Cab Fare Prediction Project Report

By: Chinmayi Lekurwale

Table of Contents

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 analytic s for fare prediction.

Data Description

We are provided with 2 data sets. i.e Train data and Test data.

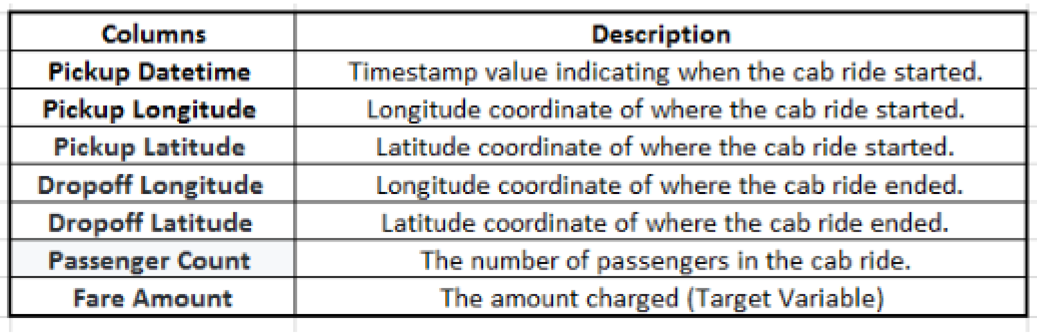
Train data is provided for training our models. Prior to training the data is snet through exploratory Data Analysis (EDA) and Feature Engineering. The best model is choosed the predict the values on Test Dataset.

Train Data:

Size of Dataset Provided: - 16067 rows, 7 Columns (including dependent variable)

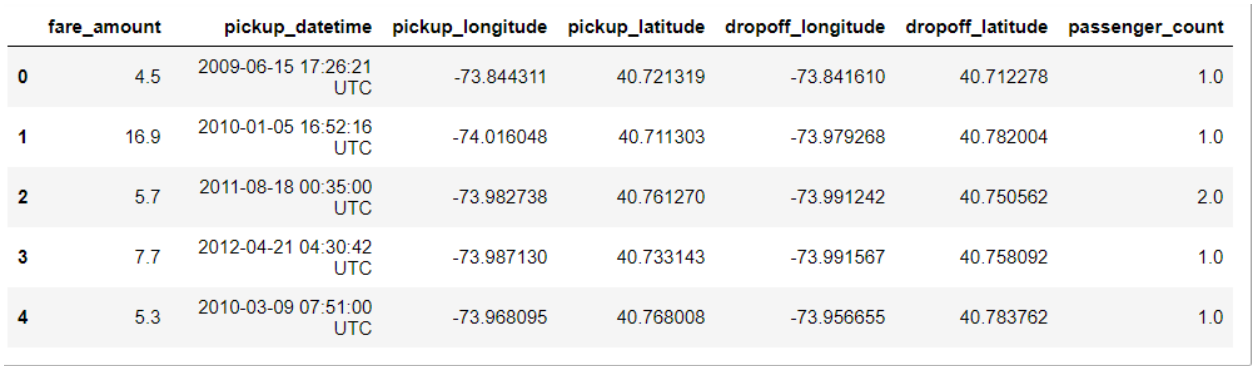
Missing Values: Yes

Outliers Presented: Yes



Columns	Description
Pickup Datetime	Timestamp value indicating when the cab ride started.
Pickup Longitude	Longitude coordinate of where the cab ride started.
Pickup Latitude	Latitude coordinate of where the cab ride started.
Dropoff Longitude	Longitude coordinate of where the cab ride ended.
Dropoff Latitude	Latitude coordinate of where the cab ride ended.
Passenger Count	The number of passengers in the cab ride.
Fare Amount	The amount charged (Target Variable)

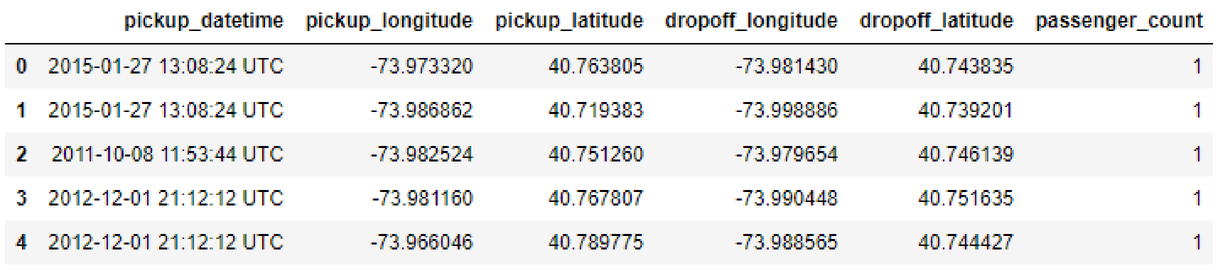
Below is the screenshot of the first five rows of the data:



	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

Test Data:

Test data consists of 9914 and 6 variables.



	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2015-01-27 13:08:24 UTC	-73.973320	40.763805	-73.981430	40.743835	1
1	2015-01-27 13:08:24 UTC	-73.986862	40.719383	-73.998886	40.739201	1
2	2011-10-08 11:53:44 UTC	-73.982524	40.751260	-73.979654	40.746139	1
3	2012-12-01 21:12:12 UTC	-73.981160	40.767807	-73.990448	40.751635	1
4	2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	1

Below are the details of the columns with their respective datatypes:

```
fare_amount          16043 non-null object
pickup_datetime      16067 non-null object
pickup_longitude     16067 non-null float64
pickup_latitude      16067 non-null float64
dropoff_longitude    16067 non-null float64
dropoff_latitude     16067 non-null float64
passenger_count      16012 non-null float64
```

Above information states that there are missing values in fare_amount and passenger_count variables.

Also, we have two columns (fare_amount,pickup_datetime) with object as a datatype and rest of the variables are of float64 datatype.

Data Preprocessing:

Data preprocessing is a **data** mining technique that involves transforming raw **data** into an understandable format.

- The type of cleaning and engineering strategies used usually depends on the business problem and type of target variable, since this will influence the algorithm and data preparation requirements.
- The most important part of data cleaning is the experimentation and checking how applying one or many of these strategies affects your ability to predict or classify in the model.

The datatypes are changed and pickup_datetime column is separated into 6 more columns.

Below is the code for the same:

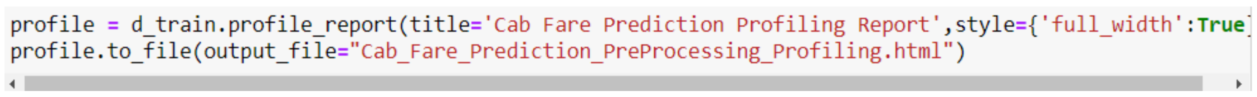
```
d_train['Year'] = d_train['pickup_datetime'].dt.year
d_train['Month'] = d_train['pickup_datetime'].dt.month
d_train['Date'] = d_train['pickup_datetime'].dt.day
d_train['Day'] = d_train['pickup_datetime'].dt.dayofweek
d_train['Hour'] = d_train['pickup_datetime'].dt.hour
d_train['Minute'] = d_train['pickup_datetime'].dt.minute
```

The new columns are Year,Month,Date,Date,Day,Hour and Minute.

Pre Processing Profiling:

We run a Pre Profiling on the data set which gives us information regarding the columns and detailed report regarding the coorelation between the variables. A detailed HTML report is generated which provides us with the visualization wh

ich further can be used for detailed analysis.



```
profile = d_train.profile_report(title='Cab Fare Prediction Profiling Report',style={'full_width':True
profile.to_file(output_file="Cab_Fare_Prediction_PreProcessing_Profiling.html")
```

Observation:

- **fare_amount** value ranges from -3 to 54343.Thr data is highly skewed with skewness value as 125.40.
- **passenger_count** value ranges from 0 to 5345 and skewness value as 84.61.
- **dropoff_latitude** has 312 (1.9% of total data) zeros.
- **dropoff_longitude** has 314 (2.0% of total data) zeros.
- **pickup_latitude** has 315 (2.0% of total data) zeros.
- **pickup_longitude** is highly correlated with dropoff_longitude with correlation value as 0.96. Hence this feature needs to rejected.

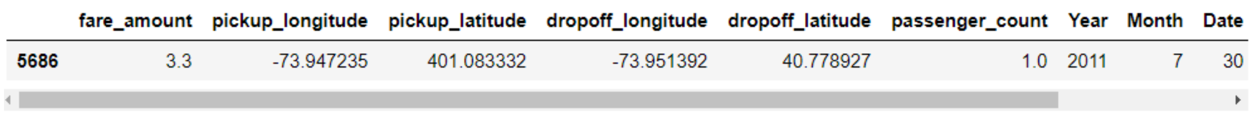
Missing Values:

```
fare_amount          25
pickup_longitude      0
pickup_latitude       0
dropoff_longitude     0
dropoff_latitude      0
passenger_count      55
Year                  0
Month                 0
Date                  0
Day                   0
Hour                  0
Minute                0
dtype: int64
```

Dropping 24+55=79 rows from dataset as there are missing values in fare_amount and passenger_count variables.

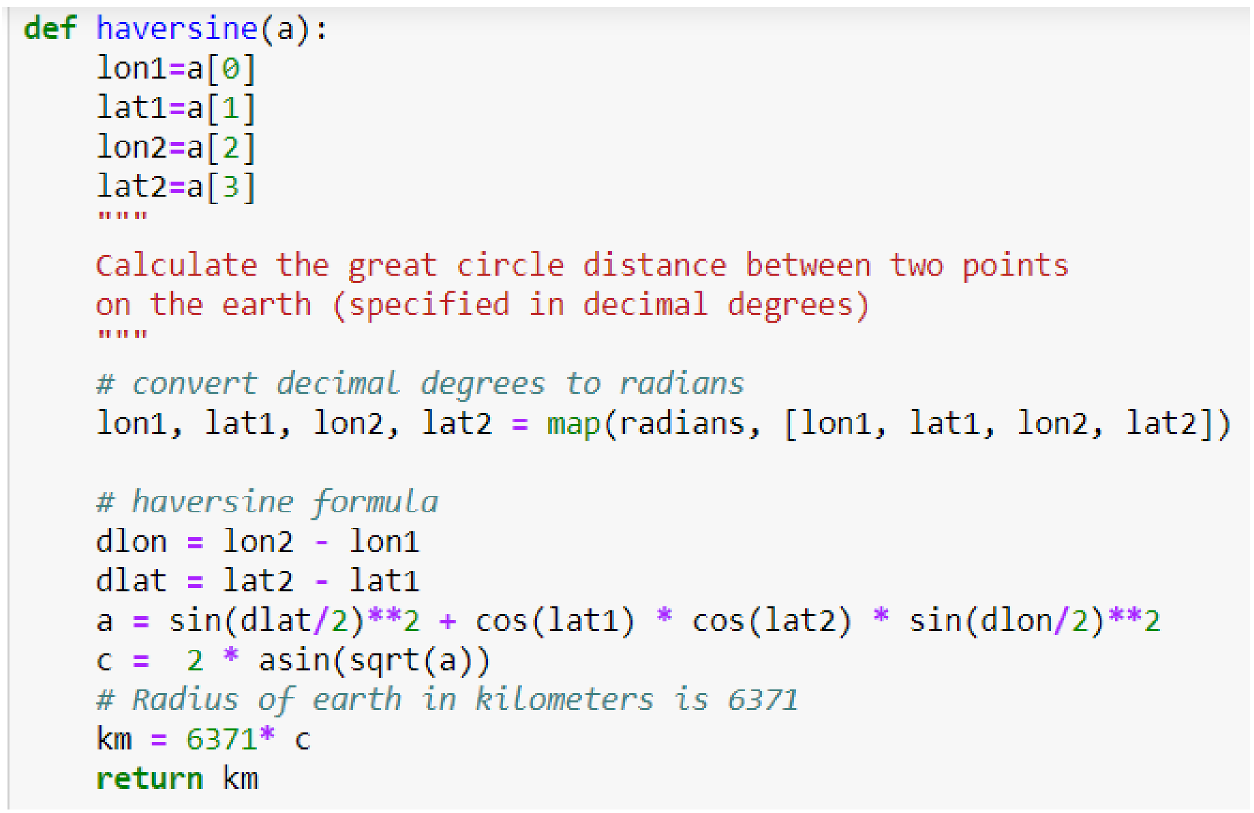
Longitude and Latitude:

The value for Longitude needs to be in a range of -180 to 180 and Latitude values from -90 to 90. Data set has one value out of the above range hence dropped the same.



	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count	Year	Month	Date
5686	3.3	-73.947235	401.083332	-73.951392	40.778927	1.0	2011	7	30

The four columns with latitude and longitude values help us to find the distance using haversine formula. Below is the formula function applied.



```
def haversine(a):
    lon1=a[0]
    lat1=a[1]
    lon2=a[2]
    lat2=a[3]
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c =  2 * asin(sqrt(a))
    # Radius of earth in kilometers is 6371
    km = 6371* c
    return km
```

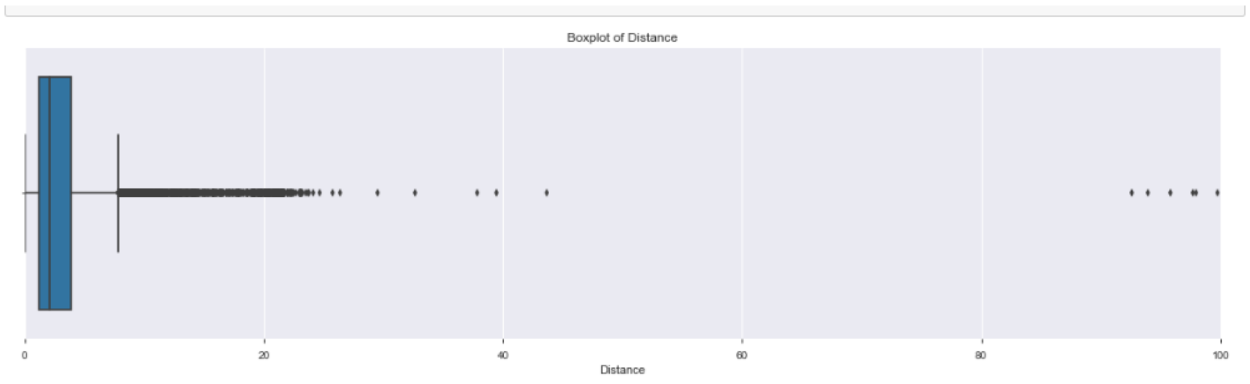
Here we are creating a new column 'Distance'. Hence the above four columns will be dropped.

Outliers:

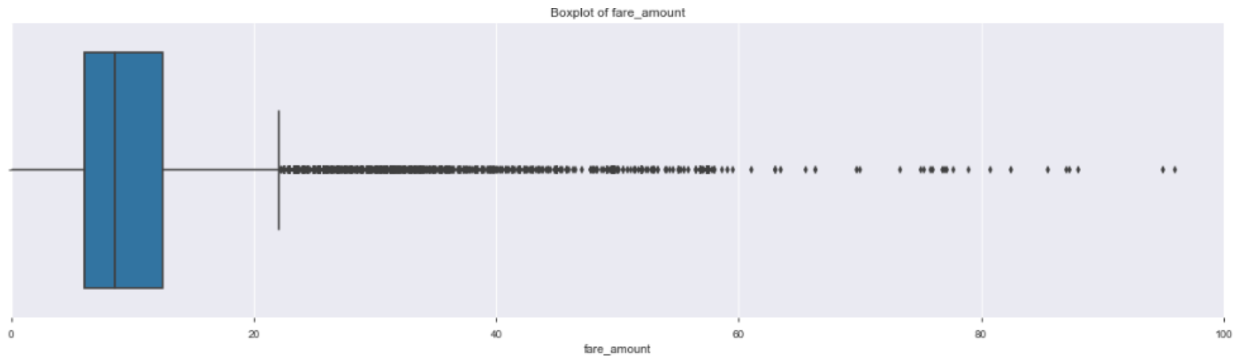
- **fare_amount** has minimum value as -3 and greatest value as 54343 which is not possible. Negative fare amounts and higher values are also not accepatable.
- **passenger_count** has values ranging from 0 to 5345 which is also not acceptable. The value needs to be starting from 1 to 6. As even an SUV can accommodate maximum 6 people..
- **distance** has values from 0 to 8667.54. The largest value is out of city limits hence this also can't be accepted.

Plotting outliers for fare_amount and Distance

Distance



Fare_amount

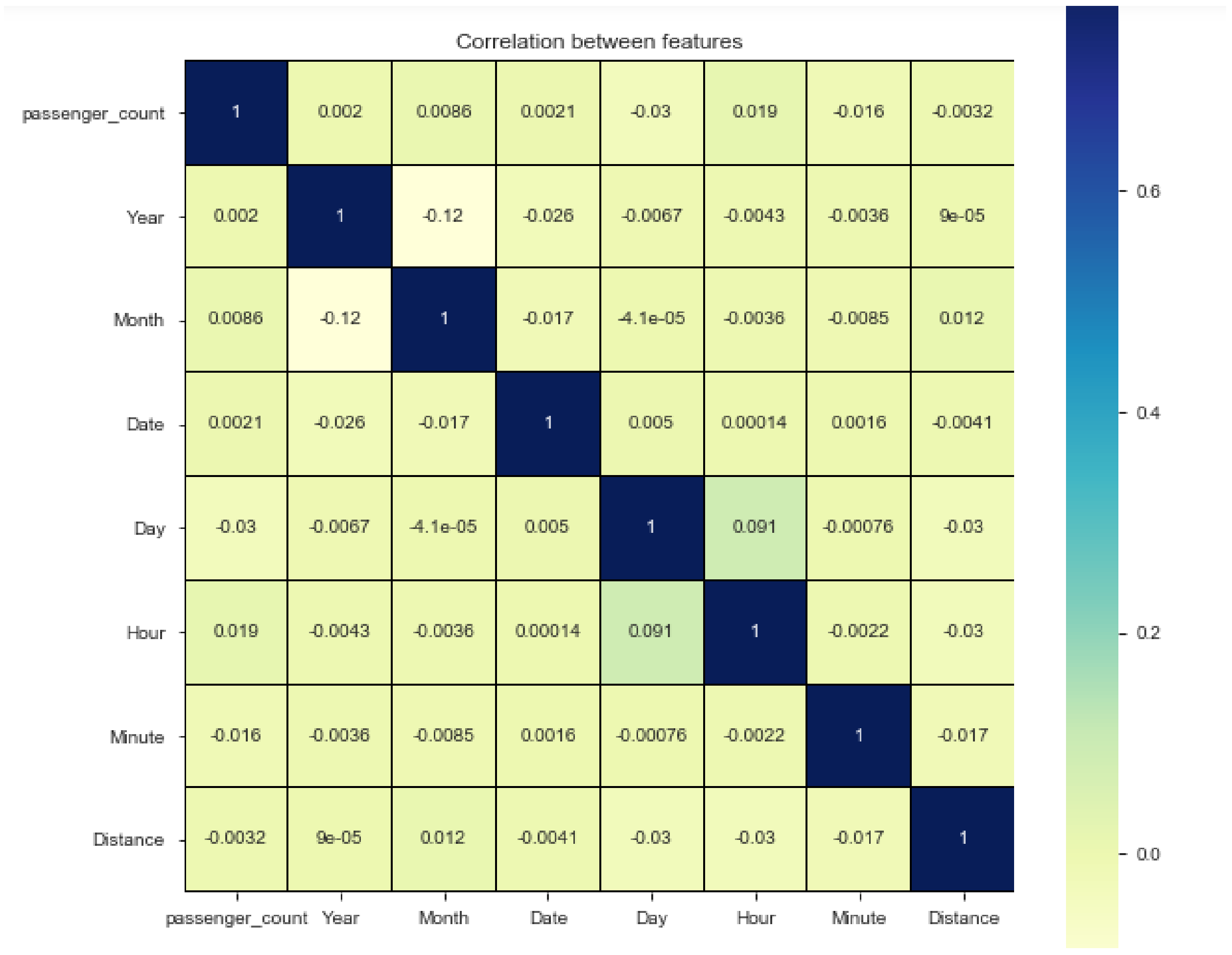


Day:

We have applied a Boolean function on 'Day' column, where 1 value represents weekday and 0 as weekends.

Visualizations:

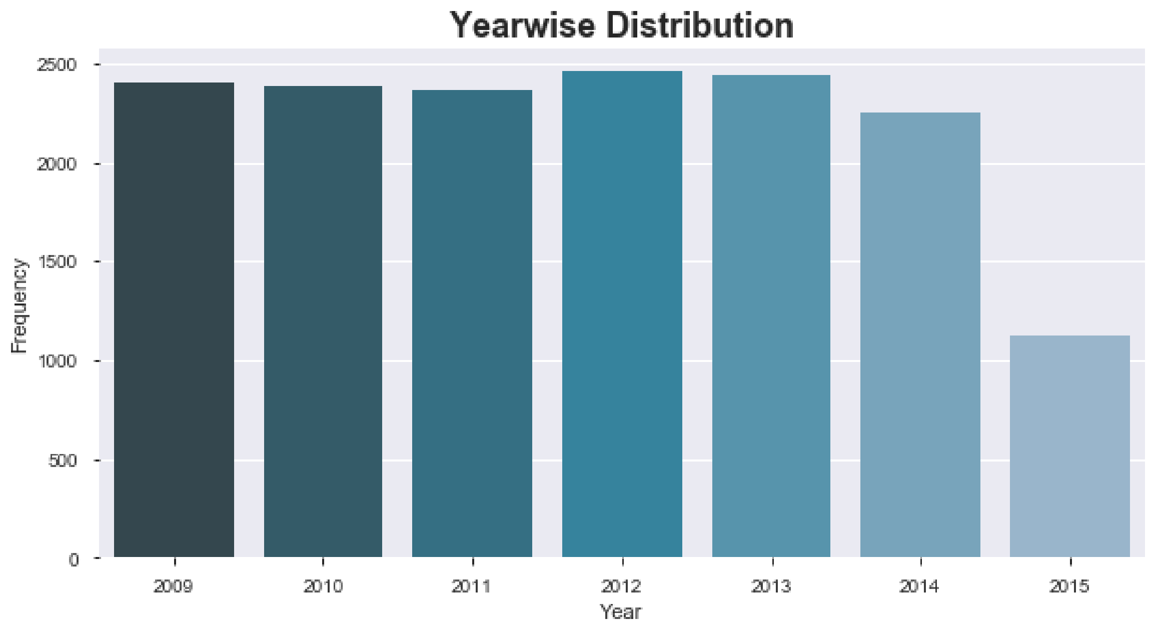
I. Plotting a het map to check the correlations between variables.



	passenger_count	Year	Month	Date	Day	Hour	Minute	Distance
passenger_count	1	0.002	0.0086	0.0021	-0.03	0.019	-0.016	-0.0032
Year	0.002	1	-0.12	-0.026	-0.0067	-0.0043	-0.0036	9e-05
Month	0.0086	-0.12	1	-0.017	-4.1e-05	-0.0036	-0.0085	0.012
Date	0.0021	-0.026	-0.017	1	0.005	0.00014	0.0016	-0.0041
Day	-0.03	-0.0067	-4.1e-05	0.005	1	0.091	-0.00076	-0.03
Hour	0.019	-0.0043	-0.0036	0.00014	0.091	1	-0.0022	-0.03
Minute	-0.016	-0.0036	-0.0085	0.0016	-0.00076	-0.0022	1	-0.017
Distance	-0.0032	9e-05	0.012	-0.0041	-0.03	-0.03	-0.017	1

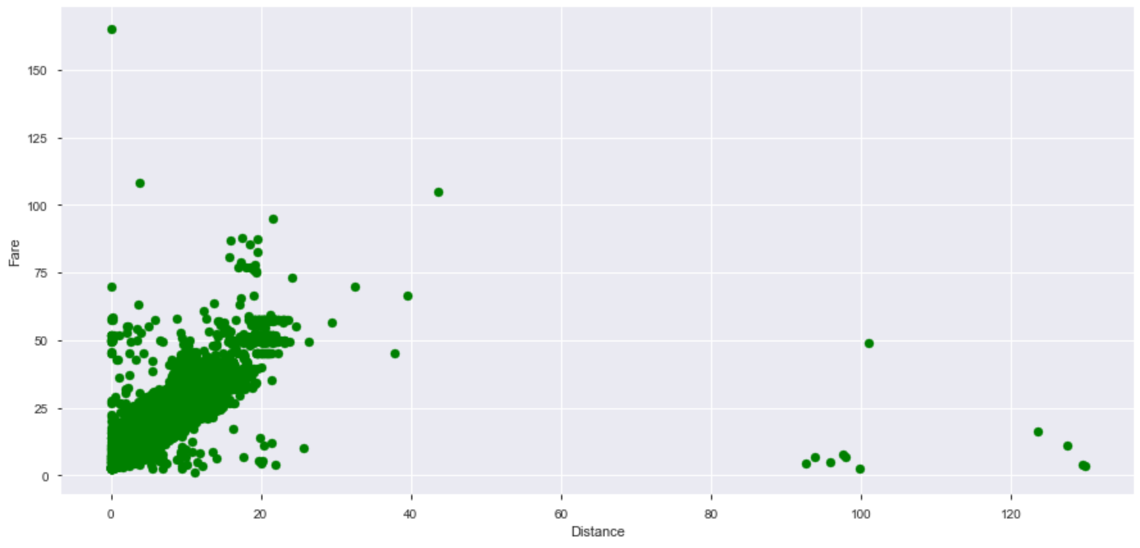
There are no variables with High correlation. So we won't drop any columns from our analysis.

II. Yearwise Data Distribution



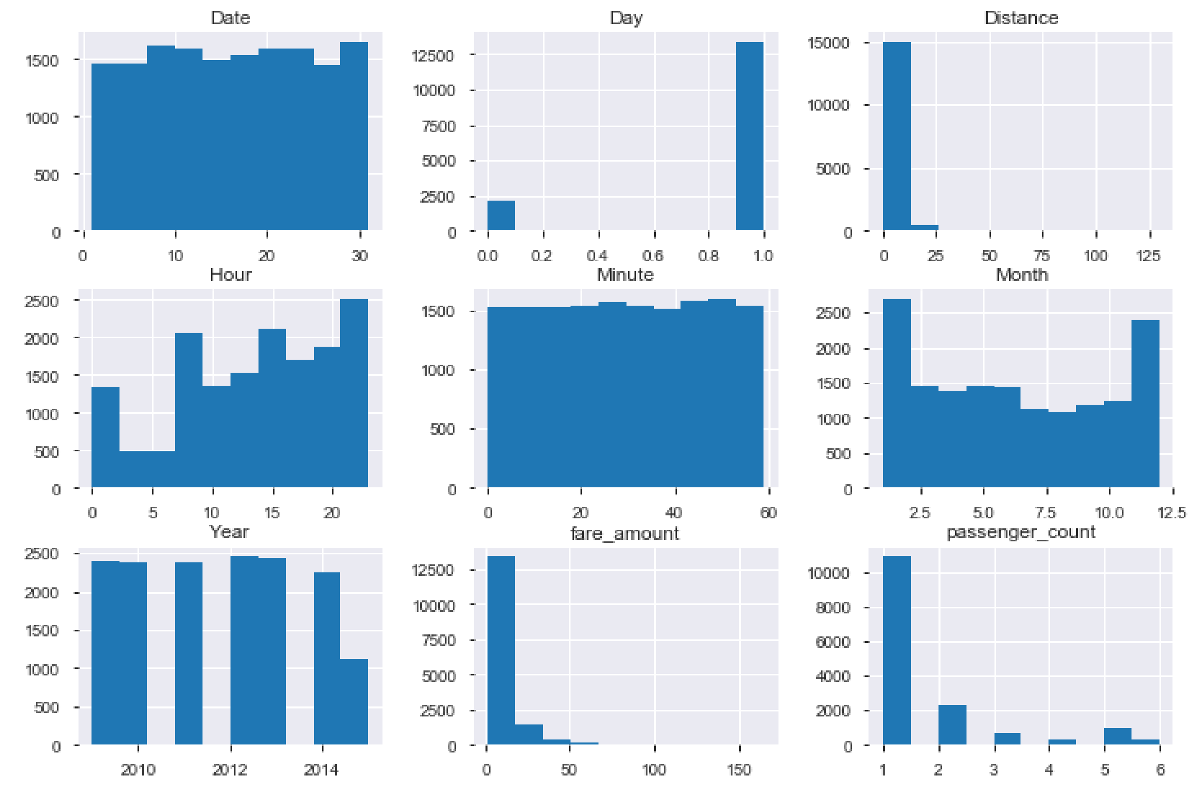
So most of the data belongs to 2009 to 2014 Year

Relationship between Distance and Fare amount



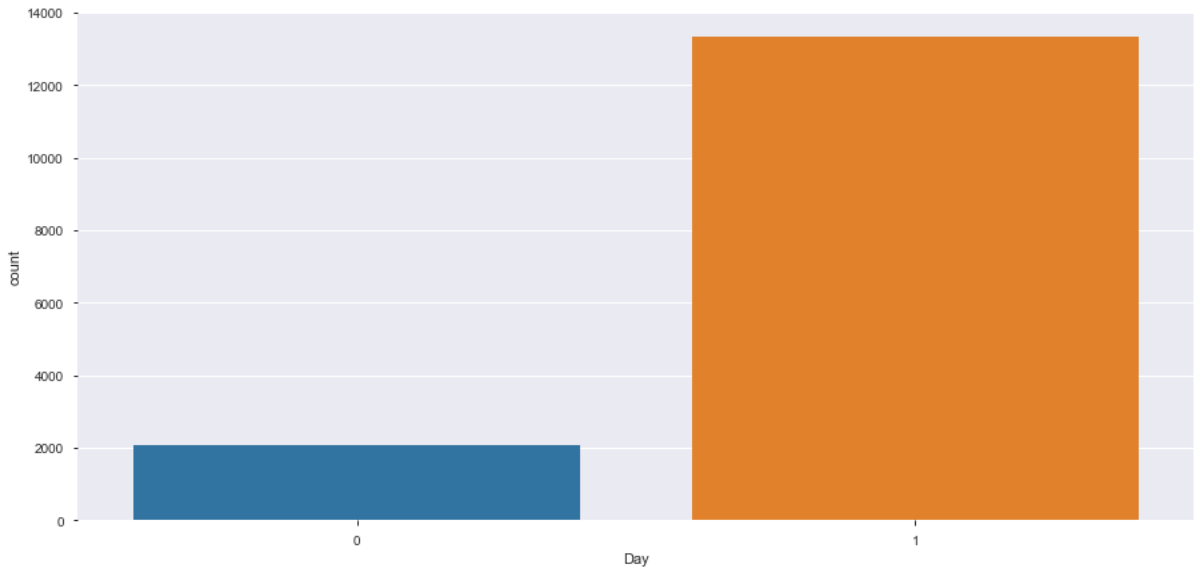
Thus, More the distance, more is the fare

Distribution of data corresponding to all variables:



We can see that no data is normally distributed. We will need to normalize the data.

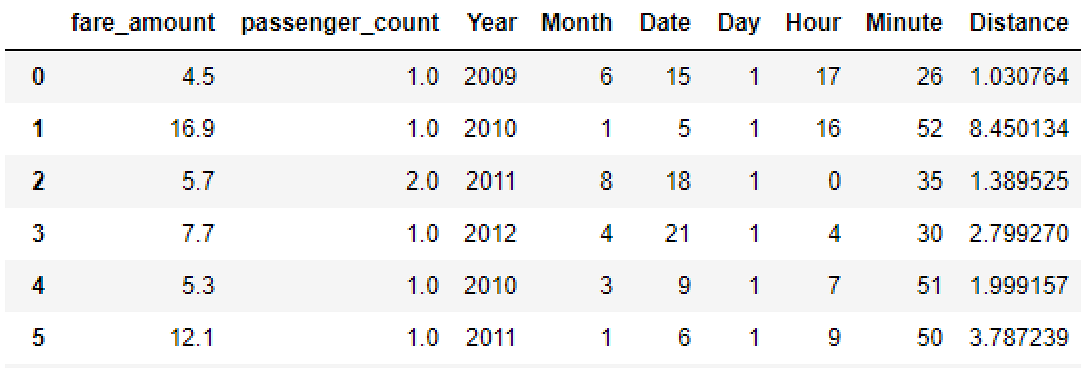
Impact of bookings on Weekdays and Weekends:



Thus, no of bookings made on weekdays are a lot more than weekends.

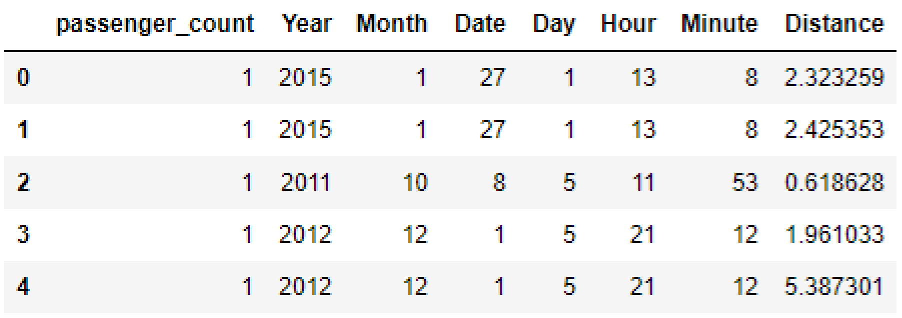
After Data Processing our final train and test datasets looks as below:

Train:



	fare_amount	passenger_count	Year	Month	Date	Day	Hour	Minute	Distance
0	4.5	1.0	2009	6	15	1	17	26	1.030764
1	16.9	1.0	2010	1	5	1	16	52	8.450134
2	5.7	2.0	2011	8	18	1	0	35	1.389525
3	7.7	1.0	2012	4	21	1	4	30	2.799270
4	5.3	1.0	2010	3	9	1	7	51	1.999157
5	12.1	1.0	2011	1	6	1	9	50	3.787239

Test:



	passenger_count	Year	Month	Date	Day	Hour	Minute	Distance
0	1	2015	1	27	1	13	8	2.323259
1	1	2015	1	27	1	13	8	2.425353
2	1	2011	10	8	5	11	53	0.618628
3	1	2012	12	1	5	21	12	1.961033
4	1	2012	12	1	5	21	12	5.387301

Feature Scaling:

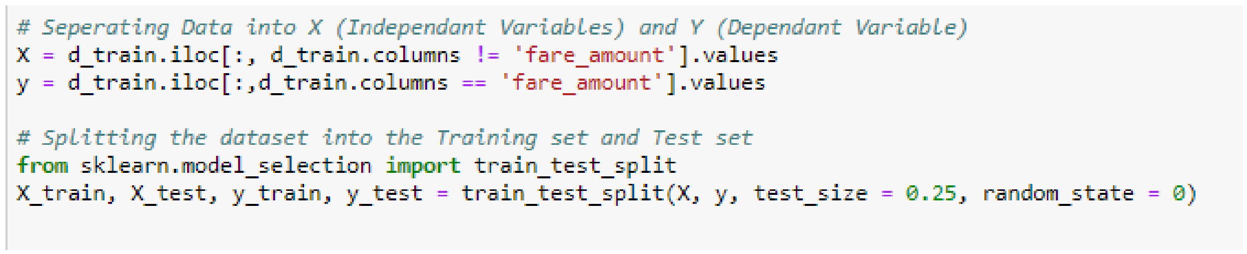
We saw in Visualizations that no data is normally distributed. Hence we will normalize the columns which belongs to numeric data type. Those are fare_amount and Distance.

Model Building:

After a thorough preprocessing, we will use some regression models on our processed data to predict the target variable. Here in this case, our target variable is fare_amount Following are the models which we have built –

I. Linear Regression
II. Decision Tree
III. 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 75% of the data as our train data. Below is the snipped image of the split of train test.



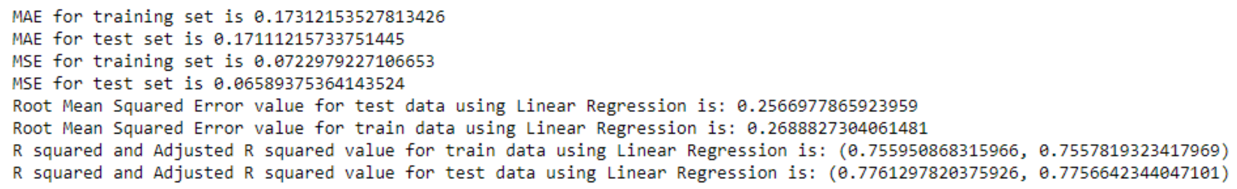
```
# Seperating Data into X (Independant Variables) and Y (Dependant Variable)
X = d_train.iloc[:, d_train.columns != 'fare_amount'].values
y = d_train.iloc[:,d_train.columns == 'fare_amount'].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
```

Linear Regression:

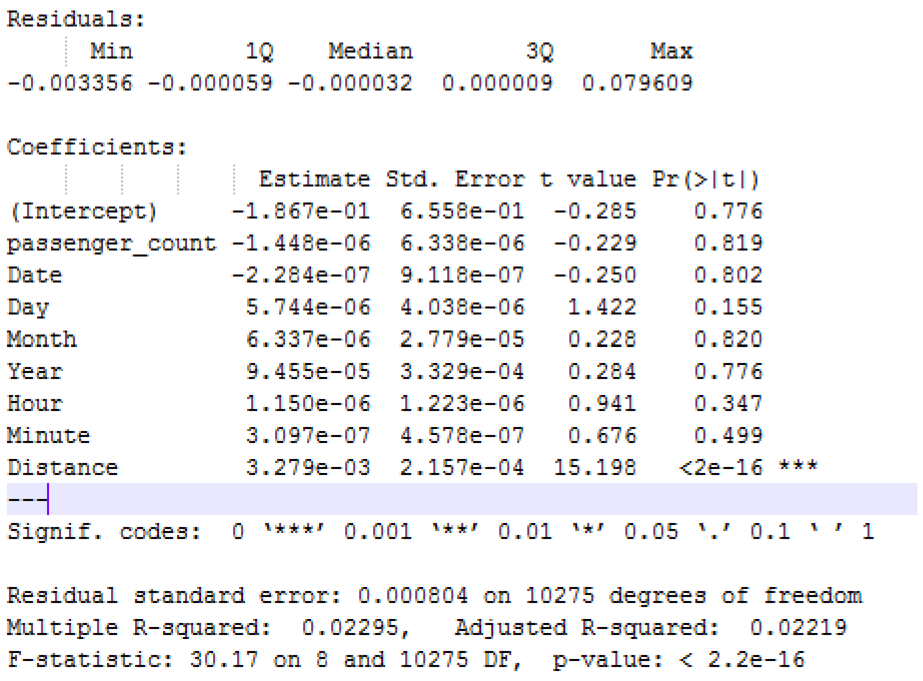
It is used to predict the value of variable *Y* based on one or more input predictor variables *X*. The goal of this method is to establish a linear relationship between the predictor variables and the response variable. Such that, we can use this formula to estimate the value of the response *Y*, when only the predictors (*X- Values*) are known.

In Python:



```
MAE for training set is 0.17312153527813426
MAE for test set is 0.17111215733751445
MSE for training set is 0.0722979227106653
MSE for test set is 0.06589375364143524
Root Mean Squared Error value for test data using Linear Regression is: 0.2566977865923959
Root Mean Squared Error value for train data using Linear Regression is: 0.2688827304061481
R squared and Adjusted R squared value for train data using Linear Regression is: (0.755950868315966, 0.7557819323417969)
R squared and Adjusted R squared value for test data using Linear Regression is: (0.7761297820375926, 0.7756642344047101)
```

In R:



```
Residuals:
      Min        1Q    Median        3Q       Max
-0.003356 -0.000059 -0.000032  0.000009  0.079609

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.867e-01  6.558e-01  -0.285    0.776
passenger_count -1.448e-06  6.338e-06  -0.229    0.819
Date            -2.284e-07  9.118e-07  -0.250    0.802
Day              5.744e-06  4.038e-06   1.422    0.155
Month            6.337e-06  2.779e-05   0.228    0.820
Year             9.455e-05  3.329e-04   0.284    0.776
Hour             1.150e-06  1.223e-06   0.941    0.347
Minute           3.097e-07  4.578e-07   0.676    0.499
Distance         3.279e-03  2.157e-04  15.198   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.000804 on 10275 degrees of freedom
Multiple R-squared:  0.02295,	Adjusted R-squared:  0.02219
F-statistic: 30.17 on 8 and 10275 DF,  p-value: < 2.2e-16
```

Here, F-Statistic explains about the quality of the model. AIC is Akkaine information criterion, if we have multiple models with same accuracy then we need to refer this to choose the best model. The table three values containing Omnibus and JB test are mostly required for timestamp data sets. Here, as we are not using any time values in our project we can ignore this table 3. T-statistic explain how much statistically significant the coefficient is. It is also used to calculate the P –Value. And if P-Value is less then 0.05 we reject null hypothesis and say that the variable is significant. Here, all the variables are less then 0.05 and are significant. The R squared and adjusted R squared values show

how much variance of the output variable is explained by the independent or input variables. Here the adjusted r square value is only 45.01%, which explains that only 45% of the variance of fare_amount is explained by the input variables. This shows that the model is performing very poor. This may be because the relationship between the independent and dependent variable might be nonlinear.

2.Decision Tree:

Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate the target value/dependent variable.

Decision trees are divided into three main parts this are :

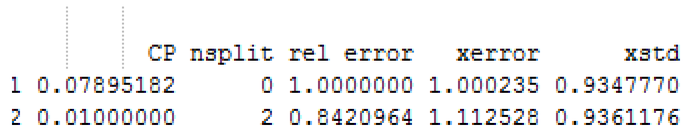
- **Root Node :** performs the first split
- **Terminal Nodes :** that predict the outcome, these are also called leaf nodes
- **Branches :** arrows connecting nodes, showing the flow from root to other leaves.

In Python:

```
MAE for training set is 0.21441583953935595
MAE for test set is 0.21462156741409646
MSE for training set is 0.08689901910496314
MSE for test set is 0.08515520334733259
Root Mean Squared Error value for test data using Decision Tree Regressor
is: 0.29181364489573236
Root Mean Squared Error value for train data using Decision Tree Regressor
is: 0.294786395725724
R squared and Adjusted R squared value for train data using Decision Tree
Regressor is: (0.7066633540546801, 0.7064603002199857)
R squared and Adjusted R squared value for test data using Decision Tree R
egressor is: (0.710690120375646, 0.7100884881851093)
```

`

In R:



```
          CP nsplit rel error   xerror      xstd
1 0.07895182      0 1.0000000 1.000235 0.9347770
2 0.01000000      2 0.8420964 1.112528 0.9361176
```

Random Forest

The next model to be followed in this project is Random forest. It is a process where the machine follows an ensemble learning method for classification and regression that operates by developing a number of decision trees at training time and giving output as the class that is the mode of the classes of all the individual decision trees.

In Python:

```
MAE for training set is 0.062397000684448821
MAE for test set is 0.16728086916242163
MSE for training set is 0.008559349757750113
MSE for test set is 0.06196120805313761
Root Mean Squared Error value for test data using Random Forest Regressor
is: 0.2489200836677057
Root Mean Squared Error value for train data using Random Forest Regressor
is: 0.09251675393003211
R squared and Adjusted R squared value for train data using Random Forest
Regressor is: (0.9711070277286029, 0.9710870274016866)
R squared and Adjusted R squared value for test data using Random Forest R
egressor is: (0.7894903782906144, 0.789052614585474)
```

Model Evaluation:

We will evaluate performance on validation dataset which was generated using Sampling.

We will deal with specific error metrics like –

Regression metrics for our Models:

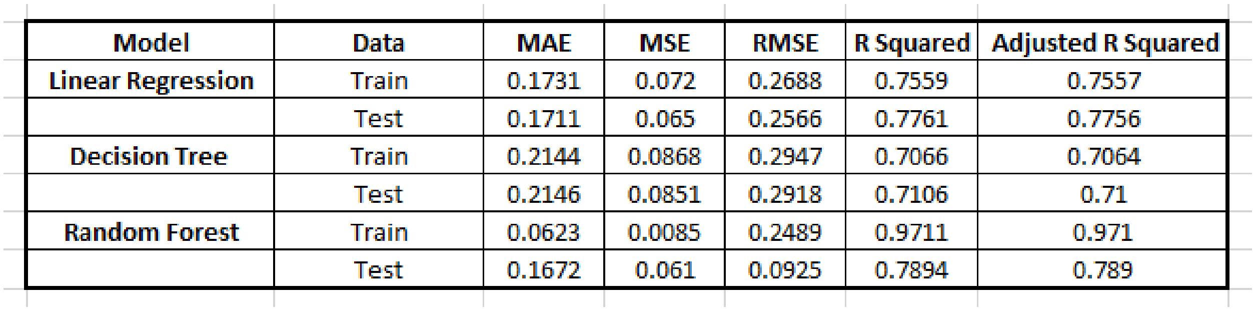
- r square

- Adjusted r square

- MAE(Mean Absolute Error)

- MSE(Mean square Error)

- RMSE(Root Mean Square Error)



Model	Data	MAE	MSE	RMSE	R Squared	Adjusted R Squared
Linear Regression	Train	0.1731	0.072	0.2688	0.7559	0.7557
	Test	0.1711	0.065	0.2566	0.7761	0.7756
Decision Tree	Train	0.2144	0.0868	0.2947	0.7066	0.7064
	Test	0.2146	0.0851	0.2918	0.7106	0.71
Random Forest	Train	0.0623	0.0085	0.2489	0.9711	0.971
	Test	0.1672	0.061	0.0925	0.7894	0.789

Hyperparameter Tuning:

1. Random Forest Search Cross Validation:

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.

```
Random Search CV Random Forest Regressor Model Performance:
R-squared = 0.8.
RMSE =  0.24309226351587415
```

2. Random Grid Search Cross Validation

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.

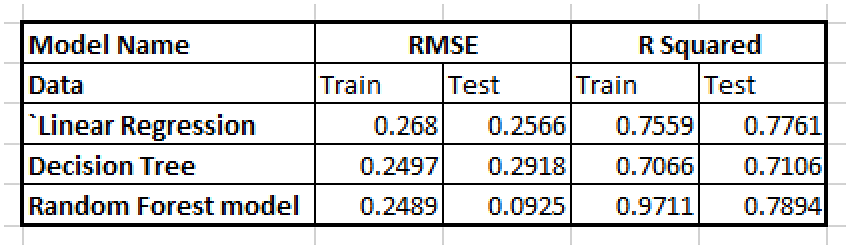
```
Grid Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'max_depth': 7, 'n_estimators': 19}
R-squared = 0.8.
RMSE =  0.2421340600470303
```

Conclusion:

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.

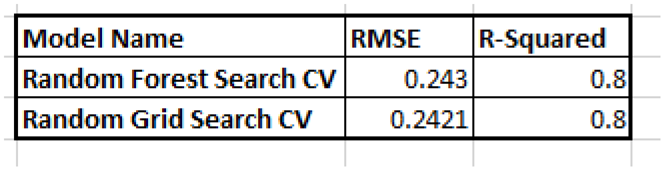
II. **R Squared(R^2):** is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. In other words, we can say it explains as to how much of the variance of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

Below are the values for different Algorithms we built:



Model Name	RMSE		R Squared	
Data	Train	Test	Train	Test
`Linear Regression	0.268	0.2566	0.7559	0.7761
Decision Tree	0.2497	0.2918	0.7066	0.7106
Random Forest model	0.2489	0.0925	0.9711	0.7894

After applying the hyperparameter tuning below are the scores for new modles:



Model Name	RMSE	R-Squared
Random Forest Search CV	0.243	0.8
Random Grid Search CV	0.2421	0.8

Model Selection

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,

• Random Forest performed comparatively well while comparing their RMSE and R-Squared value.

• Hence I applied hyperparameter tuning using Random Forest CV and Grid Search CV.

• After applying tunings Random forest model shows a good change in the values.

• 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.