Flight Ticket Price Prediction

Saurabh Kumar B20CS062

Abstract – This paper reports my observations and analysis with building a Flight Ticket Price Prediction regression model. The dataset consists of prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities. I have used various regression models, tuned hyperparameters and compared the result in this report.

I.INTRODUCTION

Flight ticket prices can be something hard to guess, today we might see a price, and when we check out the price of the same flight tomorrow, it will be a different story. This project predicts the price of flight tickets. A thorough study of the data will aid in the discovery of valuable insights that will be of enormous value to passengers.

Datasets

The file Dataset.xlsx is used as the dataset with 80% as training data and the rest 20% as testing data. The train dataset contains 10683 rows where each row represents a data consisting of following features :-

1. Airline: The name of the airline
2. Date_of_Journey: The date of the journey
3. Source: The source from which the service begins
4. Destination: The destination where the service ends
5. Dep_Time: The time when the journey starts from the source.
6. Arrival_Time: Time of arrival at the destination.
7. Route: The route of the flight
8. Duration: Total duration of the flight.
9. Total_Stops: Total stops between the source and destination.
10. Additional_Info: Additional information about the flight
11. Price: The target feature, the price of the ticket

II. METHODOLOGY

Overview

There are various regression algorithms present out of which I will implement the following in my pipeline:-

- Decision Tree Regressor
- Random Forest Regressor
- Gradient Boosting Regressor
- LightGBM Regressor
- KNeighborsRegressor
- Logistic Regression

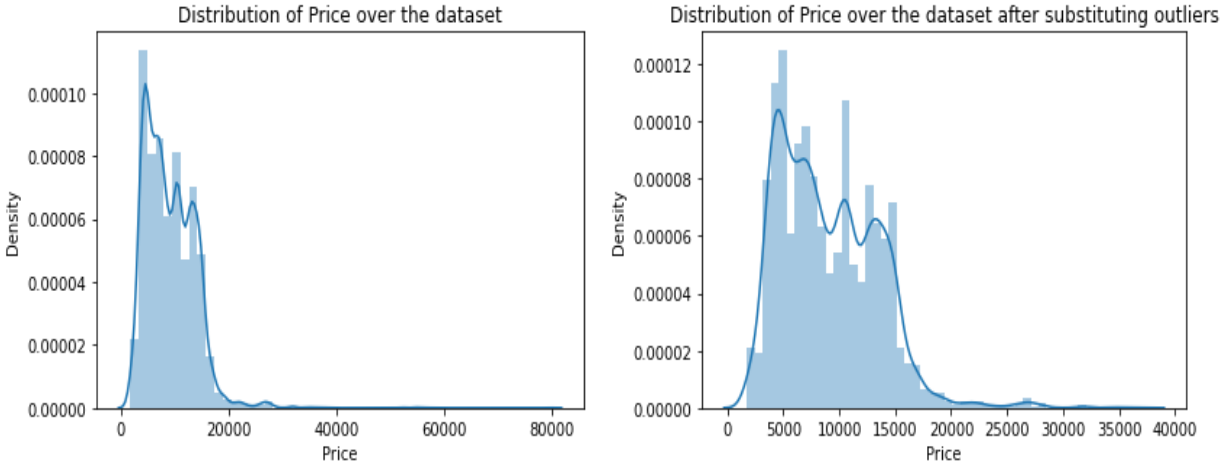
I will also perform preprocessing of data which involves data cleaning, encoding categorical data, removing outliers and hyperparameters tuning.

Exploring Dataset and Pre-processing

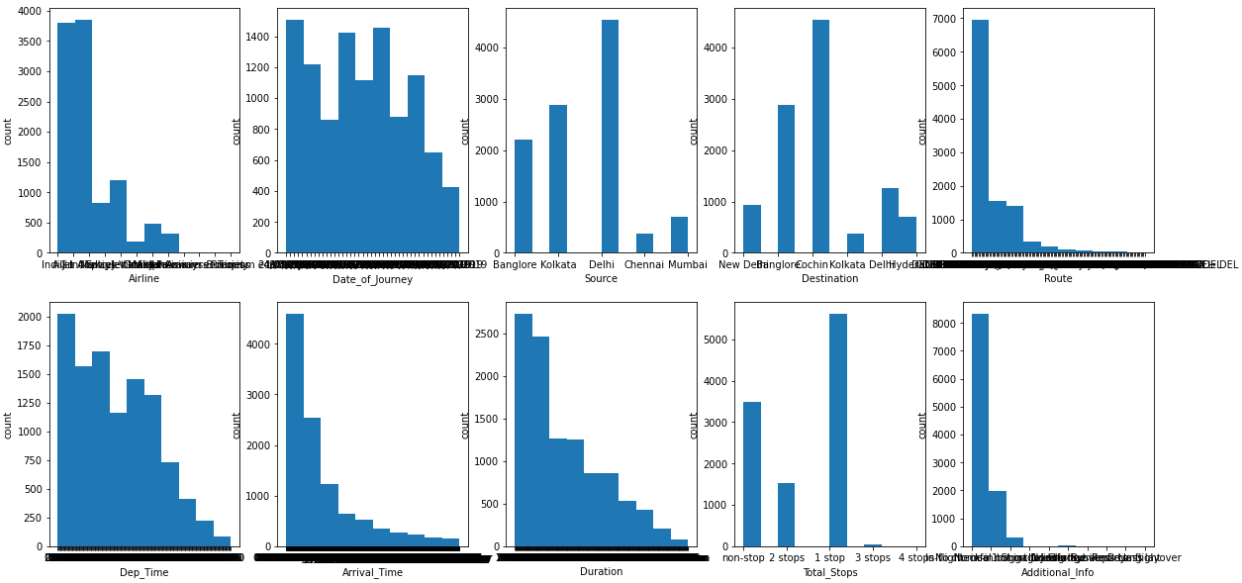
On checking for NaN values in the dataset, two features each having one NaN value are found which I

remove from the dataset. The data type of each column except the Price is Object. Price column is of integer datatype. The data preprocessing steps involve extracting the day and month of journey from *Date_of_Journey* column, extracting the hour and minute of departure and arrival of the flight from the *Dep_Time* and *Arrival_Time* respectively and also extracting the hour and minutes of flight from *Duration* column. The next preprocessing step involves encoding categorical columns either using *LabelEncoder* or *OneHotEncoder* based on the number of unique entities present in each column. The final preprocessing step involves tracing the outliers present in the dataset. Outliers are substituted with the median value of the *Price* column.

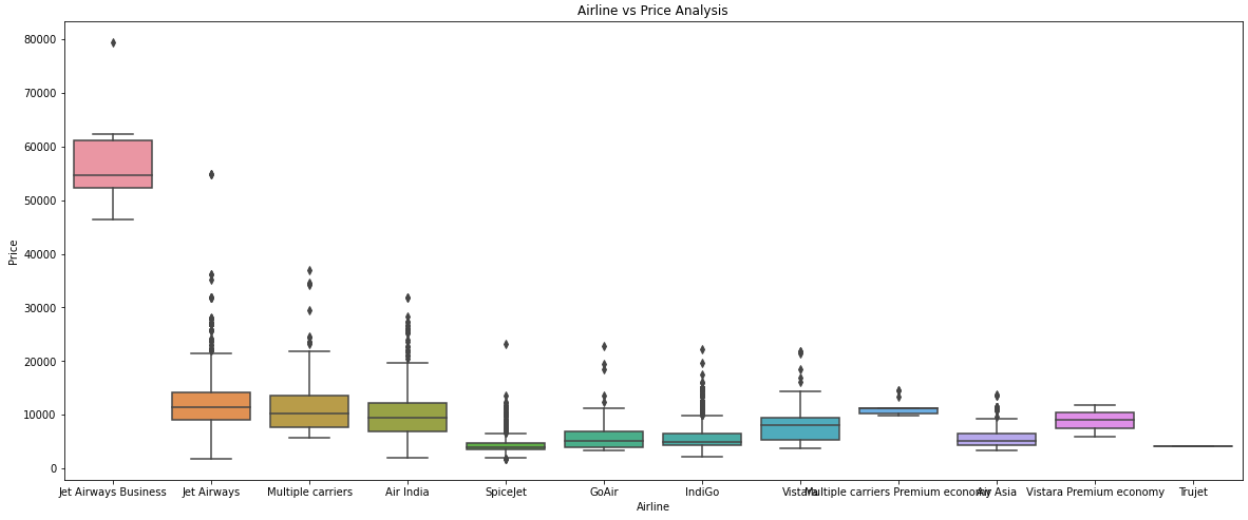
Data Visualization

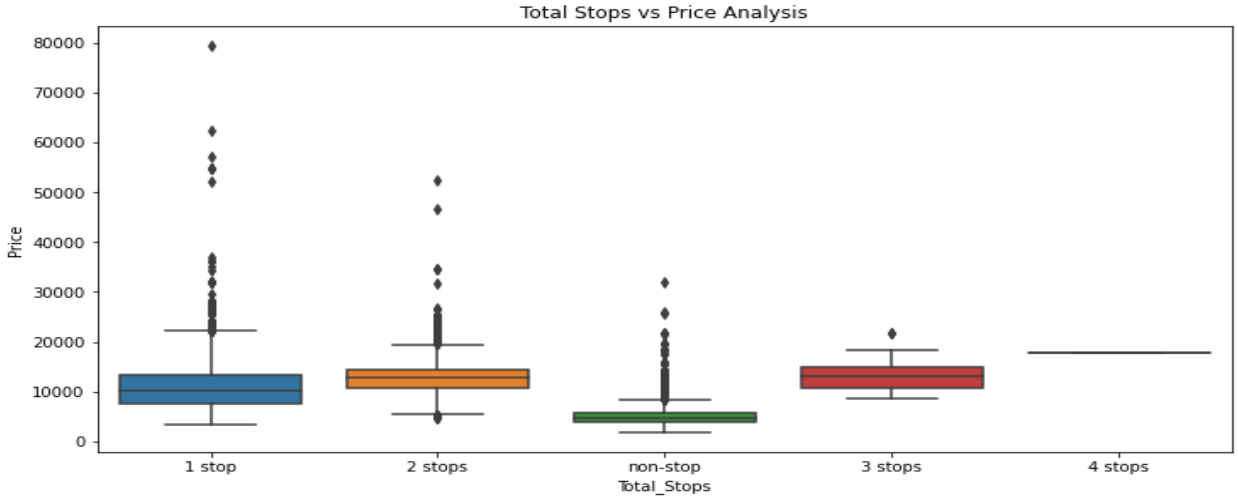


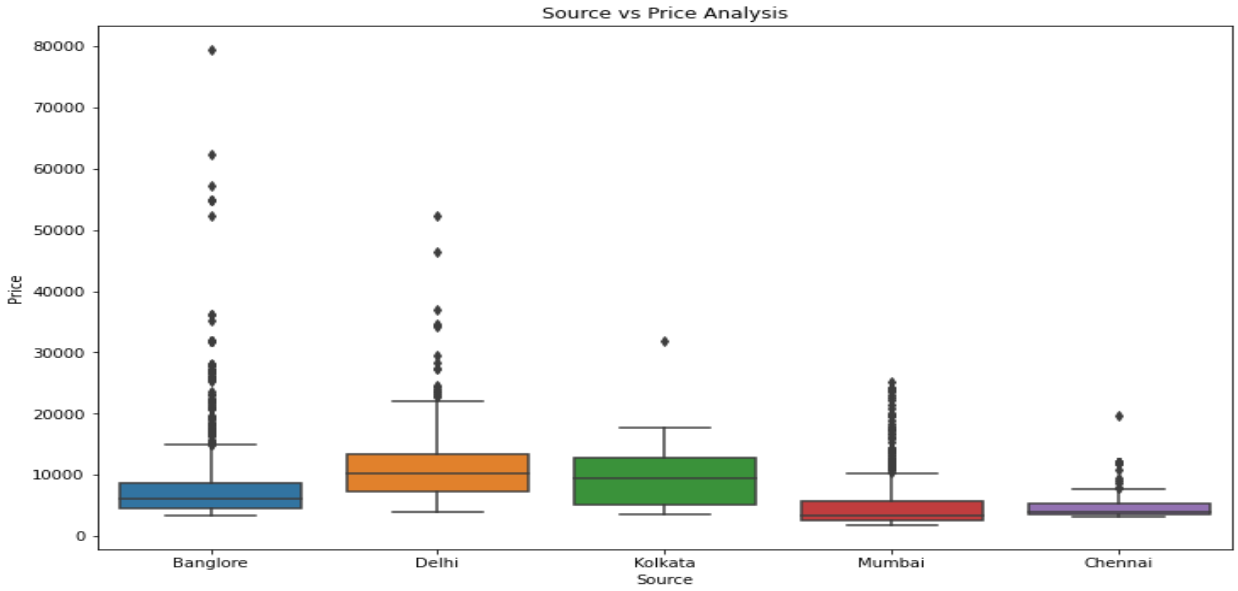
Histogram plot showing distribution of various features across the dataset

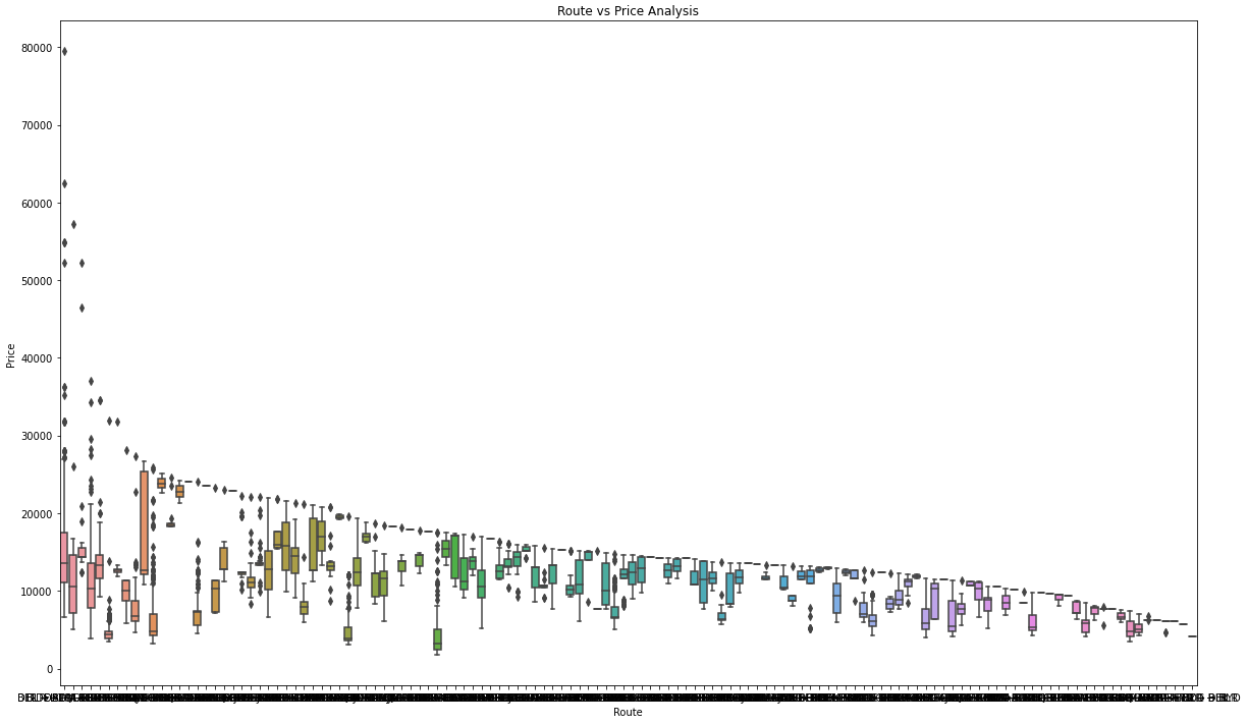


Features vs Price Analysis using boxplot from seaborn library









Implementation of Regression algorithms

I have implemented a pipeline *training_pipeline* in which the regressor model object is given as input and the output is the trained model with various evaluation metrics calculated along with the trained model on both training dataset and testing dataset.

- *Decision Tree regressor* : Decision tree learning is one of the predictive modeling approaches used in statistics, data mining and machine learning. Algorithms for constructing decision trees usually work top-down, by choosing a variable at each step that best splits the set of items.

```
DecisionTreeRegressor :-

Evaluation metrics on Training dataset:-
r2_score :-  0.994996702653313
mean squared error (MSE):- 94759.3166569144
root mean squared error (RMSE):- 307.8300125993474
mean absolute error (MAE):- 42.527052857421495

Evaluation metrics on Testing dataset:-
r2_score :-  0.8803183332160341
mean squared error (MSE):- 2451053.1451801592
root mean squared error (RMSE):- 1565.5839629927739
mean absolute error (MAE):- 636.6300889096865
DecisionTreeRegressor(random_state=42)
```

- *Random Forest Regressor* : Random Forest regressors use boosting ensemble methods to train upon various decision trees and produce aggregated results. It is one of the most used machine learning algorithms.

```
RandomForestRegressor :-

Evaluation metrics on Training dataset:-
r2_score :-  0.9836959792601387
mean squared error (MSE):- 308787.9366379096
root mean squared error (RMSE):- 555.6869052244344
mean absolute error (MAE):- 250.65720772102875

Evaluation metrics on Testing dataset:-
r2_score :-  0.9175798978695857
mean squared error (MSE):- 1687944.8288222419
root mean squared error (RMSE):- 1299.2093090885094
mean absolute error (MAE):- 602.6279896550571
```

- *Gradient Boosting Regressor* : Gradient Boosting builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.

```
GradientBoostingRegressor :-

Evaluation metrics on Training dataset:-
r2_score :-  0.920083970290578
mean squared error (MSE):- 1513559.5269413493
root mean squared error (RMSE):- 1230.2680711704052
mean absolute error (MAE):- 811.0321147844285

Evaluation metrics on Testing dataset:-
r2_score :-  0.8935453472216222
mean squared error (MSE):- 2180166.9255032614
root mean squared error (RMSE):- 1476.5388330495277
mean absolute error (MAE):- 923.170597250966
```

- *KNeighborsRegressor* : Regression based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated with the nearest neighbors in the training set.

```
KNeighborsRegressor :-

Evaluation metrics on Training dataset:-
r2_score :-  0.8012210038916556
mean squared error (MSE):- 3764749.630450556
root mean squared error (RMSE):- 1940.296273884624
mean absolute error (MAE):- 1243.8616032767698

Evaluation metrics on Testing dataset:-
r2_score :-  0.6992093356202922
mean squared error (MSE):- 6160123.967019186
root mean squared error (RMSE):- 2481.959702940236
mean absolute error (MAE):- 1619.439026672906
```

- *LightGBM Regressor* : LightGBM is a gradient boosting framework based on decision trees to increase the efficiency of the model and reduce memory usage. It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfills the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks.

```
LightGBMRegressor :-

Evaluation metrics on Training dataset:-
r2_score :-  0.930327237380417
mean squared error (MSE):- 1319558.4667384967
root mean squared error (RMSE):- 1148.720360548422
mean absolute error (MAE):- 731.92532392017

Evaluation metrics on Testing dataset:-
r2_score :-  0.908122409646397
mean squared error (MSE):- 1881632.0231758596
root mean squared error (RMSE):- 1371.7259285935581
mean absolute error (MAE):- 819.0864937711192
```

- *Logistic Regression* : Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

```
LogisticRegression :-

Evaluation metrics on Training dataset:-
r2_score :-  0.3804773683719499
mean squared error (MSE):- 11733370.44727911
root mean squared error (RMSE):- 3425.4007717753425
mean absolute error (MAE):- 2027.2741954359274

Evaluation metrics on Testing dataset:-
r2_score :-  0.3780268331884108
mean squared error (MSE):- 12737868.11043519
root mean squared error (RMSE):- 3569.0150056332336
mean absolute error (MAE):- 2209.4047730463267
```

Hyperparameter Tuning

I will tune the hyperparameter such as *n_estimators*, *max_depth, max_features, min_samples_leaf, min_samples_split* for *RandomForestRegressor* and *GradientBoostingRegressor* using *RandomizedSearchCV* and report the outcomes of both operations. I will vary n_estimators from 100 to 1300 with a interval of 100, max_depth is varied between 5 and 16, two values for max_features (auto, sqrt) are selected, min_samples_leaf is varied between 1 to 8 with an interval of 2 and at last min_samples_split is varied between 2 to 100 in multiples of 5.

RandomizedSearchCV is very useful when we have many parameters to try and the training time is very long. The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

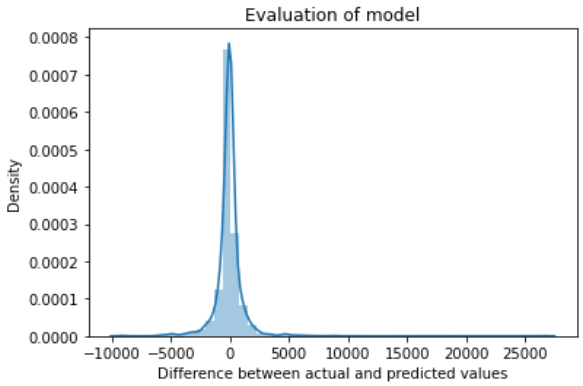
- *RandomForestRegressor*

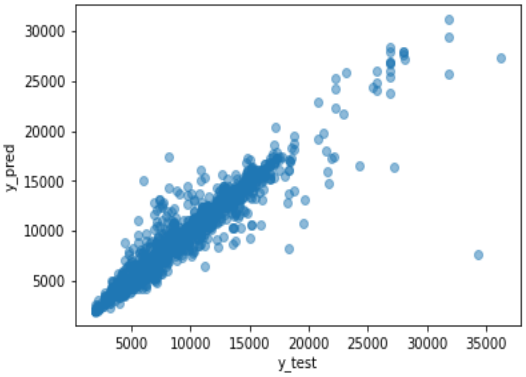
  The best set of parameters using RandomForestRegressor comes out to be following:

```
{'max_depth': 10,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 900}
```

  The evaluation metric scores using optimal random forest regressor are:-

```
Evaluation metrics on Testing dataset:-
r2_score :-  0.8925175424175865
mean squared error (MSE):- 2201216.1326647075
root mean squared error (RMSE):- 1483.6495990174726
mean absolute error (MAE):- 826.6196036781125
```





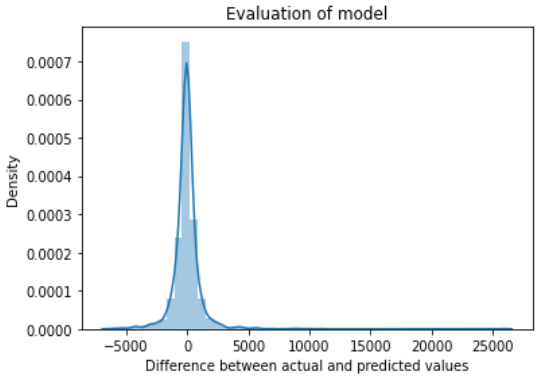
- *Gradient Boosting Regression*

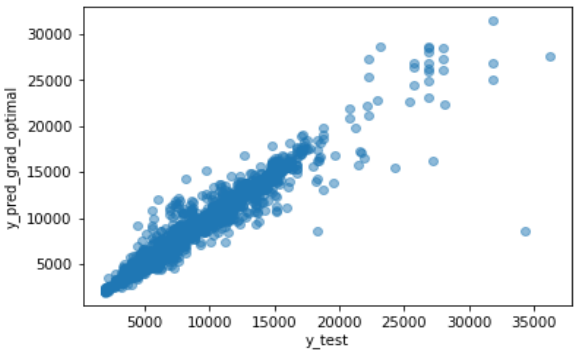
  The best set of parameters using GradientBoostingRegressor comes out to be following:

```
{'max_depth': 11,
 'max_features': 'auto',
 'min_samples_leaf': 5,
 'min_samples_split': 100,
 'n_estimators': 500}
```

  The evaluation metric scores using optimal gradient boosting regressor are:-

```
Evaluation metrics on Testing dataset:-
r2_score :-  0.9263119035413069
mean squared error (MSE):- 1509115.3510874733
root mean squared error (RMSE):- 1228.4605614701163
mean absolute error (MAE):- 636.4680577782116
```





Results and Analysis

Table 1. Evaluation Metrics over primitive models

	r2-Score	RMSE
Decision Tree Regressor	0.8832	1565.5839
Random Forest Regressor	0.9101	1299.2093
GradientBoostingRegressor	0.8935	1476.5388
KNeighborsRegressor	0.6992	2481.9597
LightGBM Regressor	0.9081	1371.7259
Logistic Regression	0.3780	3569.0150

Table 2. Evaluation Metrics over hyperparameter tuned models

	r2-Score	RMSE
Random Forest Regressor	0.8925	1483.6495
GradientBoostingRegressor	0.9263	1228.4606

Conclusions

The table shows that all the tree-based models like *DecisionTreeRegressor, RandomForestRegressor, GradientBoostingRegressor and LightGBMRegressor* performed equally well. The ensemble learners have performed better in terms of the RMSE value. The hyperparameter tuning improved the results of *Gradient Boosting Regressor.* The r2-score improved from 0.8935 to 0.9263.