Cab Fare Prediction

Author: Abhijeet Nag
08/9/2019

Contents

1.1 Problem Statement

1.2 Data

Chapter 2 Methodology

- Pre-Processing
- Modelling
- Model Selection

Chapter 3 Pre-Processing

3.1 Data exploration and Cleaning (Missing Values and Outliers)

3.2 Creating some new variables from the given variables

3.3 Selection of variables

3.4 Some more data exploration

- Dependent and Independent Variables
- Uniqueness of Variables
- Dividing the variables categories

3.5 Feature Scaling

3.6 Feature Selection

Chapter 4 Modelling

4.1 Linear Regression

4.2 Decision Tree

4.3 Random Forest

Chapter 5 Conclusion

5.1 Model Evaluation

5.2 Model Selection

5.3 Some Visualization facts

Chapter 1

Introduction

Now a day's cab rental services are expanding with the multiplier rate. The ease of using the services and flexibility gives their customer a great experience with competitive prices.

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

Understanding of data is the very first and important step in the process of finding solution of any business problem. Here in our case we were provided with dataset with following features, we need to go through each and every variable of it to understand and for better functioning.
Size of Dataset Provided: - 16067 rows, 7 Columns (including dependent variable)
Missing Values: Yes
Outliers Presented: Yes
Below mentioned is a list of all the variable names with their meanings:

Number of attributes:

- pickup_datetime - timestamp value indicating when the cab ride started.
- pickup_longitude - float for longitude coordinate of where the cab ride started.
- pickup_latitude - float for latitude coordinate of where the cab ride started.
- dropoff_longitude - float for longitude coordinate of where the cab ride ended.
- dropoff_latitude - float for latitude coordinate of where the cab ride ended.
- passenger_count - an integer indicating the number of passengers in the cab ride.

Missing Values: Yes

Chapter 2

Methodology

➢ Pre-Processing

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.
EDA involves some of the steps below mentioned.

- Data exploration and Cleaning
- Missing value treatment
- Outlier Analysis
- Feature Selection
- Features Scaling
    - Skewness and Log transformation
- Visualization

➢ Modelling

Once all the Pre-Processing steps has been done on our data set, we will now further move to our next step which is modelling. Modelling plays an important role to find out the good inferences from the data. Choice of models depends upon the problem statement and data set. As per our problem statement and dataset, we will try some models on our preprocessed data and post comparing the output results we will select the best suitable model for our problem. As per our data set following models need to be tested:

- Linear regression
- Decision Tree
- Random forest

❖ We have also used hyper parameter tunings to check the parameters on which our model runs best. Following is the techniques of hyper parameter tuning we have used:
    - Grid Search CV

➢ Model Selection

The final step of our methodology will be the selection of the model based on the different output and results shown by different models. We have multiple parameters which we will study further in our report to test whether the model is suitable for our problem statement or not.

Chapter 3

Pre-Processing

3.1 Data exploration and Cleaning (Missing Values and Outliers)

The very first step which comes with any data science project is data exploration and cleaning which includes following points as per this project:

a. Separate the combined variables.
b. As we know we have some negative values in fare amount so we have to remove those values.
c. Passenger count would be max 6 if it is a SUV vehicle not more than that. We have to remove the rows having passenger_count more than 6 and less than 1.
d. There are some outlier figures in the fare (like top 3 values) so we need to remove those.
e. Latitudes range from -90 to 90. Longitudes range from -180 to 180. We need to remove the rows if any latitude and longitude lies beyond the mentioned range.

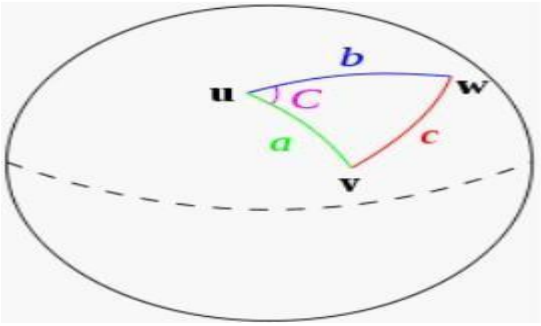
3.2 Creating some new variables out of the given variable.

Here in our data set our variable name pickup_datetime contains date and time for pickup which is of type timestamp. So we tried to extract some important variables from pickup_datetime:

- Pickup_month
- Pickup_date
- Pickup_weekday
- Pickup_Hour
- Pickup_Minute

Also, we tried to find out the distance using the haversine formula which says:

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.



So our new extracted variables are:

- fare_amount
- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- passenger_count
- pickup_Month
- pickup_Date
- pickup_weekday
- pickup_Hour
- pickup_Minute
- Distance

3.3 Selection of variables

Now as we know that all above variables are of now use so we will drop the redundant variables:

- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude

Now only following variables we will use for further steps:

```
In [145]: cab_train.head()
```

Out[145]:

	fare_amount	passenger_count	pickup_date	pickup_hour	pickup_month	pickup_weekday	pickup_minute	distance
0	4.5	1.0	15.0	17.0	6.0	0.0	26.0	1.030764
1	16.9	1.0	5.0	16.0	1.0	1.0	52.0	8.450134
2	5.7	2.0	18.0	0.0	8.0	3.0	35.0	1.389525
3	7.7	1.0	21.0	4.0	4.0	5.0	30.0	2.799270
4	5.3	1.0	9.0	7.0	3.0	1.0	51.0	1.999157

3.4 Some more data exploration

In this report we are trying to predict the fare prices of a cab rental company. So here we have a data set of 16067 observations with 8 variables including one dependent variable.

3.4.1 Below are the names of Independent variables:

passenger_count, pickup_Minute,pickup_ Month, pickup_Date, pickup_Weekday, pickup_Hour, distance

Our Dependent variable is: **fare_amount**

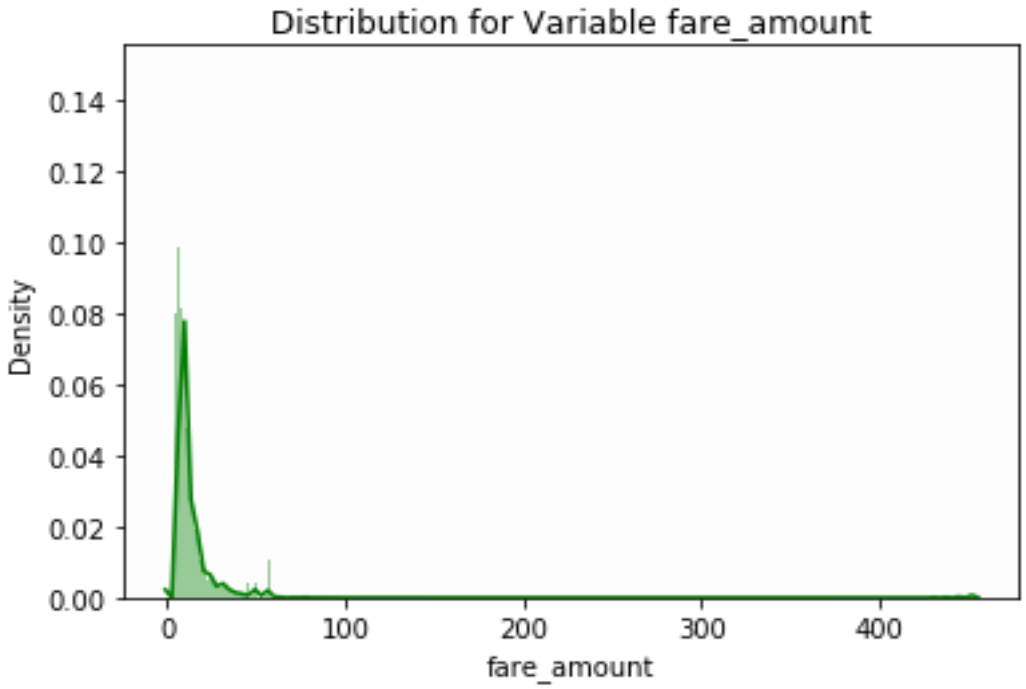
3.4.2 Dividing the variables into two categories basis their data types:

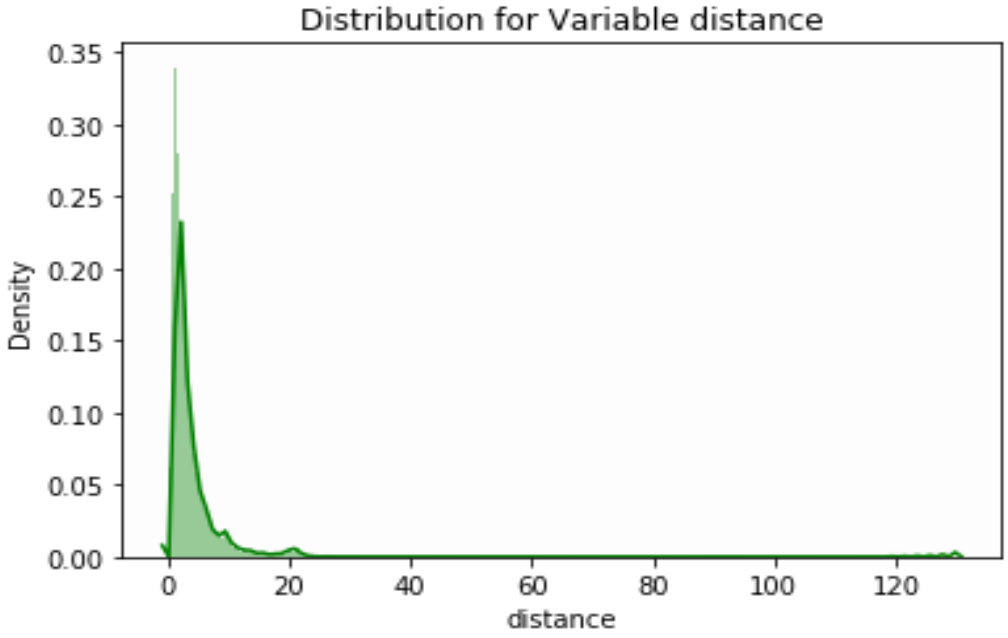
Continuous variables - 'fare_amount', 'distance'. 'pickup_Month', 'pickup_Date', 'pickup_weekday', 'pickup_hour'
Categorical Variables - 'passenger_count'

3.5 Feature Scaling

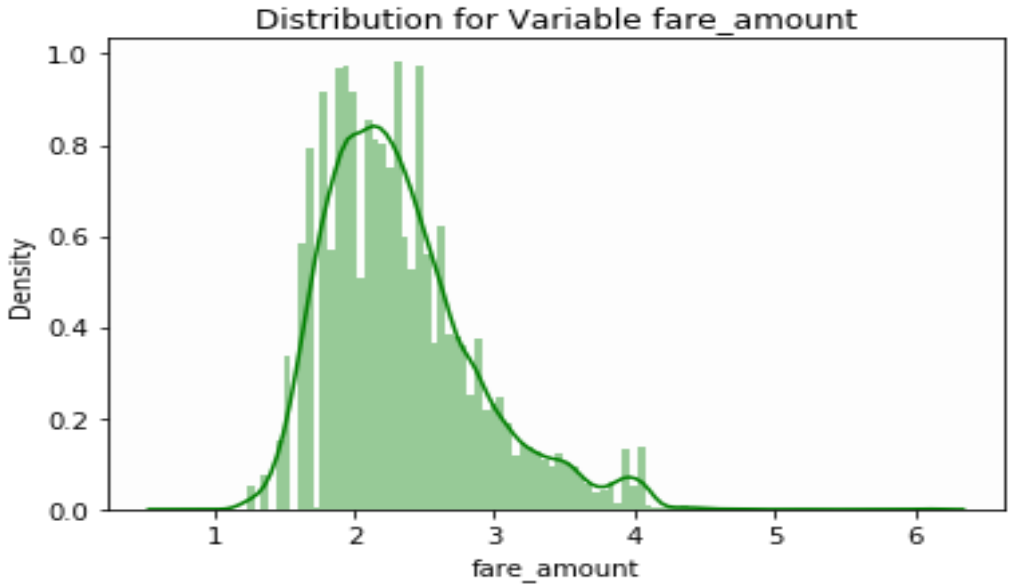
Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we tried to show the skewness of our variables and we find that our target variable absenteeism in hours having is one sided skewed so by using **log transform** technique we tried to reduce the skewness of the same.

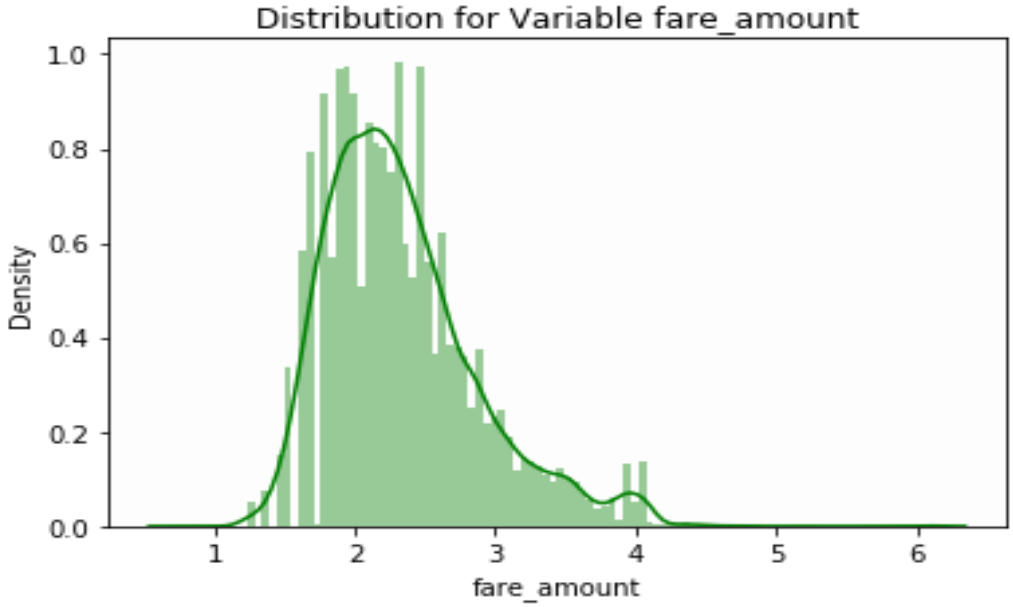
Below mentioned graphs shows the probability distribution plot to check distribution before log transformation:





Below mentioned graphs shows the probability distribution plot to check distribution after log transformation:



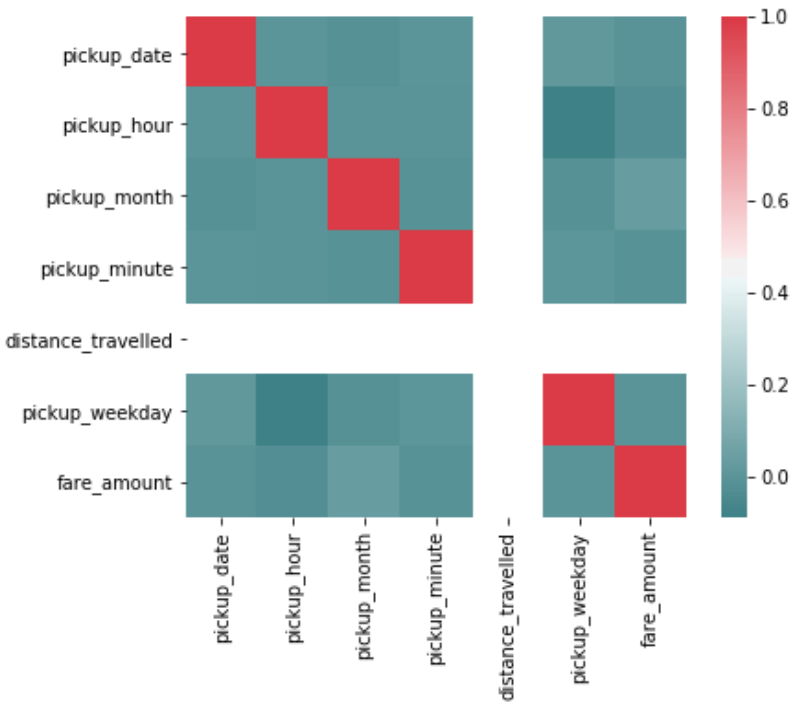


As our continuous variables appears to be normally distributed so we don't need to use feature scaling techniques like normalization and standardization for the same.

3.6 Feature Selection

In Feature selection we plot the correlational analysis of the predictor variable and target variable too. We will look for high correlation between predictor and target variable and low correlation among the predictor variable. If there is high correlation between two predictor variable then we will delete one of them. This is called Dimension Reduction.
As our target variable is continuous so we can only go for correlation check. As chi-square test is only for categorical variable.

Below Figure show a correlation plot for all numeric variable present in dataset



Chapter 4

Modelling

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.

Models built are:-

- c50 (Decision tree for regression target variable)
- Random Forest (with 200 trees)
- Linear regression

Before running any model, we will split our data into two parts which is train and test data. Here in our case we have taken 80% of the data as our train data. Below is the snipped image of the split of train test.

```
In [157]: # Splitting the data into training set and test set
          from sklearn.model_selection import train_test_split
          train, test = train_test_split(cab_train, test_size=0.2)
```

4.1 Linear Regression

It is one of the most widely known modeling technique. Linear regression is usually among the first few topics which people pick while learning predictive modeling. In this technique, the dependent variable is continuous, independent variable(s) can be continuous or discrete, and nature of regression line is linear.

Further the data is again divided into train and test with 80 % train data and 20 % test data using random sampling.

Creating Model:-

In R:-

```
################### Multiple Linear Regression ##############################
regressor= lm(formula = fare_amount ~., data = training_set)
#Predicting the test set results
y_pred= predict(regressor,test_set[, -1])
```

In Python:-

```
##################### LINEAR REGRESSION #################
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(train.iloc[:, 1:8], train.iloc[:, 0])
predictions_LR= regressor.predict(test.iloc[:, 1:8])
```

4.2 Decision Tree

This model is also known a Decision tree for regression target variable.
For this model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of data set and test contains 20% data and contains 14 variable where 14th variable is the target variable.

Creating Model:-

In R-

```
######################### Decision TREE ######################

regressor_DT= rpart(formula = fare_amount ~., data = training_set, method = "anova")
#Predicting the test set result
y_pred_DT= predict(regressor_DT, test_set[, -1])
```

In Python-

```
########################### Decision Tree ############
from sklearn.tree import DecisionTreeRegressor
regressor_DT = DecisionTreeRegressor(max_depth=10).fit(train.iloc[:,1:8], train.iloc[:,0])
predictions_DT = regressor_DT.predict(test.iloc[:,1:8])
```

4.3 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other task, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Creating Model:-

In R-

```
################ Random FOrest ########################

regressor_RF= randomForest(formula= fare_amount ~., data= training_set ,importance= TRUE, ntree= 300, keep.forest = TRUE)

#Predicting the test set result
y_pred_RF= predict(regressor_RF, test_set[,-1])
```

In Python-

```
#################### Random Forest ######################
from sklearn.ensemble import RandomForestRegressor
regressor_RF = RandomForestRegressor(n_estimators = 200, max_depth= 7).fit(train.iloc[:,1:8], train.iloc[:,0])
RF_Predictions = regressor_RF.predict(test.iloc[:,1:8])
```

Chapter 5

Conclusion

5.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Bike Renting, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

Mean Absolute Percentage Error (MAPE)

MAPE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous sections

```
# MAPE Function
def MAPE(y, yhat):
    mape = np.mean(np.abs((y - yhat) / y))*100
    print("MAPE:", mape)
```

In above function y is the actual value and yhat is the predicted value. It will provide the error percentage of model.

Root Mean Squared Error(RMSE)

RMSE is the most popular evaluation metric used in regression problems. It follows an assumption that error are unbiased and follow a normal distribution.

```
#RMSE Function
def RMSE(y_test,yhat):
    mse = np.mean((y_test-yhat)**2)
    print("Mean Square : ",mse)
    rmse=np.sqrt(mse)
    print("Root Mean Square : ",rmse)
```

MAPE ,RMSE and R-squared value in Python are as follow

```
: #Mape for Linear Regression
MAPE(test.iloc[:,0],predictions_LR)
RMSE(test.iloc[:,0],predictions_LR)

MAPE: 8.06334883804959
Mean Square :  0.08547478254297824
Root Mean Square :  0.2923607062226014

: #calculate R^2 for Linear Regression Model
r2_score(test.iloc[:,0], predictions_LR)

: 0.7196094035065019

: #Mape for Decision Tree
MAPE(test.iloc[:,0],predictions_DT)
RMSE(test.iloc[:,0],predictions_DT)

MAPE: 8.480695528000217
Mean Square :  0.08877278102570765
Root Mean Square :  0.29794761456623153

: #calculate R^2 for Decision Tree Model
r2_score(test.iloc[:,0], predictions_DT)

: 0.7087906832442755

: #Mape for Random Forest
MAPE(test.iloc[:,0],RF_Predictions)
RMSE(test.iloc[:,0],RF_Predictions)

MAPE: 7.823785670752574
Mean Square :  0.07214469237686673
Root Mean Square :  0.2685976403039809
```

MAPE & RMSE value in R:

```
> regr.eval(test_set[, 1], y_pred, stats = c('mae', 'rmse', 'mape', 'mse'))
       mae       rmse       mape        mse
0.13330366 0.17037908 0.15778813 0.02902903
> regr.eval(test_set[, 1], y_pred_DT, stats = c('mae', 'rmse', 'mape', 'mse'))
       mae       rmse       mape        mse
0.09196320 0.12439316 0.10823026 0.01547366
> regr.eval(test_set[, 1], y_pred_RF, stats = c('mae', 'rmse', 'mape', 'mse'))
       mae       rmse       mape        mse
0.09072870 0.12210601 0.10868652 0.01490988
```

Model Selection

We can see that in both R and Python Random Forest Model fits the best out of Decision Tree and Linear Regression. MAPE Value is the lowest for Random Forest. So to improve the model and enhance its performance so that it can perform efficiently on new test set, implemented **K Fold Cross Validation**.

In Python-

```
#Applying K Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies= cross_val_score(estimator= regressor_RF, X= train.iloc[:,1:8], y= train.iloc[:,0], cv= 10)
accuracies.mean()
```

After applying K Fold Cross Validation on Random Forest We did hyper tunning using grid search cv.

Grid Search CV: This algorithm set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient.

Check results after using Grid Search CV on Random forest

```
# As we know the given problem is a regression so we are considering two performance metrics:-
#1) MAPE (2) RMSE
# Suppose Predicted cab fare_amount is 100 and the actual is 102, then here we will not see the difference
#in error i.e 2 but we will calculate the percentage error which is 2%. Again, if the predicted fare_amount
# is 10 instead of 12, error difference is again 2 but the percentage error is 20% .So we will be considering
# MAPE as our error metrics
#As we can see clearly see the value of both the Error Metrics i.e MAPE and RMSE both are minimum for Random Forest model
# and value of R-square is also higher for Random Forest. So Choosing Random Forest as our Final Model and
#applying grid search to find the best value of hyper-parameters
from sklearn.model_selection import GridSearchCV
parameters= [{'max_depth': [2,5,7,8,10], 'n_estimators' : [10, 100, 200, 300, 400, 500]}]

grid_search= GridSearchCV(estimator = regressor_RF, param_grid= parameters, cv= 5, n_jobs= -1)
grid_search= grid_search.fit(train.iloc[:,1:8], train.iloc[:, 0])
```

```
#Calculating the best value of the provided hyper-parameter
best_param= grid_search.best_params_
```

```
best_param
```

```
{'max_depth': 7, 'n_estimators': 200}
```

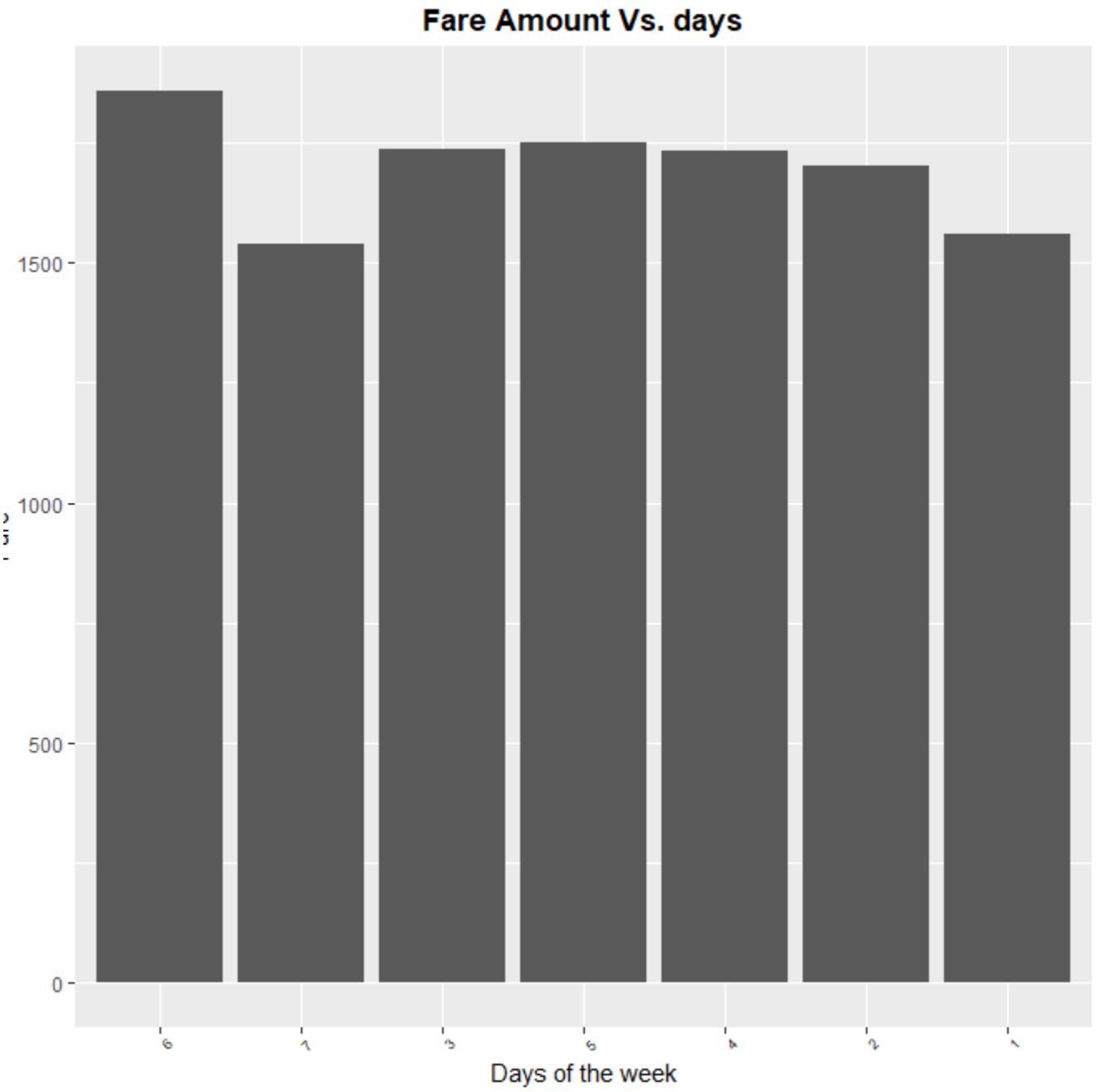
```
print(grid_search.best_score_)
```

```
0.7811608772639813
```

When we check the best_params we find that the best value of hyper parameter ‘max_depth’ & ‘n_estimator’ are 7 and 200 respectively.

Visualization

In this bar plot we are analyzing the fare_amount based on which day of the week it is.



In this bar plot we are analyzing the fare_amount based on month