A Project Report on

**Cab Fare Prediction**

Submitted by :

**Abhishek Darekar**

## TABLE OF CONTENTS

**CHAPTER TITLE PAGE NO.**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | | Title of Chapter | Page No. |
| 01 | | Introduction | 3 |
|  | 1.1 | Problem Statement | 3 |
|  | 1.2 | Data | 3 |
| 02 | | Exploratory data Analysis | 4 |
|  | 2.1 | Change datatype of required Variables | 4 |
|  | 2.2 | Remove unrealistic values(Outliers) from Attributes | 5 |
| 03 | | Missing Value Analysis | 6 |
|  | 3.1 | Find Missing value percentage Column wise | 6 |
|  | 3.2 | Impute missing values | 7 |
| 04 | | Outlier Analysis | 8 |
|  | 4.1 | Find Outliers and set to NA | 8 |
|  | 4.2 | Impute the values set to NA | 9 |
| 05 | | Feature Engineering | 10 |
|  | 5.1 | Derive new Features | 10 |
| 06 | | Calculate the distance using haversine | 11 |
| 07 | | Feature Selection | 12 |
|  | 7.1 | Correlation analysis and Delete the insignificant variables | 12 |
| 08 | | Feature Scaling | 14 |
| 09 | | Split Data into Train and Validation (test) | 15 |
| 10 | | Modelling | 16 |
|  | 10.1 | Linear Regression | 17 |
|  | 10.2 | Decision Trees | 19 |
|  | 10.3 | Random Forest | 21 |
|  | 10.4 | Ensemble Learning XG Boost | 23 |
| 11 | | Finalize the Model | 25 |
|  | 11.1 | Model Evaluation | 25 |
|  | 11.2 | Model Selection | 26 |
| 12 | | Save The Model | 27 |
| 13 | | Apply model on Test Data. | 28 |
| 14 | | Reference | 39 |
| 15 | | Appendix | 30 |

Chapter 1 : Introduction

### 1.1 Problem Statement

You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city.

### 1.2 Data

Here we have a data set with following features, we need to go through each and every variable of it to understand and for better functioning.

Size of Dataset Provided: -Rows : 16067, Columns : 7 (includes 1 dependent variable)

Missing Values: Yes

Outliers Presented: Yes

Below mentioned is a list of all the variable names and what they stand for:

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.

Chapter 2 :  **Exploratory data Analysis(EDA)**

The EDA is an approach--not a set of techniques, but an attitude/philosophy about how a data analysis should be carried out.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand.

### 2.1 Change datatype of required Variables

Attribute Name Data Type.

|  |  |  |
| --- | --- | --- |
| **0** | fare\_amount | object |
| **1** | pickup\_datetime | object |
| **2** | pickup\_longitude | float64 |
| **3** | pickup\_latitude | float64 |
| **4** | dropoff\_longitude | float64 |
| **5** | dropoff\_latitude | float64 |
| **6** | passenger\_count | float64 |

Before proceeding we check the data types of variable, Change the data type of variables which are not appropriate so they can be processed correctly moving forward.

We change the Data type of the following variables :

1. fare\_amount : Change from object to numeric Data type.
2. Pickup\_datetime : Change from object to datetime Data type .

### 2.2 Remove unrealistic values(Outliers) from Attributes

Such incorrect values will mean we will be feeding wrong data to our model.

To avoid this we will be removing all the values which are incorrect.

The following are the parameters for removing values from each attribute :

1. passenger\_count : Delete all values less than 1 and greater than 6.
2. Fare\_amount : Delete all values less than 0.
3. Pickup\_longitude and latitude : All values outside the range

-Longitude Range : -180 to 180.

- Lattitude: -90 to 90.

4. dropoff\_longitude and latitude : Same as above

Train Data Shape after performing these Deletions :

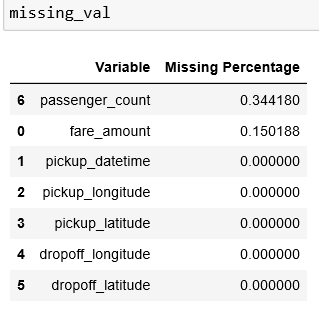
(15980, 7) i.e 15980 Rows and 7 Columns.

We will be Dealing with these missing values in the next step.

Chapter 3 : Missing value Analysis.

There were missing values in the dataset in addition to that we deleted Values in the last stage. Which has resulted in our dataset having the below missing values.

### 3.1 Find Missing value percentage Column wise



The missing percentage indicates the number of missing values in a attribute in relation to the whole dataset, it has been arranged in decending order.

Now we will impute these using a suitable method in the next step.

### 3.2 Impute missing values

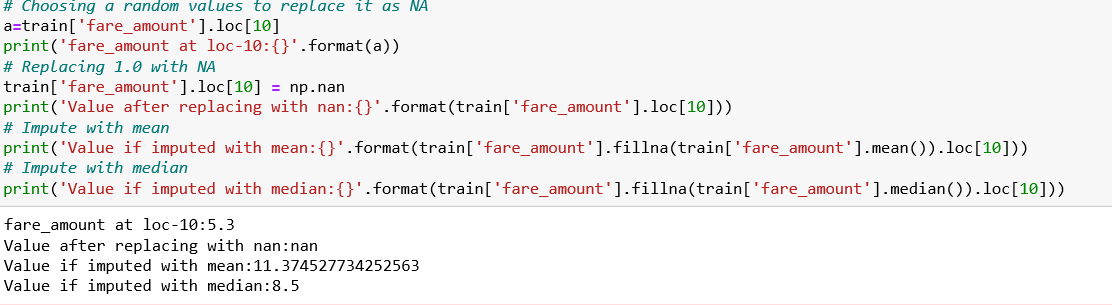
We cannot just impute the missing values with any values. We need data that makes sense and is as accurate as possible.

There are various methods like : mode, mean, median, Knn .

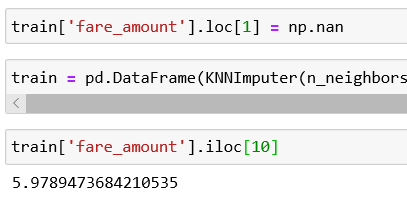
1.**Mode** : We will not be using mode because the data will be Biased towards the majority values in that column.

Test the performance of mean, median, Knn by imputing a Known value from a cell after we delete it and choosing the method that gives the closest answer.

Using Mean and Mode :



Using KNN imputation :



We Use KNN Imputation as we can see from this KNN gives the imputed value closest to the original value.

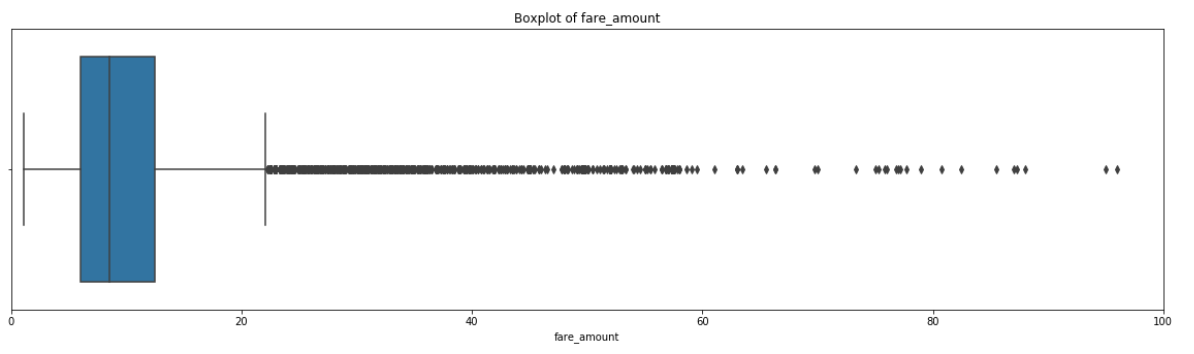
Chapter 4 : Outlier Analysis

Outliers can have a drastic effect in the data. Few Extreme data points can change the mean of the data and make the data like something which is not the overall picture.

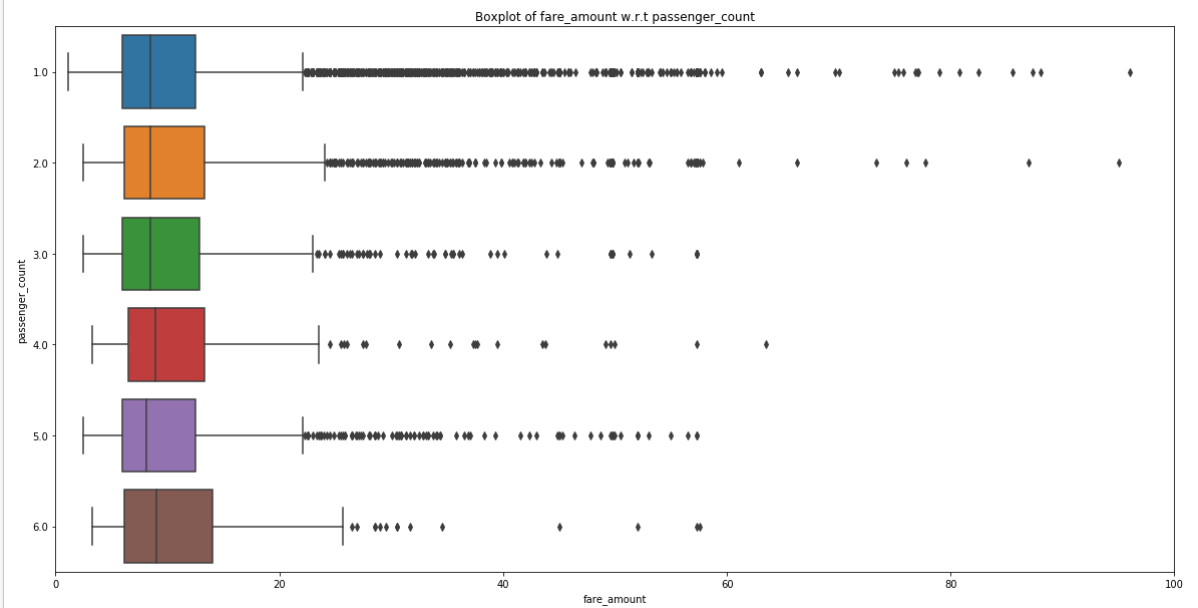
Too prevent such Data from entering the model phase, we will Find the outliers and impute them using suitable values.

### 4.1 Find Outliers and set to NA

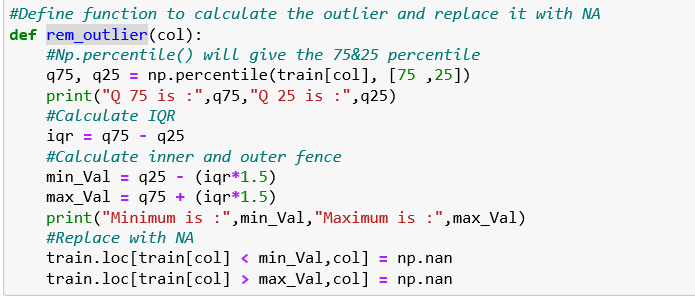
1. Boxplot of fare\_amount to see all the outliers



2.Boxplot of fare\_amount wrt Passenger\_count to see all the outliers.



We have defined a function to deal with outliers i.e all values less than the 25 percentile and all values greater than the 75 percentile of the boxplot.



Now we have identified and deleted the variables.

Chapter 5 : Feature Engineering

**Feature engineering** is the process of using domain knowledge to extract features from raw data via data mining techniques. These features can be used to improve the performance of **machine learning** .

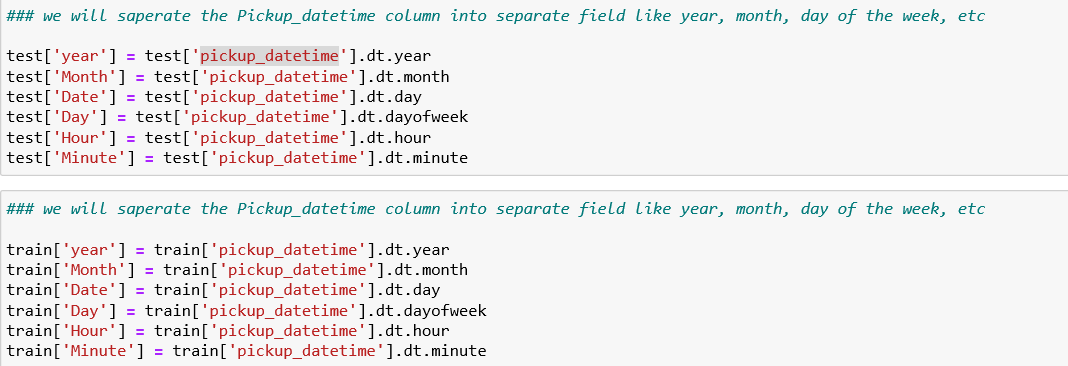
These new features can be created by adding or breaking down the current Features.

In this case we will breakdown the “pickup\_datetime” to create several new features like :

Year, month, day, Date, hour, minute.

5.1 Derive new Features

We do this for both train and test :



Variable Meaning

|  |  |
| --- | --- |
| year | Year of pickup |
| month | Month of pickup |
| Date | Date of pickup |
| Day | Weekday of pickup |
| Hour | Hours of pickup |
| minute | Minute of pickup |

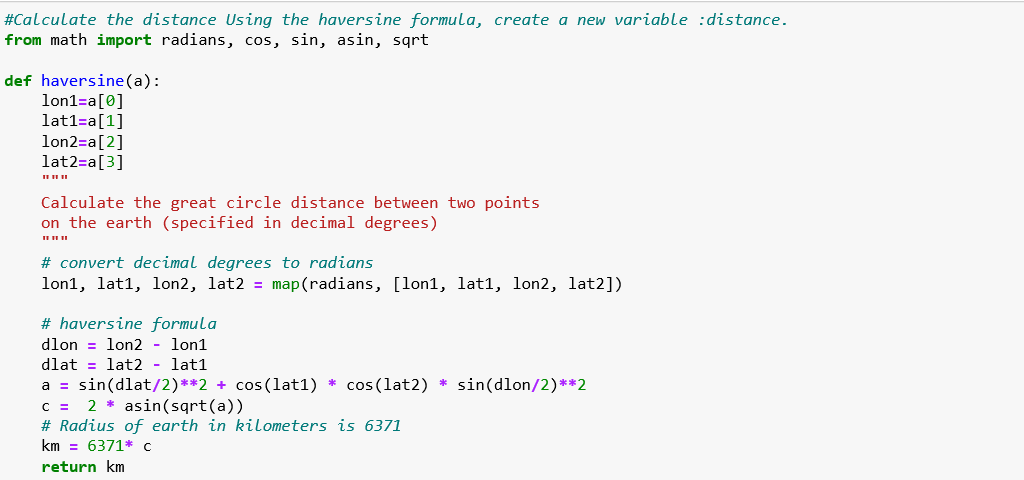
There new features will give us in detailed view of each entry of the Data.

Chapter 6 : Calculate the distance using harvensine.

We have been provided with pickup and dropoff longitude and latitude, so we need to use this to calculate the distance travelled by the passenger.

It will be a feature engineering step as we will use various variables to calculate this attribute.

Create the function for haversine :



Use the function to calculate the distance for both train and test data.



Chapter 7 : Feature Selection.

Feature Selection is the process where we select the features which contribute most to your prediction variable or output.

Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features.

In this step we will remove all the insignificant attributes which contain similar data and does not provide any different information.

We will do correlation analysis as we have numeric Data :

7.1 Correlation analysis and Delete the insignificant variables

**Correlation Analysis** :

1.Works only on numeric Data.

2.Shows how 2 variables are related to each other.

3. Range -1 to +1 :

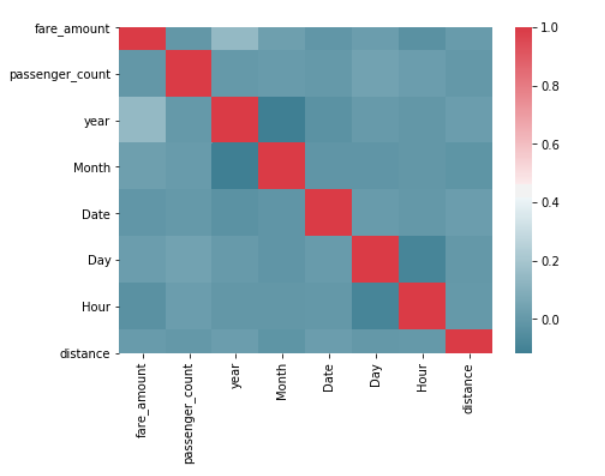
-1= Negatively correlated

0= No correlation

+1= positively correlated

1. Ideally Independent variables must have no correlation .

There should be high correlation between Dependent variable.



**Observation** :

Extreme Red = Highly positively correlated data.

Extreme Blue= Highly Negatively Correlated data.

-1.year and month are highly negatively correlated

-2.Day and hour are highly negatively correlated

-So we can delete one from each so that we don’t have redundant information.

Chapter 8 : Feature Scaling.

Feature Scaling is done to limit the range of the variables so that they can be compared on common ground.

This will make sure one attribute does not have too much influence on the whole model.

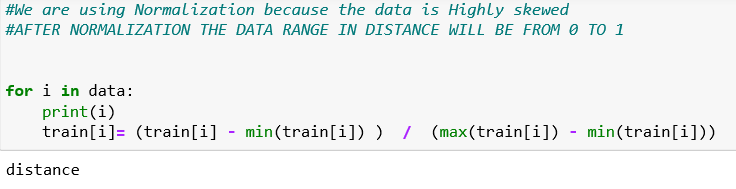
This is done only on continuous variables.

There are 2 main methods of doing this :

1.Normalization : Used if Data is not uniformly distributed (Skewed Data).

2.Standardization : Used if Data is well distributed.

Need to check data distribution before deciding which method to use. As the Data is left skewed we opted for Normalization on the below Data :



Normalization:

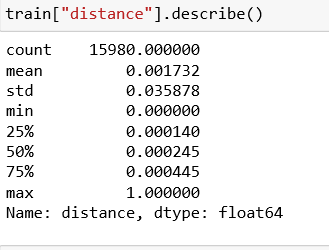
1.Brings Data to Scale.

2.Range between 0 to 1.

3.Reduces unwanted variation in data.

4.It is sensitive to outliers.

Ex: Distance Data after normalization :



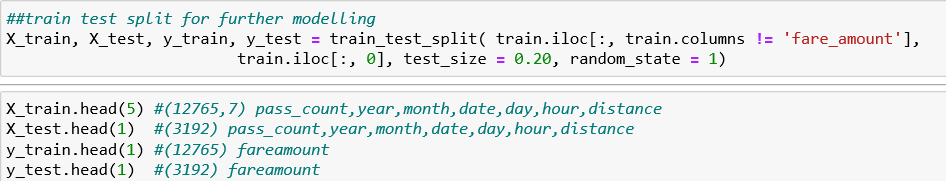
Chapter 9 : Split Data into Train and Validation (test)

We will be splitting the Data into train\_data and test\_data.

Train\_data will be used to build the model and test\_data will be run on that model to check the performance of the model.

We will be using 20% Data as test\_data and 80% data as train\_data.

We will use train\_test\_split to do so :



Here we can also see the number of rows, columns in each variable, Which gives an idea about how the data is exactly been split.

Chapter 10 : Modelling

After thorough preprocessing and exploratory data analysis phase, the data is ready to enter the model building phase.

In this phase multiple machine learning algorithm will be used and tested to predict the test case i.e the cab fare prediction.

Here our target variable i.e. fare amount is numeric (predicting and forecasting type of problem) so that here we are using regression models to predict test case.

Different Models we have implemented are :

1.Linear Regression Model (Multiple).

2.Decision tree model.

3.Random Forest Model.

10.1 Linear Regression Model (Multiple).

**Linear Regression** is one of the basic and most common method of prediction.

It finds a linear relationship between the target and one or more predictors.

It means the target variables should be continuous in nature.

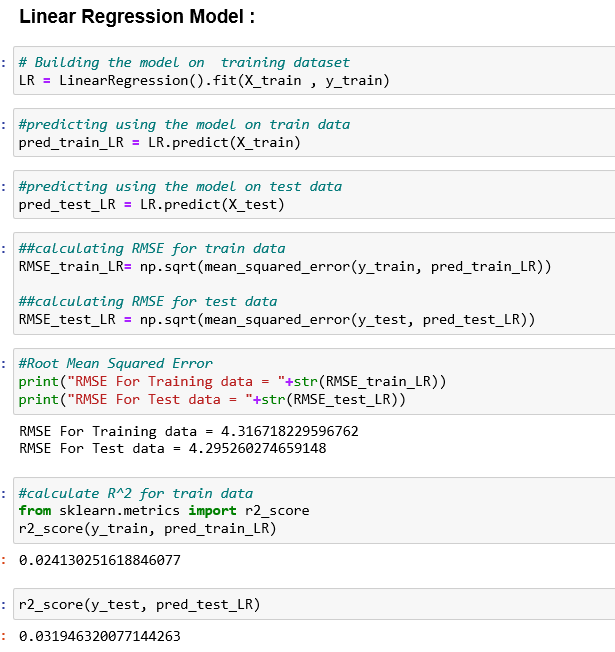
**Multiple regression** is an extension of simple linear regression.

It is used when we want to predict the value of a variable based on the value of two or more other variables.

The main idea is to identify a line that best fits the data.

This algorithm is not very flexible, and has a very high bias.

Applied as below :



Performance of the Model :

**Metric Python**

|  |  |
| --- | --- |
| Root Mean Squared Error (Train) | 4.316718229596762 |
| Root Mean Squared Error (Test) | 4.295260274659148 |
| r2\_score (Train) | 0.024130251618846077 |
| r2\_score (Test) | 0.031946320077144263 |

10.2 Decision tree model.

Decision tree can be used for both classification and Regression.

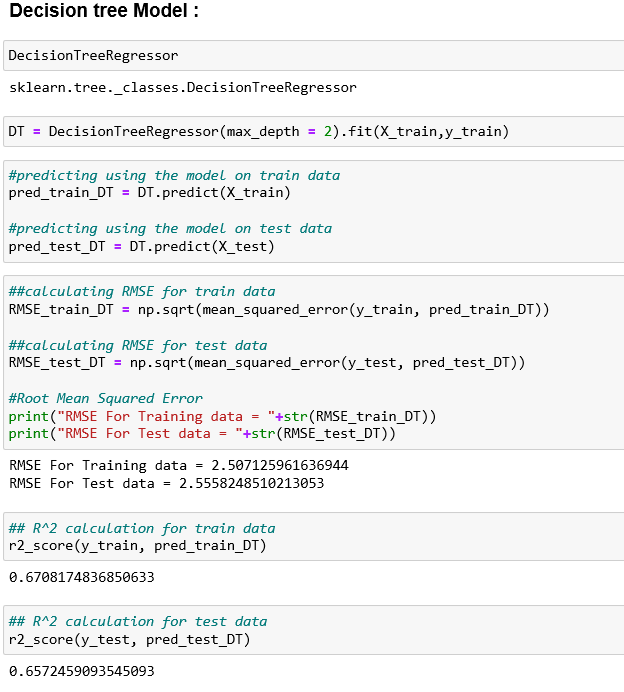
A **decision tree** is a decision support tool that uses a tree like model of decisions and their possible consequences.

Decision tree model is a predictive model based on Branching series of Boolean test.

Each branch connects nodes with “and” and multiple branches are connected by “OR”

It is Easy to understand for non technical user too.

Applied as below :



Performance of the Model :

Decision Tree Model :

**Metric Python**

|  |  |
| --- | --- |
| Root Mean Squared Error (Train) | 2.507125961636944 |
| Root Mean Squared Error (Test) | 2.5558248510213053 |
| r2\_score (Train) | 0.6708174836850633 |
| r2\_score (Test) | 0.6572459093545093 |

10.3 Random Forest Model.

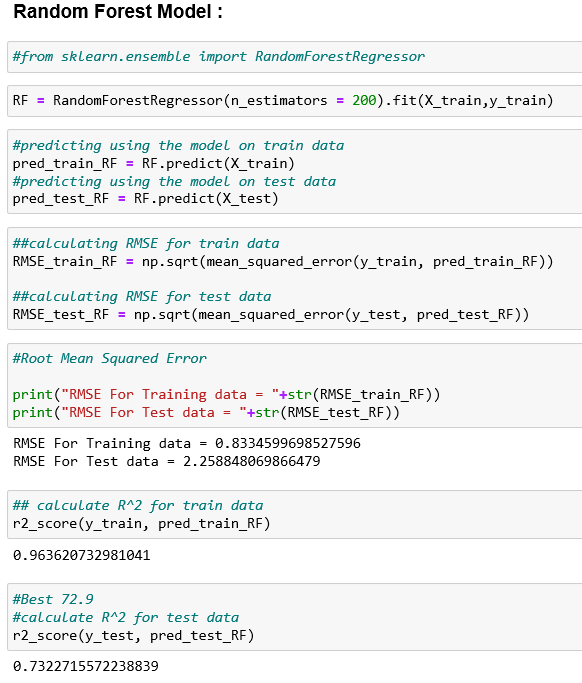
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks.

They 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.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

Applied as below :



Performance of the Model :

**Metric Python**

|  |  |
| --- | --- |
| Root Mean Squared Error (Train) | 0.8334599698527596 |
| Root Mean Squared Error (Test) | 2.258848069866479 |
| r2\_score (Train) | 0.963620732981041 |
| r2\_score (Test) | 0.7322715572238839 |

# THESE 2 RMSE VALUES CAUSE UNDER FITTING.

10.4 **Ensemble technique ---- XGBOOST**

XGBoost is a type of boosting method.

Boosting effectively learns from its mistakes with each iteration.

It is a type of boosting that aggregates a number of weak models to create a strong model.

Weak model : Slightly better than random guessing.

Strong model : Strongly correlated with true classification.

Trees in Boosting are not independent as they learn from previous models so they depend on them for learning.

Decision is based on weighted voting, where weight depends on how well a model did. Unlike Random forest where Majority voting Is used.

Applied as below :



Performance of the Model :

**Metric Python**

|  |  |
| --- | --- |
| Root Mean Squared Error (Train) | 2.0416335896850657 |
| Root Mean Squared Error (Test) | 2.18771617456052 |
| r2\_score (Train) | 0.7817068755181392 |
| r2\_score (Test) | 0.7488677863292859 |

Chapter 11 : Finalize the model.

After implementing different models they need to be studied and one model has to be finalized.

Selection of the final model depends on the type of dataset , different metrics considered.

11.1 Model Evaluation

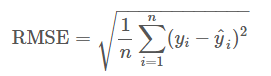
We will be using 2 metrics to evaluate the model :

1.RMSE

2.R Squared(R^2)

1. **RMSE (Root Mean Square Error)**:

* Itis 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.
* It explains how close the actual data points are to the model’s predicted values. It measures standard deviation of the residuals.
* The RMSE for your training and your test sets should be very similar if you have built a good model. If the RMSE for the test set is much higher than that of the training set, it is likely that you've badly over fit the data, i.e. you've created a model that works well in sample, but has little predictive value when tested out of sample.
* RMSE is the standard deviation of the prediction errors.
* It is a measure if goodness of fit in regression line.



2. **R Squared(R^2) :**

* It measures the proportion of the variation in your dependent variable explained by all of your independent variables in the model.
* It assumes that every independent variable in the model helps to explain variation in the dependent variable.
* In reality, some variables don't affect dependent variable and they don't help building a good model.
* Value of R-squared is between 0-1, where 0 means independent variable unable to explain the target variable and 1 means target variable is completely explained by the independent variable.
* So Higher the R-squared, the better the model fits your data.

11.2 Model Selection

**Model** **RMSE** **R Squared**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Train** | **Test** | **Train** | **Test** |
| **Linear** **Regression** | **4.3167** | **4.2952** | **0.0241** | **0.0319** |
| **Decision** **tree** | **2.5071** | **2.5558** | **0.6708** | **0.6572** |
| **Random** **Forest** | **0.8334** | **2.2588** | **0.9636** | **0.7322** |
| **XG** **Boost** | **2.0416** | **2.1877** | **0.7817** | **0.7488** |

As defined in the model evaluation stage these are the RMSE and R Squared scored of all the models we implemented.

What we desire is a Lower values of RMSE and higher value of R-Squared Value , which indicate better fit of model.

**Conclusion :**

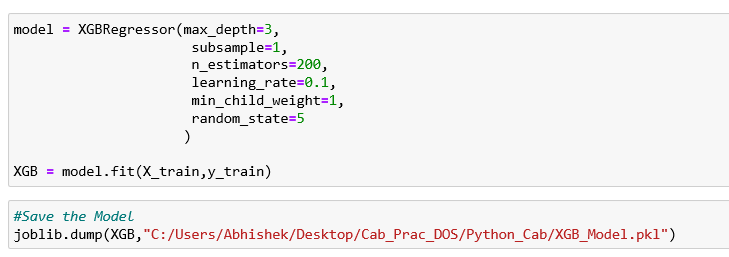
* As we can see from the scores XG Boost gives us the best results.
* It gives minimum RMSE value and Acceptable R Squared scores.
* The values difference of RMSE score for train and test data is very less so there is no any problem of model over-fitting or under-fitting.
* So we finalize the XG Boost model and apply it on test data provided to predict fare amount.

Chapter 12 : Save the model.

We need to save the trained models in a file and restore them in order to reuse it.

Here we are saving only the finalized model i.e XG Boost model.

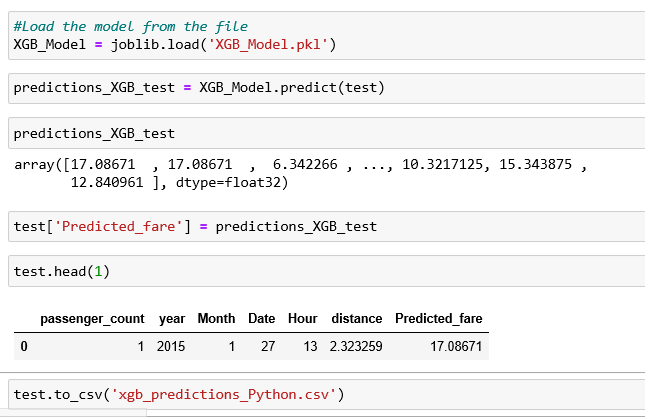
The saving of model is a general practice which will save time and efforts because once saved the model can be simply called and applied on the data.



Chapter 13: Apply model on Test Data.

After a model has been save it can be imported anytime and applied on the Data.

We import the XG BOOST model which we have saved and use it to predict the cab fare from the test dataset which was provided with the problem statement.



Chapter 14 : References

1. Distance calculation using Haversine algorithm.

<https://www.movable-type.co.uk/scripts/latlong.html>

Chapter 15 : Appendix

1. Python Code Attached.
2. R code Attached.