Project report on cab fare prediction

**2020**

Archit kumar jha

[Type the company name]

7/4/2020



**s.no. list of contents**

**1. introduction**

**2. problem statement**

**3. data provided**

**4. pre-processing**

**5. modeling**

**6. model selection**

**7. haversine formula**

**8. uniqueness in variables**

**9. feature scaling**

* **In python**
* **In r**

**10. final modeling**

* + - **Splitting inpython**
    - **Splitting in r**

**11. linear regression**

* **In python**
* **In r**

**12. decision tree**

* **In python**
* **In r**

**13. random forest**

* **In python**
* **In r**

**14. hyper parameter tuning ( in python only)**

**15. modelling evaluation**

* **In python**
* **In r**

**16. visualisations**

**introduction**

In this world of rapid growing economy, increasing population and a huge workforce in every country Personal and professional commutes are a necessity, cab rental and hiring services are of one the solutions for commute. In this modern age of digital growth industries have become more customer service oriented based, hence cab rental services are one of them, and are increasing day by day,These provide customers service and is feedback driven, these deliver flexible services with competitive prices offered.

**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.

**Data provided**

The data provided were in csv format consisting test and train data namely (train\_cab and test ) first thing to start any data process is evaluation and observation of data to get proper understanding that what type of data it is and what it needs.

Train\_cab data had

16067 rows and 7 columns

Namely

|  |  |
| --- | --- |
| variables columns | Summary |
| Passenger\_count | Number of passengers in a cab |
| Fare\_amount | The fare for the service |
| Pickup\_datetime | The date and time of service availed |
| Pickup\_lattitude | Pickup co-ordinates of latitude |
| Pickup\_longitude | Pickup co-ordinates of longitude |
| Dropoff\_lattitude | Dropoff co-ordinates of latitude |
| Dropoff\_longitude | Dropoff co\_ordinates of longitude |

**Pre-Processing**

we need a predictive model, that will predict the dependent variable using the independent variables. we need to observe and alter the data before we start building models on the data these includes preprocessing steps such as exploring the data, data cleansing, data visualization, data plotting in different graph and plots, these steps is work under Exploratory Data Analysis, which includes following steps:

• Data exploration and Cleaning ( removing errors and outliers)

• Missing values treatment ( replacing with Na values and removing them)

• Outlier Analysis ( removing the outliers)

• Feature Selection (before plotting graphs features are selected)

• Features Scaling Skewness and Log transformation (the observed data was skewed)

• Visualization (plotting and observing graphs on the data)

**Modelling**

When all the Pre-Processing is done on data, we can now use the data for our model building on top of data. Selecting a model depends upon the problem statement and data set. As the data needs regression models as it contains numerical variables and timestamp analysis for time series data , we try few models on our preprocessed data and after comparing the output results we will select the best suitable model for our problem.

• Linear regression

• Decision Tree regression

• Random forest regression

We have also used hyper parameter tunings(in python) to check the parameters on which our model runs best.

Random Search CV (only in python)

Grid Search CV (only in python)

**Model Selection**

The final step is model selection which is based on the different output and results shown by different models. The model giving best values on the test data is selected and saved for further use

**Processing**

Exploring the data and cleaning it for missing values and outliers. Here we observe following points as per this project:

a. adjust the format and data type of the variables.

b. we observe some zero, missing values and very high values (outliers) in fare amount

c. Passenger count is set 6. As even the SUV’s don’t allow more than 6 people so we remove the rows having passengers counts more than 6 and less than 1.

d. 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 ranges.

Creating some new variables from the given variables. in data set variable name pickup\_datetime contains date and time for pickup. So we arrange it in a proper format and extract values from pickup\_datetime:

• Year

• Month

• Date

• Day of Week

• Hour

**Haversine formula for extracting distance**

We 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. new extracted variables are:

▪ fare\_amount

▪ pickup\_datetime

▪ pickup\_longitude

▪ pickup\_latitude

▪ dropoff\_longitude

▪ dropoff\_latitude

▪ passenger\_count

▪ year

▪ Month

▪ Date

▪ Day of Week

▪ Hour

▪ Distance

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

|  |  |
| --- | --- |
| Variable Names | Variable Data Types |
| fare\_amount | float64 |
| passenger\_count | Object |
| Year | Object |
| Month | Object |
| Date | Object |
| Day of Week | Object |
| Hour | Object |
| Distance | float64 |

**Now we have 8 variables in all together consisting of**

7 independent variables (passenger\_count, year, Month, Date, Day of Week, Hour, distance)

1 dependent variable (fare\_amount)

**Uniqueness in Variable**

We observe at the unique number in the variables which help us to decide the variable is categorical or numerical. by using python script ‘nunique’ we tried to find out the unique values in each variable. Variable Name Unique Counts

fare\_amount 450

passenger\_count 7

year 7

Month 12

Date 31

Day of Week 7

Hour 24

distance 15424

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

Continuous variables - 'fare\_amount', 'distance'.

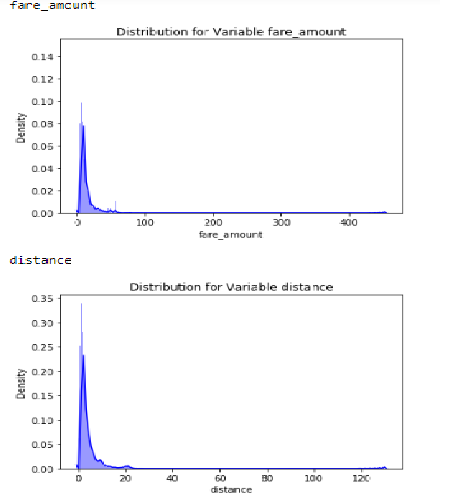
Categorical Variables - ' year', 'Month', 'Date', 'Day of Week', 'Hour', 'passenger\_count'

**Feature Scaling**

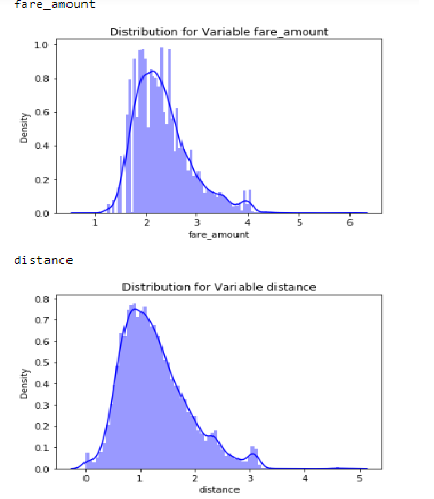
Skewness is asymmetry in a statistical distribution, in which the curve appears distorted or skewed either to the left or to the right. Skewness can be quantified to define the extent to which a distribution differs from a normal distribution. Here we tried to show the skewness of our variables and we 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:

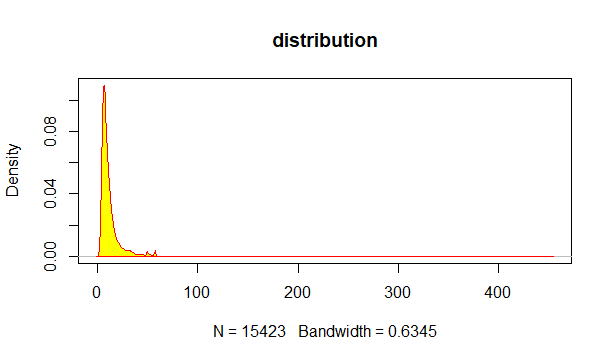
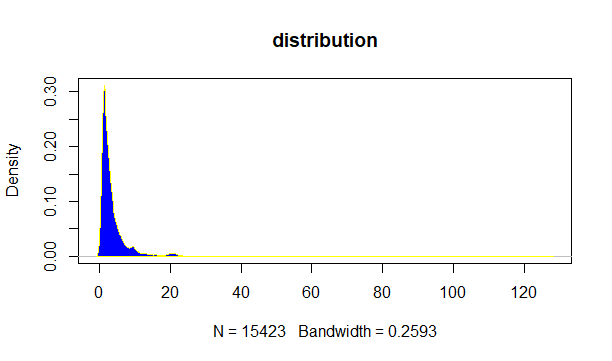
Feature scaing in python



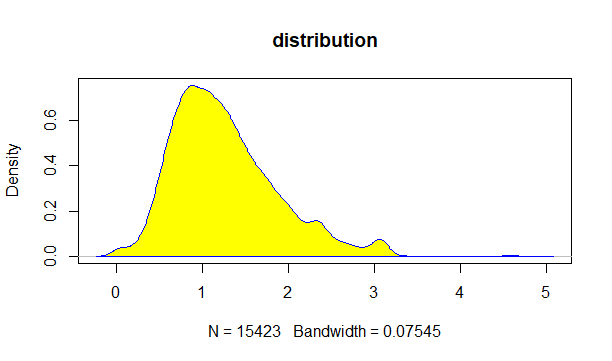
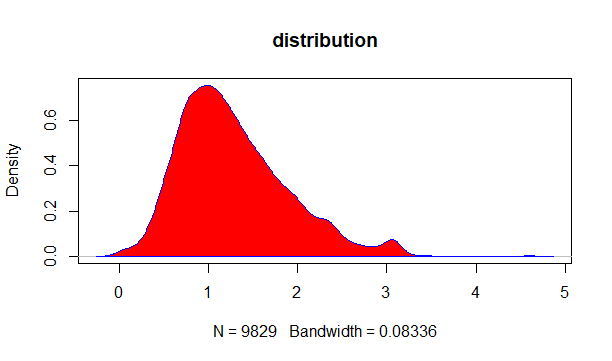
**After log transformation of skewed data the data appeared like**



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

**Feature scaling in R** 

**After log transformation**

**Modelling**

After preprocessing, we will use some regression models on our processed data to predict the target variable. Following are the models which we have built –

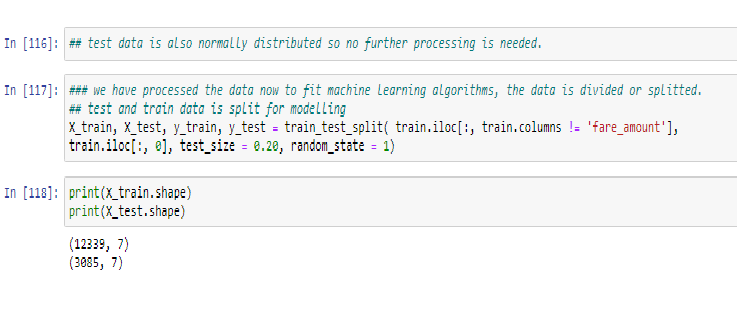
**Linear Regression**

**Decision Tree**

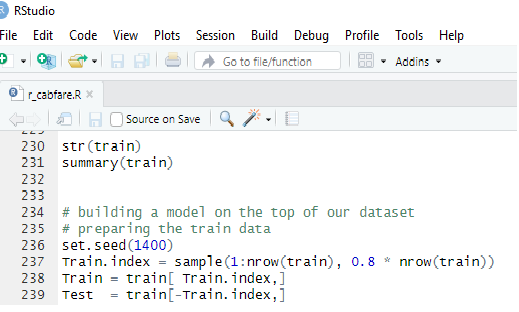
**Random Forest**

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.

**Splitting in python**



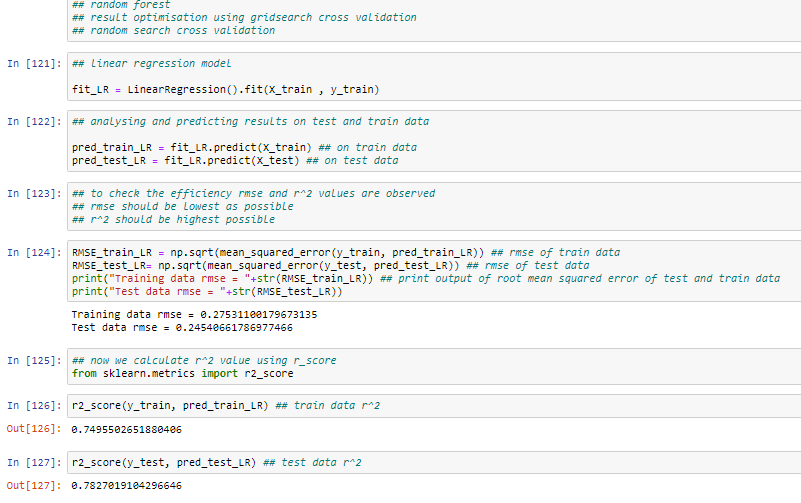
**Splitting in R**



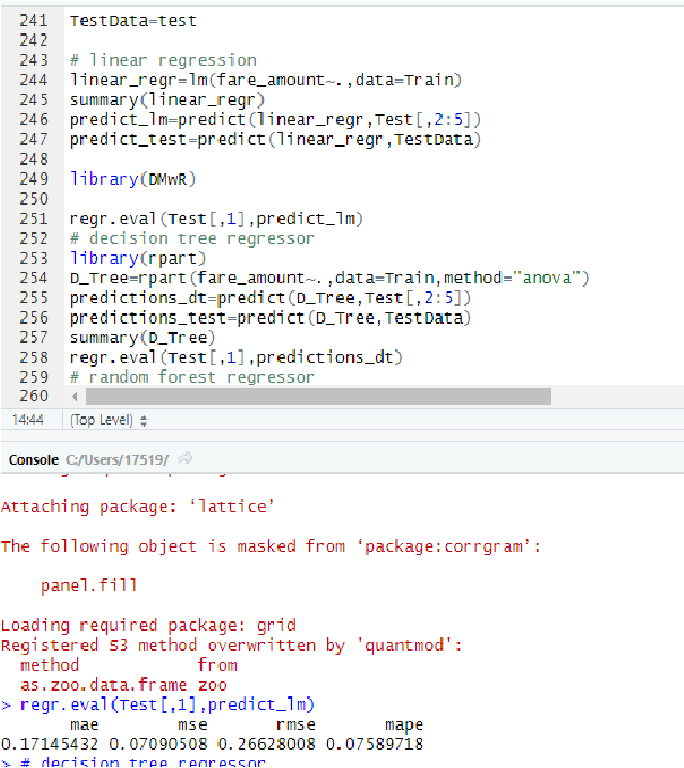
**Linear Regression**

Multiple linear regression is the most common form of linear regression analysis. Multiple regression is an extension of simple linear regression. It is used as a predictive analysis, when we want to predict the value of a variable based on the value of two or more other variables. The variable we want to predict is called the dependent variable (or sometimes, the outcome, target or criterion variable).

**In python**



**In R linear regression**

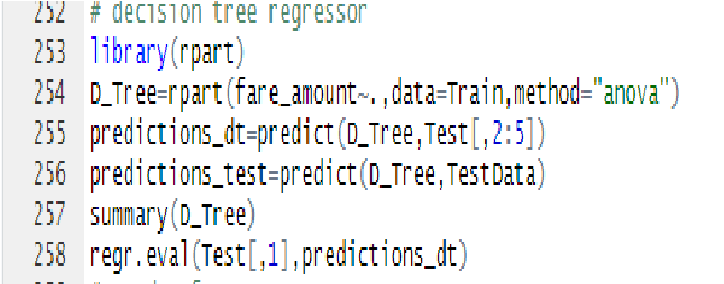


**Decision Tree**

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

**Decision Tree Algorithm in python**

**Decision tree in R**



regr.eval(Test[,1],predictions\_dt)

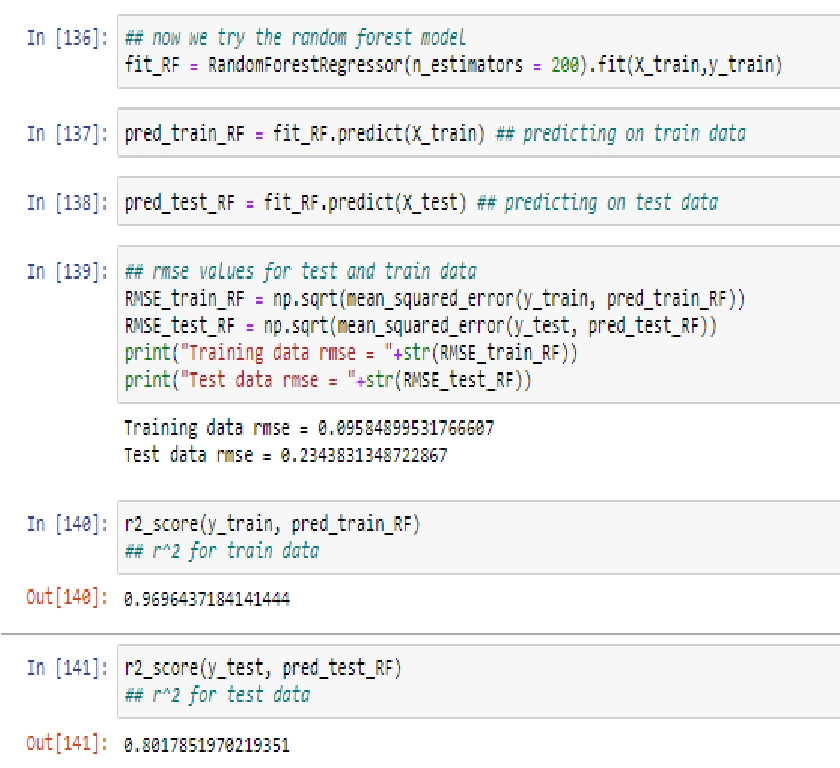
mae mse rmse mape

0.19294865 0.07600018 0.27568130 0.08562853

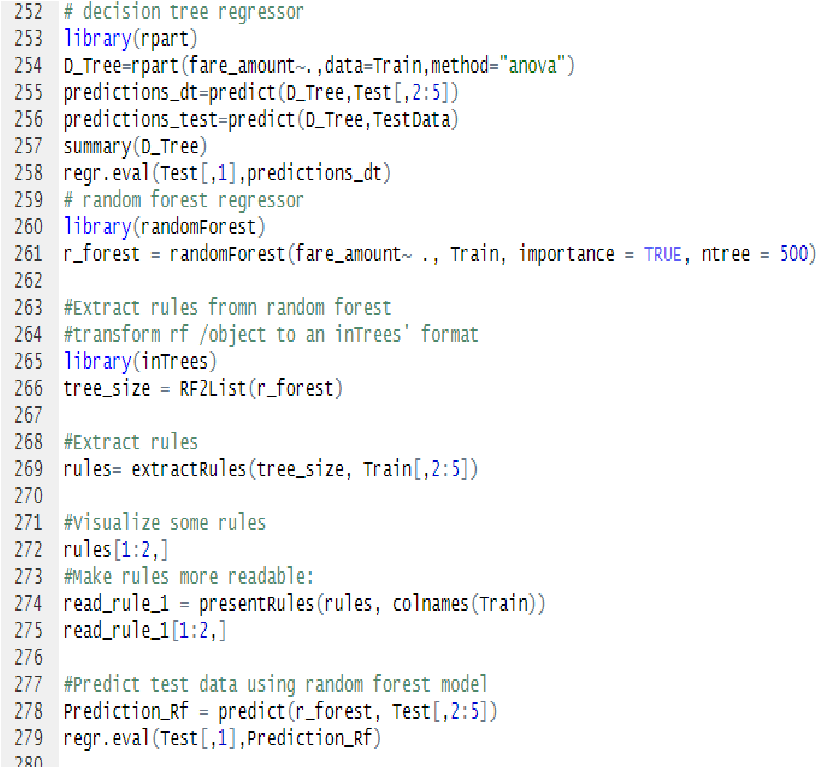
**Random Forest**

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. Random decision forests correct for decision trees' habit of overfitting to their training set. To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

**Random forest in python**



**Random forest in R**



mae mse rmse mape

0.21946783 0.09061675 0.30102617 0.09790464

**Hyper Parameters Tunings**

Model hyperparameters are set by the data scientist ahead of training and control implementation aspects of the model. The weights learned during training of a linear regression model are parameters while the number of trees in a random forest is a model hyperparameter because this is set by the data scientist. Hyperparameters can be thought of as model settings. These settings need to be tuned for each problem because the best model hyperparameters for one particular dataset will not be the best across all datasets. The process of hyperparameter tuning (also called hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.

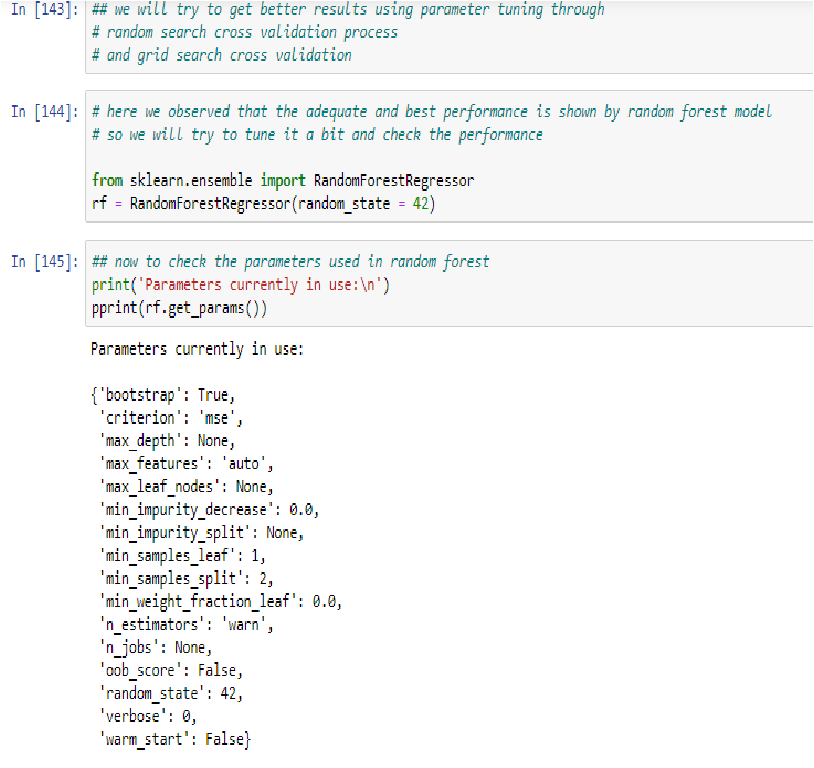
Here we have used two hyper parameters tuning techniques

**Random Search CV**

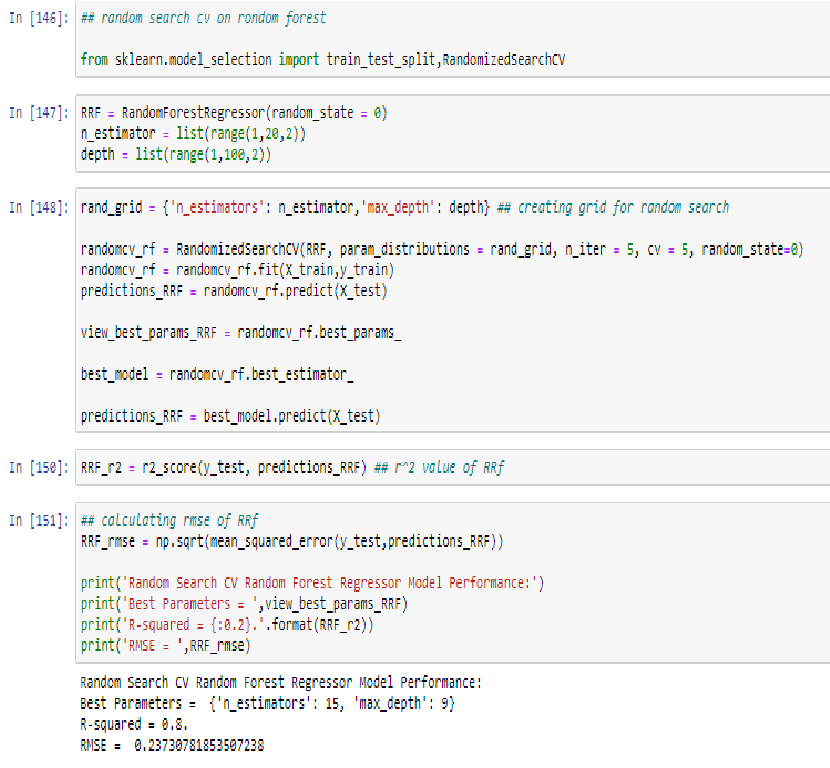
**Grid Search CV**

1. Random Search CV: This algorithm set up a grid of hyperparameter values and select random combinations to train the model and score. The number of search iterations is set based on time/resources. 2. 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 Random Search CV on Random forest



**Random search cross validation on random forest**



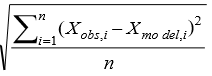
**Grid search cross validation on random forest**



**Model Evaluation in python**

The main concept of looking at what is called residuals or difference between our predictions f(x[I,]) and actual outcomes y[i].

I. RMSE (Root Mean Square Error): is a frequently used measure of the difference between values predicted by a model and the values actually observed from the environment that is being modelled.

RMSE = 

II. R Squared(R^2): is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains as to how much of the variance of the target variable is explained.

III. We have shown both train and test data results, the main reason behind showing both the results is to check whether our data is overfitted or not.

Results for python

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | RMSE | | R^2(squared) test | |
| train | test | train | Test |
| Linear regression | 0.27 | 0.24 | 0.74 | 0.78 |
| Decision tree | 0.29 | 0.28 | 0.70 | 0.70 |
| Random forest | 0.09 | 0.23 | 0.96 | 0.80 |

We use parameter tuning

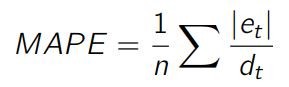
|  |  |  |
| --- | --- | --- |
| Tuned parameter model | RMSE test | R^2 squared value Test |
| Random search cv on random forest | 0.8. | 0.237 |
| Grid cv on random forest | 0.8. | 0.23 |

**Criteria and values for model selection in R**

**MAPE**

The**Mean Absolute Percentage Error (MAPE)** is one of the most commonly used KPIs to measure forecast accuracy.

MAPE is the sum of the individual absolute errors divided by the demand (each period separately). Actually, it is the average of the percentage errors.

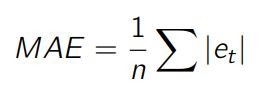


MAPE is a really strange forecast KPI.

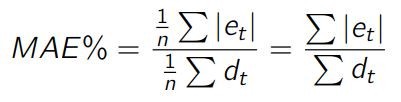
It is quite well-known among business managers, despite being a really poor-accuracy indicator. As you can see in the formula, MAPE divides each error individually by the demand, so it is skewed: high errors during low-demand periods will have a major impact on MAPE. Due to this, optimizing MAPE will result in a strange forecast that will most likely undershoot the demand. Just avoid it.

**MAE**

The **Mean Absolute Error (MAE)** is a very good KPI to measure forecast accuracy. As the name implies, it is the mean of the absolute error.



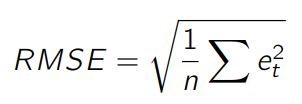
One of the first issues of this KPI is that it is not scaled to the average demand. If one tells you that MAE is 10 for a particular item, you cannot know if this is good or bad. If your average demand is 1000, it is of course astonishing, but if the average demand is 1, this is a very poor accuracy. To solve this, it is common to divide MAE by the average demand to get a %:

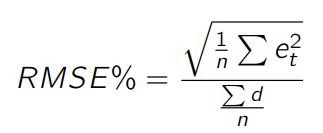


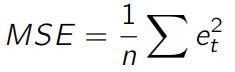
**MAPE/MAE Confusion**— It seems that many practitioners use the MAE formula and call it MAPE. This can cause a lot of confusion. When discussing forecast error with someone, I would always advise you to explicitly show how you compute the forecast error to be sure to compare apples and apples.

**RMSE**

The **Root Mean Squared Error (RMSE)** is a strange KPI but a very helpful one as we will discuss later. It is defined as the square root of the average squared error.

Just as for MAE, RMSE is not scaled to the demand. We can then define RMSE% as such,

Actually, many algorithms (especially for machine learning) are based on the **Mean Squared Error (MSE)**, which is directly related to RMSE.

MSE is used by many algorithms as it is faster to compute and easier to manipulate than RMSE. But it is not scaled to the original error (as the error is squared), resulting in a KPI that we cannot really relate to the original demand scale. Therefore, we won’t use it to evaluate our statistical forecast models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Models | mae | mse | Rmse | Mape |
| Linear regression | 0.17145432 | 0.07090508 | 0.26628008 | 0.07589718 |
| Decision tree | 0.19294865 | 0.07600018 | 0.27568130 | 0.08562853 |
| Random forest | 0.21946783 | 0.09061675 | 0.30102617 | 0.09790464 |

**Model Selection in python**

On the basis RMSE and R Squared results a good model should have least RMSE and max R Squared value. So, from above tables we can see:

• From the observation of all RMSE Value and R-Squared Value we have concluded that, • Both the models- Gradient Boosting Default and Random Forest perform comparatively well while comparing their RMSE and R-Squared value. • After this, I chose Random Forest CV and Grid Search CV to apply cross validation technique and see changes brought about by that. • After applying tunings Random forest model shows best results compared to gradient boosting. • So finally, we can say that Random forest model is the best method to make prediction for this project with highest explained variance of the target variables and lowest error chances with parameter tuning technique Grid Search CV.

Finally, I used this method to predict the target variable for the test data file shared in the problem statement. Results that I found are attached with my submissions.

Model selection in R

On the base of mape and rmse values of all three models (linear regression ,decision tree, random forest)

Evidently the mape and rmse values of random forest is far more better than other two meathods

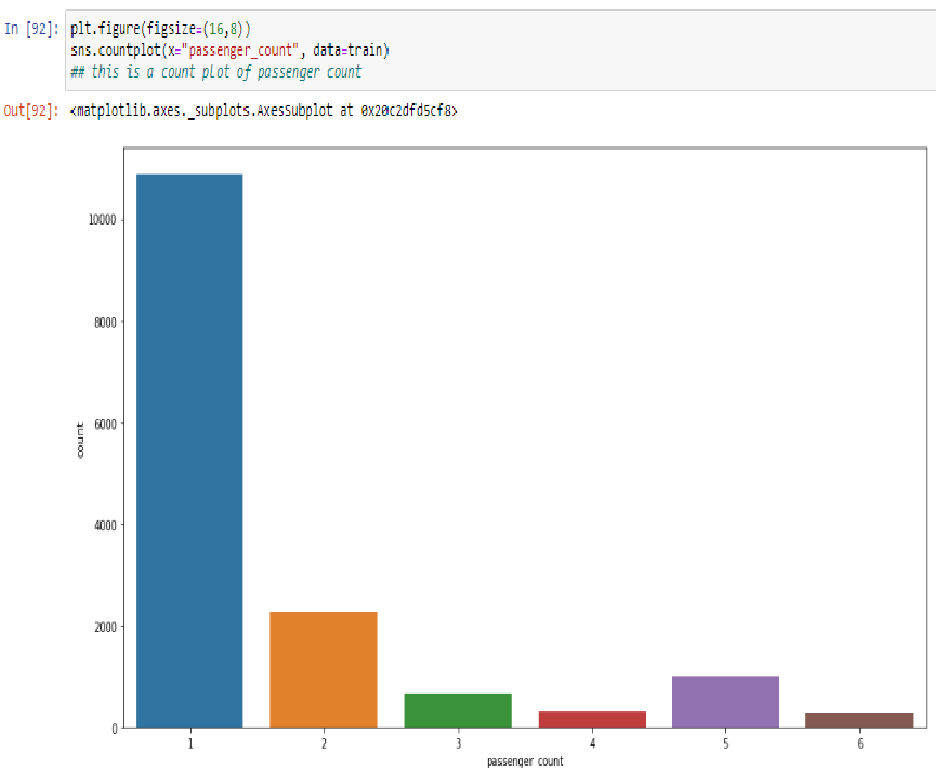
So we select the random forest meathod on the basis of the results observed. Higher the mape values better the model

And lower the rmse values best the model works.

**Visualizations**

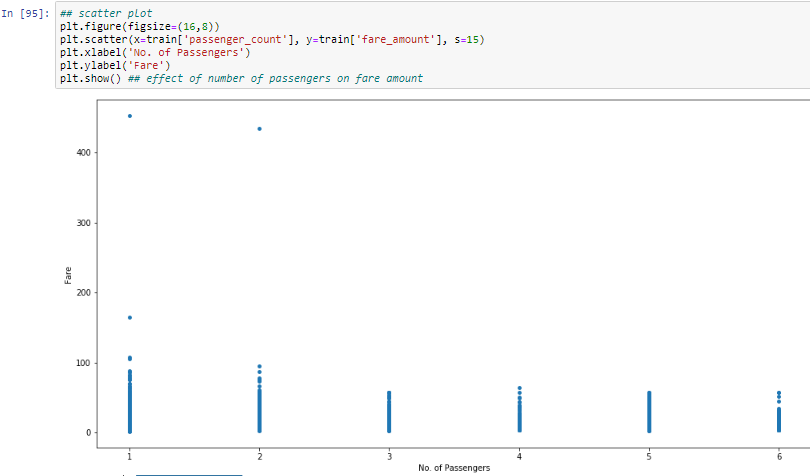
**1. Number of passengers and fare**

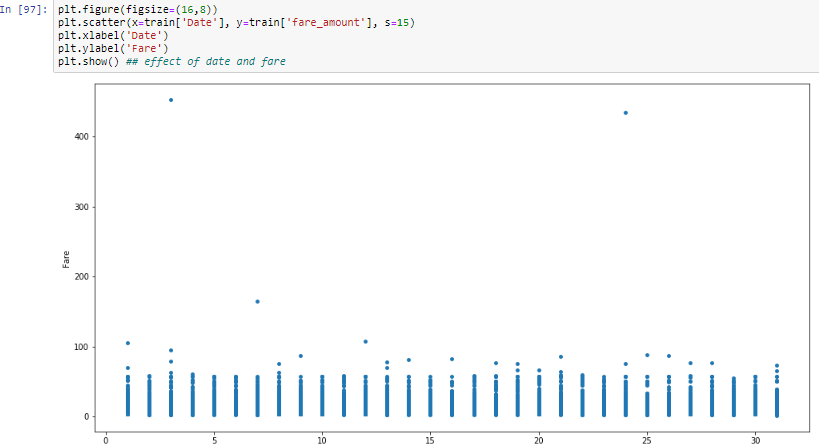
We can see in below graph that single passengers are the most frequent travelers, and the highest fare also seems to come from cabs which carry just 1 passenger.



**2. Date of month and fares**

The fares throughout the month mostly seem uniform.

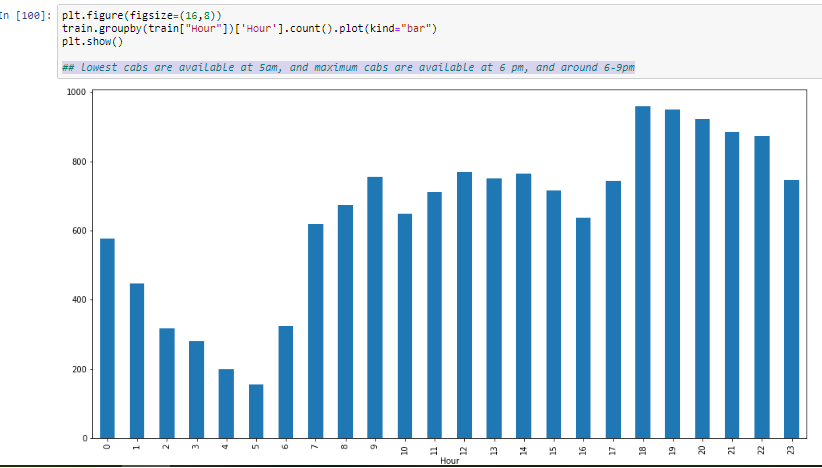




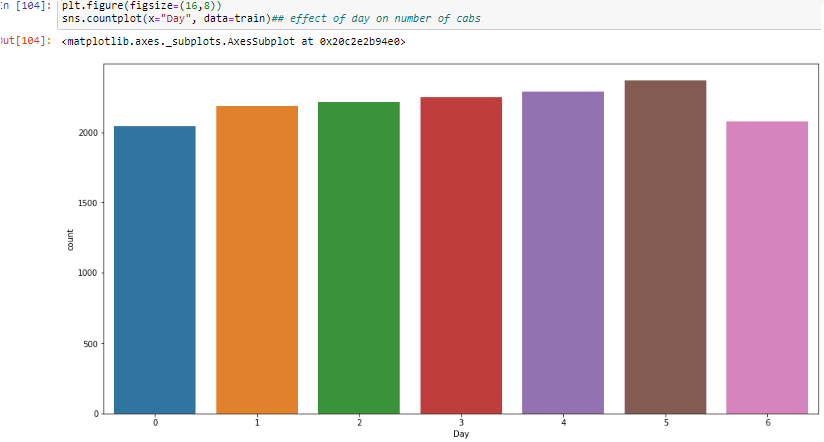
**3. Hours and Fares**

cab fares are high at 7 am, and 22 pm

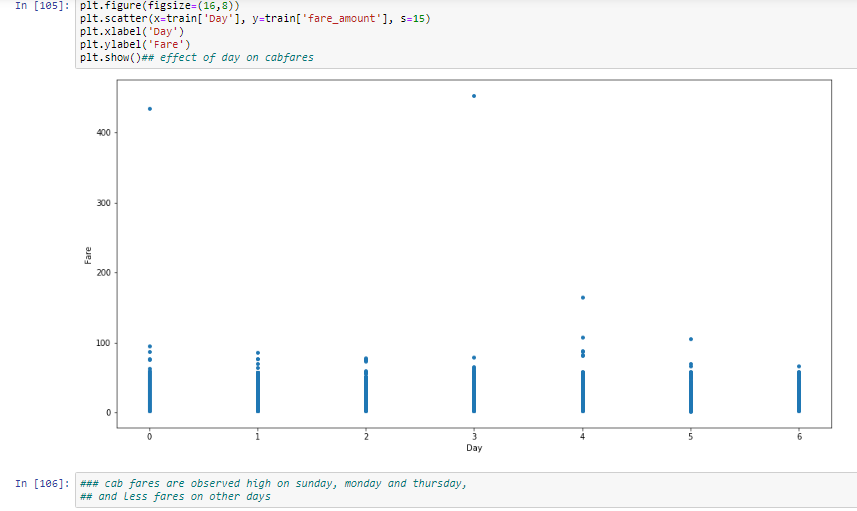
lowest cabs are available at 5am, and maximum cabs are available at 6 pm, and around 6-9pm



**4.Impact of Day on the Number of Cab rides :**

Observation : The day of the week does not seem to have much influence on the number of cabs ride

**5. effect of cab fare on the days**



**6. effect of distance on fares**

