In the model, most of the default parameters were set and we implemented our model.
The results were good. Light gradient boosting model seems to perform better than all previous models with least RMSEs (1.174 & 3.762) in both training and validation and variance between train and validation is 2.53.

But before we fix this model for our test data, lets try XGBoost algorithm to see if it performs better than this.

vi. **Xtreme Gradient Boosting: -**

This model is similar to LGBM. Infact LGBM improves upon XGBoost. On implementation, we get RMSEs (1.3065 & 3.697) on training and validation data. As it is seen, it performed slightly better than Random forest and LGBoost.

Out of all the models we have implemented, Random Forest, LGBoost and XGBoost have given us best results for validation data. Since, the validation RMSEs are so close, we use variance and AIC score to evaluate which model is best for our purpose.

Variance values for RF, LGB and XGB are 2.06, 2.53 and 2.39 respectively and AIC values are 8356, 8437 and 8327 respectively. We have to choose model with minimum variance and AIC.

Thus, we choose XGB to predict our final test data as it seems to have good compromise between variance, AIC and validation RMSE value i.e. it seems to be more balanced.

The fare amount in the test dataset was predicted and predicted values were appended to the test dataset. The complete file with the predicted fares was exported as 'Final Predictions.csv' and is included in the 'R' folder in 'Cab_Fare_Prediction.zip'.

One of the ways in which we can improve the accuracy of our model is by feeding it more data. The more the data we give it, more the patterns our model will get to train on.

4. Conclusion

- The objective of this project, which was to predict the fare_amount of a cab ride in the New York City by implementing various machine learning models and selecting the best among them which presented least RMSE.
- For this, we first had to clean the train and test data and carry out EDA to understand our data better.
- Then, from our comparison and analysis of different machine learning models, it was observed that Xtreme Gradient Boost had best combination of all parameters (low RMSE, Variance and AIC) as compared to other models. Thus, XGBoost was selected for predicting fare values for our test data.