Project Report On

Flight Price Prediction

Submitted By: Sanjog Dhanvijay

INTRODUCTION

Business Problem Framing

The Airline industry is considered as one of the most sophisticated industries in using complex pricing strategies. Now-a-days flight prices are quite unpredictable. The ticket prices change frequently. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible.

Using technology, it is possible to reduce the uncertainty of flight prices. So here we will be predicting the flight prices using efficient machine learning techniques.

When booking a flight, travelers need to be confident that they're getting a good deal. The Flight Price Analysis API uses an Artificial Intelligence algorithm trained on Amadeus historical flight booking data to show how current flight prices compare to historical fares. More precisely, it shows how a current flight price sits on a distribution of historical airfare prices. As retrieving price metrics through aggregation techniques and business intelligence tools alone could lead to incorrect conclusions – for example, in cases where there are insufficient data points to compute specific price statistics – we used machine learning to forecast prices. This provides an efficient way to interpolate missing data and predict coherent flight prices.

Conceptual Background of the Domain Problem

Flight prices are unpredictable. It's more than likely that we spent hours on the internet researching flight deals, trying to figure out an airfare pricing system that seems completely random every day. Flight price appears to fluctuate without reason and longer flights aren't always more expensive than shorter ones.

The question is how to know the proper Flight price, for that I have built a Machine learning model which can predict the Flight price. Using various features like Airline, Source, Destination, Arrival time, Departure time, Stops, Traveling date and the Price for the same trip.

Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

So using all this previously known information and analyzing the data, I have achieved a good model that has 81% accuracy. So, let's understand what all the steps we did to reach this good accuracy.

ANALYTICAL PROBLEM FRAMING

Mathematical/Analytical Modeling of the Problem

As a first step, I have scrapped the required data from makemytrip.com website. I have fetched data for different sources and destinations and saved it to csv format.
In this problem We have Price as my target column and it was a continuous column. Soclearly it is a regression problem and I have to use all regression algorithms while building themodel.

- There were no null values in the dataset.
- Since we have scrapped the data from makemytrip.com website the raw data was not in the right format, so I had to use feature engineering to extract the required feature format.
- To get better insight on the features I have used plotting like distribution plot, bar plot, strip plot and count plot. With these plotting I was able to understand the relation between the features in a better manner.
- I did not find any skewness or outliers in the dataset.
- I have used all the regression algorithms while building models, then tuned the best model and saved the best model.
- At last I have predicted the Price using the saved model.

Data Sources & their formats

The data was collected from makemytrip.com website in csv format. The data was scraped using selenium. After scrapping required features the dataset is saved as a csv file.

Also, my dataset had 3631 rows and 9 columns including target. In this particular dataset I have an object type of data which has been changed as per our analysis about the dataset. The information about features is as follows:

- Airline: The name of the airline.
- Journey_date: The date of the journey
- From: The source from which the service begins.
- To: The destination where the service ends.
- Route: The route taken by the flight to reach the destination.
- DTime: The time when the journey starts from the source.
- ATime: Time of arrival at the destination.
- Stops: Total number of stops between the source and destination.
- Price: The price of the ticket

Data preprocessing done

- As a first step I have scrapped the required data using selenium from makemytrip.com website. And saved the data frame in csv format.
- And I have imported required libraries and I have imported the dataset which was in csv format.
- Then I did all the statistical analysis like checking shape, nunique, value counts, info etc.
- I have also dropped the Unnamed:0 & Unnamed:0.1 column as I found it was the index column of csv file.
- Next as a part of feature engineering I converted the data types of datetime columns and Price columns.
- After that I saw many similar values which could be grouped together under a common name so I replaced all those values under a single heading.
- While checking for null values I found no null values in the dataset.
- I have extracted useful information from the raw dataset. Thinking that this data will help us more than raw data.
- Then, I check the statistical description of our dataset.
- Lastly, I checked for empty values present in our target column, and found no empty values.

Data Inputs- Logic- Output Relationships

The dataset consists of a target and other features. The features are independent and the target is dependent, as the values of our independent variables change, so does our target variable change.
To analyze the relation between features and target I have done EDA where I analyzed the relation using many plots like bar plot, count plot, pair plot, strip plot, dist plot etc.
I have checked the correlation between the target and features using heat map and bar plot, where I got the positive and negative correlation between the label and features.
Since I had numerical columns I have plotted dist plots to see the distribution of skewness in each column data.I have used a bar plot for each pair of categorical features that shows the relation between target and independent features. I have used strip plots to see the relation between numerical columns and target columns.I can notice there is a good relationship between maximum columns and target.

Important features that affect Price positively and negatively are as follows:

Features having high Positive correlation with target: From, To.

Features having high Negative correlation with target: Stops, Airline.

Hardware and Software Requirements & Tools used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- **Processor: core i5 or above**
- **RAM: 8 GB or above**
- **ROM/SSD: 250 GB or above**

Software required:

- **Anaconda - language used Python 3**

Libraries Used:

- **import numpy as np**
- **import pandas as pd**
- **import seaborn as sns**
- **import matplotlib.pyplot as plt**
- **from sklearn.preprocessing import LabelEncoder**
- **from sklearn.preprocessing import StandardScaler**
- **from statsmodels.stats.outliers_influence import variance_inflation_factor**
- **from sklearn.linear_model.LinearRegression**
- **from sklearn.tree import DecisionTreeRegressor**
- **from sklearn.neighbors.KNeighborsRegressor**
- **from sklearn.svm.SVR**
- **from sklearn.ensemble import RandomForestRegressor**
- **from xgboost import XGBRegressor**
- **from sklearn.ensemble import GradientBoostingRegressor**
- **from sklearn.ensemble import ExtraTreesRegressor**
- **from sklearn.model_selection import cross_val_score**
- **from sklearn.model_selection import GridSearchCV**

DATA ANALYSIS & VISUALIZATION

Identification of possible problem-solving approaches (methods)

- **Since the data collected was not in the right format, we had to clean it and bring it to theproper format for our analysis.**
- **There were no outliers and skewness in the dataset.**
- **We have dropped all the unnecessary columns in the dataset according to our understanding.**
- **Use of Pearson's correlation coefficient to check the correlation between dependent and independent features.**
- **Next, we encoded the various categorical columns using Label Encoder.**
- **Also I have used Standardization to scale the data.**
- **After scaling we have to check multicollinearity using VIF.**
- **Then followed by model building with all Regression algorithms.**

Since Price was my target and it was a continuous column with improper format which had to be changed to continuous float data type column, this particular problem was a Regression problem. And I have used all Regression algorithms to build my model.

By looking into the r2 score and error values I found RandomForestRegressor as a best model with highest r2_score and least error values.

Below is the list of Regression algorithms I have used in my project:

- **LinearRegression**
- **SVR**
- **KNearestNeighborsRegressor**
- **RandomForestRegressor**
- **XGBRegressor**
- **ExtraTreesRegressor**
- **GradientBoostingRegressor**
- **DecisionTreeRegressor**

Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

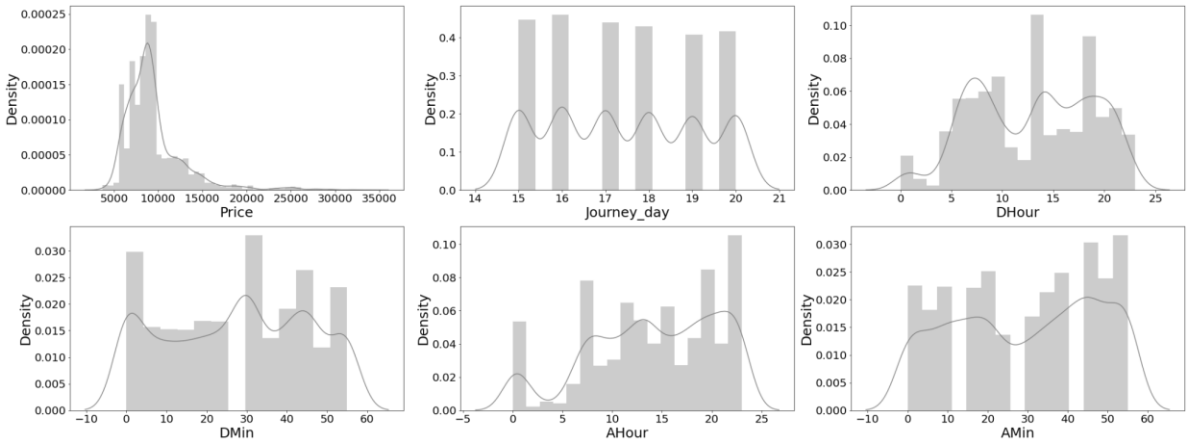
- **I have used mean absolute error which gives a magnitude of difference between the prediction of an observation and the true value of that observation.**
- **I have used root mean square deviation as one of the most commonly used measures for evaluating the quality of predictions.**
- **I have used the R2 score which tells us how accurate our model is.**

Visualizations

I have used count plots for the categorical features & bar plots to see the relation of categorical variables with the target and I have used 2 types of plots for numerical columns one is dist plot for univariate and strip plot for bivariate analysis.

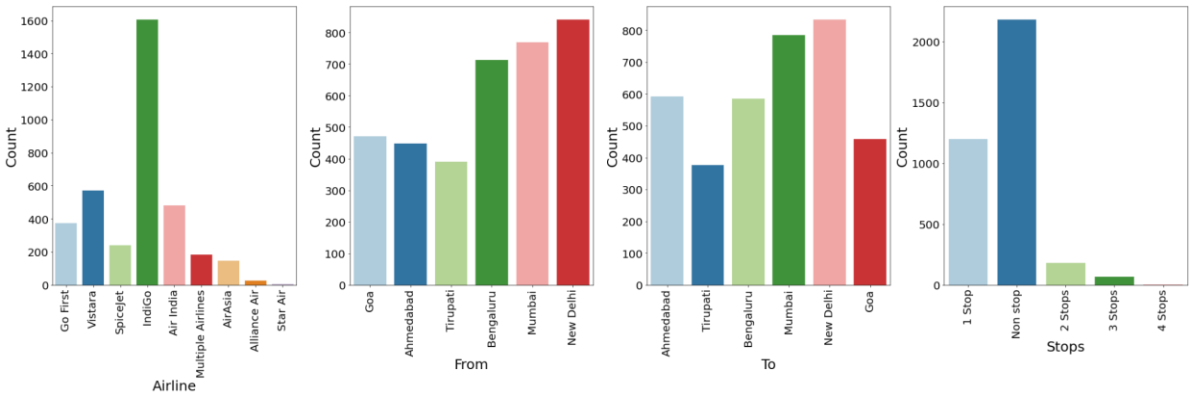
<u>UNIVARIATE ANALYSIS</u>:

Numerical columns:



<u>OBSERVATION</u>: There is no skewness in any of the numerical columns.

Categorical columns:



<u>OBSERVATIONS</u>:

<u>Airline</u>: Indigo has maximum count which means most of the passengers preferred Indigo for their traveling.

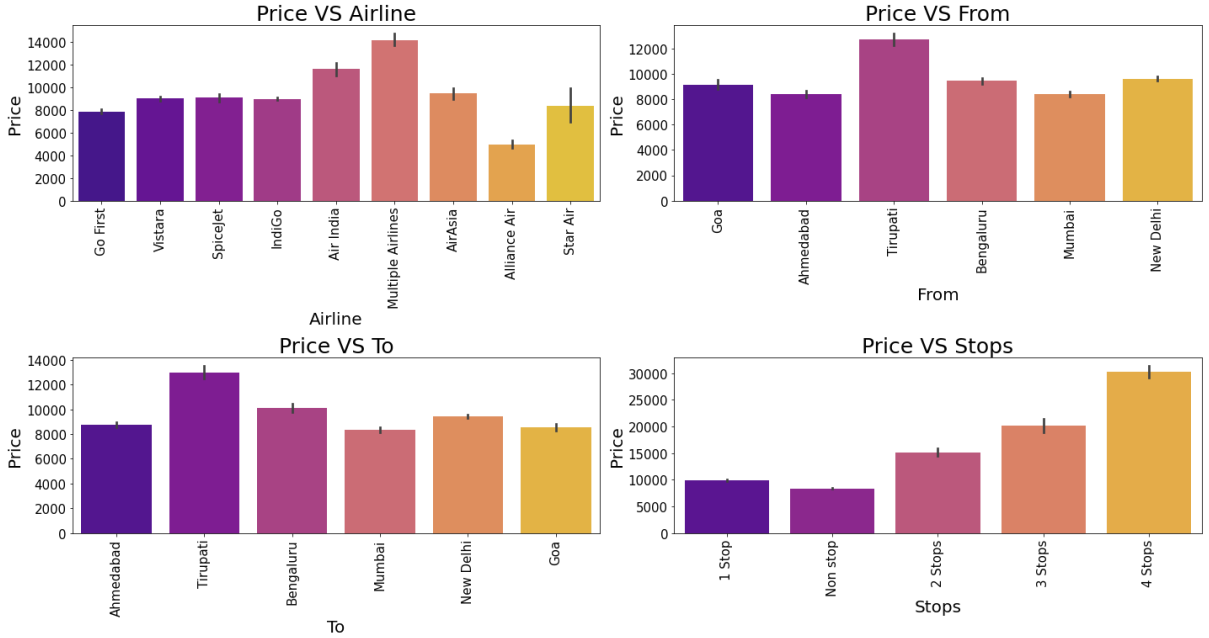
<u>From</u>: New Delhi has maximum count for source which means maximum passengers are choosing New Delhi as their source.

<u>To</u>: New Delhi has maximum count for Destination which means maximum passengers are choosing New Delhi as their Destination.

<u>Stops</u>: Most of the flights have No stops in between and are direct flights. Secondly, most flights have 1 stop in between.

BIVARIATE ANALYSIS:

Categorical columns:



OBSERVATIONS:

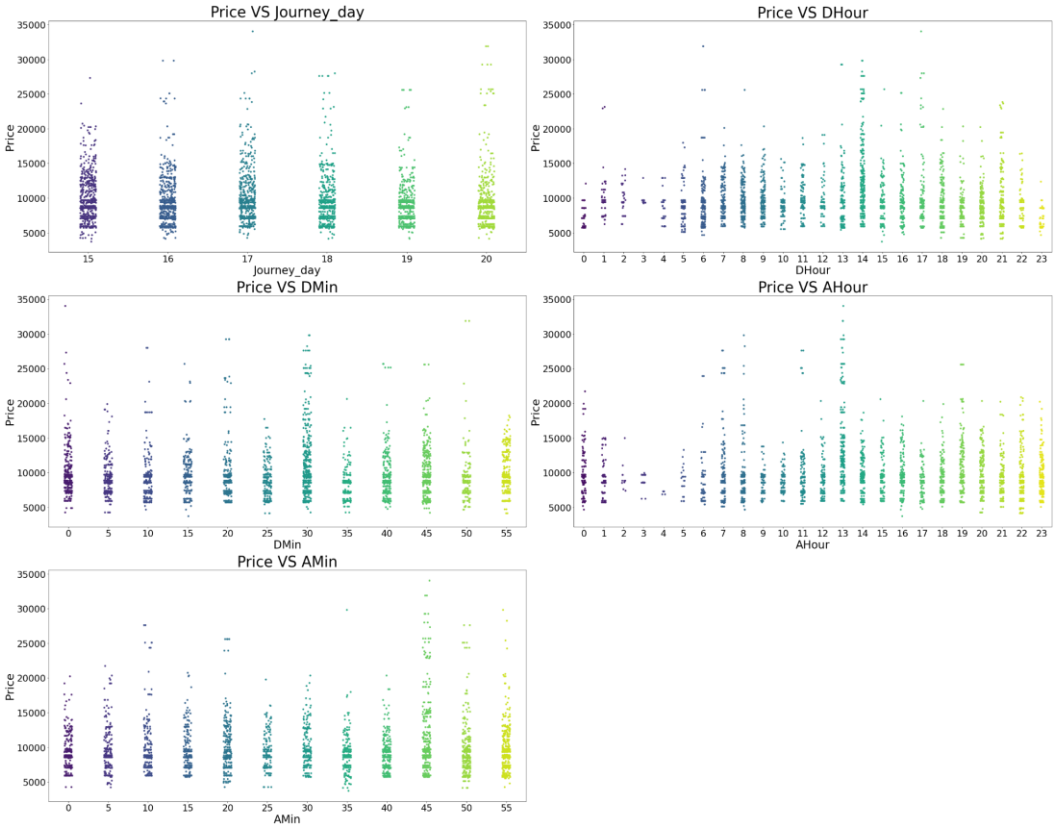
Price vs Airline - For Multiple Airlines the Price is high compared to other Airlines.

Price vs From - Taking Tirupati as Source costs the highest Price Compared to other Source points.

Price vs To - Taking Tirupati as Destination costs highest Price Compared to other Destination points.

Price vs Stops - 4 stops or higher are costlier, compared to no stops & 1 stop.

Numerical columns:



OBSERVATIONS:

Price vs Journey_day - In all the dates the price is almost the same.
Price vs DHour - At 2PM departure time of every day the flight Prices are high so it looks good to book flights rather than this departure time.
Price vs DMin - And Departure minute has less relation with target Price.
Price vs AHour - At 6AM, 7AM, 8AM, 11AM & 1PM Arrival time of every day the flight Prices are high so it looks good to book flights rather than this arrival time.
Price vs AMin - Arrival minute has less relation with target Price.

MODEL BUILDING & PREDICTION

Run and Evaluate selected models

Regression Models:

LINEAR REGRESSION:

```
In [71]: from sklearn.linear_model import LinearRegression

         lr=LinearRegression()
         lr.fit(x_train,y_train)
         lr.score(x_train,y_train)

         pred_lr=lr.predict(x_test)
         print('R2_Score: ',r2_score(y_test,pred_lr))
         print('Mean absolute error: ',mean_absolute_error(y_test,pred_lr))
         print('Mean squared error: ',mean_squared_error(y_test,pred_lr))
         print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_lr)))
```

```
R2_Score:  0.1897812736568698
Mean absolute error:  2101.264726904694
Mean squared error:  10953837.20037541
Root Mean squared error:  3309.65816971714
```

The Linear Regression model gave us an R2 Score of 18.97 %.

DECISION TREE REGRESSOR:

```
In [72]: from sklearn.tree import DecisionTreeRegressor

         dtr = DecisionTreeRegressor()
         dtr.fit(x_train,y_train)
         dtr.score(x_train,y_train)

         pred_dtr=dtr.predict(x_test)
         print('R2_Score: ',r2_score(y_test,pred_dtr))
         print('Mean absolute error: ',mean_absolute_error(y_test,pred_dtr))
         print('Mean squared error: ',mean_squared_error(y_test,pred_dtr))
         print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_dtr)))
```

```
R2_Score:  0.6859877947695018
Mean absolute error:  987.4315886134068
Mean squared error:  4245321.001836548
Root Mean squared error:  2060.417676549235
```

The Decision Tree Regressor Model gave us a R2 Score of 68.59 %.

KNEAREST NEIGHBORS REGRESSOR:

In [73]:
```
from sklearn import neighbors

knn = neighbors.KNeighborsRegressor()
knn.fit(x_train,y_train)
knn.score(x_train,y_train)

pred_knn=knn.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_knn))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_knn))
print('Mean squared error: ',mean_squared_error(y_test,pred_knn))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_knn)))
```

```
R2_Score:  0.6813181031806808
Mean absolute error:  1356.6927456382002
Mean squared error:  4308453.387915519
Root Mean squared error:  2075.68142736681
```

The KNearest Neighbors Regression Model gave us a R2 Score of 68.13 %

SUPPORT VECTOR REGRESSOR (SVR):

In [74]:
```
from sklearn.svm import SVR

svr=SVR()
svr.fit(x_train,y_train)
svr.score(x_train,y_train)

pred_svr=svr.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_svr))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_svr))
print('Mean squared error: ',mean_squared_error(y_test,pred_svr))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_svr)))
```

```
R2_Score:  -0.06085742829601459
Mean absolute error:  2240.540971724598
Mean squared error:  14342373.466004254
Root Mean squared error:  3787.132617958375
```

The SVR Model gave us a R2 Score of -6.08 %.

RANDOM FOREST REGRESSOR:

In [75]:
```
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor()
rfr.fit(x_train,y_train)
rfr.score(x_train,y_train)

pred_rfr=rfr.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_rfr))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_rfr))
print('Mean squared error: ',mean_squared_error(y_test,pred_rfr))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_rfr)))
```

```
R2_Score:  0.8163277124591364
Mean absolute error:  858.1481154619792
Mean squared error:  2483176.789832809
Root Mean squared error:  1575.809883784465
```

The Random Forest Regression Model gave us a R2 Score of 81.63 %.

GRADIENT BOOSTING REGRESSOR:

In [76]:
```
from sklearn.ensemble import GradientBoostingRegressor

gbr=GradientBoostingRegressor()
gbr.fit(x_train,y_train)
gbr.score(x_train,y_train)

pred_gbr=gbr.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_gbr))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_gbr))
print('Mean squared error: ',mean_squared_error(y_test,pred_gbr))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_gbr)))
```

```
R2_Score:  0.7485119744115258
Mean absolute error:  1154.6295506731049
Mean squared error:  3400018.7857585284
Root Mean squared error:  1843.9139854555388
```

The Gradient Boosting Regressor Model gave us a R2 Score of 74.85 %.

EXTRA TREES REGRESSOR:

In [77]:
```
from sklearn.ensemble import ExtraTreesRegressor

etr=ExtraTreesRegressor()
etr.fit(x_train,y_train)
etr.score(x_train,y_train)

pred_etr=etr.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_etr))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_etr))
print('Mean squared error: ',mean_squared_error(y_test,pred_etr))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_etr)))
```

```
R2_Score:  0.8128817927484211
Mean absolute error:  807.9719237832874
Mean squared error:  2529764.2634241735
Root Mean squared error:  1590.5232671747287
```

The Extra Trees Regressor Model gave us a R2 Score of 81.28 %.

In [78]:
```
from xgboost import XGBRegressor

xgb=XGBRegressor()
xgb.fit(x_train,y_train)
xgb.score(x_train,y_train)

pred_xgb=xgb.predict(x_test)
print('R2_Score: ',r2_score(y_test,pred_xgb))
print('Mean absolute error: ',mean_absolute_error(y_test,pred_xgb))
print('Mean squared error: ',mean_squared_error(y_test,pred_xgb))
print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred_xgb)))
```

```
R2_Score:  0.8135892888736583
Mean absolute error:  906.873558023416
Mean squared error:  2520199.1952224993
Root Mean squared error:  1587.5135259967076
```

XGBOOST REGRESSOR:

The XGBoost Regressor Model gave us a R2 Score of 81.35 %.

From the above regression models, the highest R2 score belongs to the Random Forest Regressor Model, followed closely by XGBoost Regressor Model & Extra Trees Regressor Model.Next, the Gradient Boosting Regressor Model.
After that, the KNearest Neighbors Regression Model & Decision Tree Regressor Model. Lastly, the Linear Regression Model.
Finally, the lowest score belongs to the SVR Model.

Hyper Parameter Tuning:

Since the R2 Score is the Highest and the RMSE score is the lowest in Random Forest Regressor Model, we shall consider it for hyper parameter tuning.

We shall use GridSearchCV for Hyper Parameter Tuning.

```
In [79]: from sklearn.model_selection import GridSearchCV
```

```
In [80]: parameters={
             'criterion': ['squared_error','absolute_error'],
             'max_depth': [10,20,30],
             'max_features': ['sqrt','log2'],
             'n_estimators': [100,200,300]}
         grid_rfr = GridSearchCV(rfr, param_grid = parameters, cv = 5)
```

```
In [81]: grid_rfr.fit(x_train, y_train)
```

```
Out[81]: GridSearchCV(cv=5, estimator=RandomForestRegressor(),
                       param_grid={'criterion': ['squared_error', 'absolute_error'],
                                   'max_depth': [10, 20, 30],
                                   'max_features': ['sqrt', 'log2'],
                                   'n_estimators': [100, 200, 300]})
```

```
In [82]: grid_rfr.best_params_
```

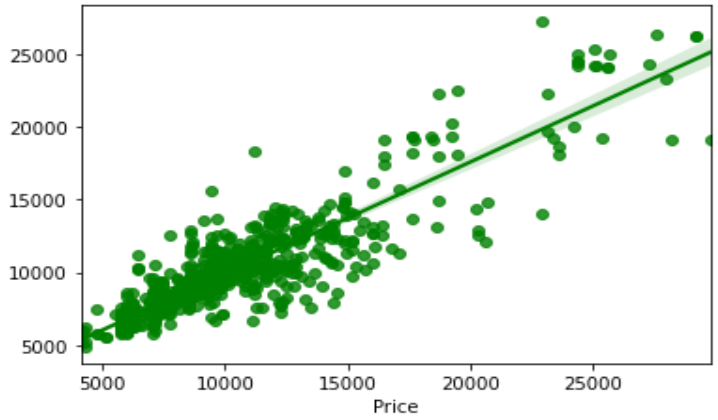
```
Out[82]: {'criterion': 'squared_error',
          'max_depth': 20,
          'max_features': 'log2',
          'n_estimators': 300}
```

```
In [112]: flight_model = RandomForestRegressor(criterion='squared_error',max_features='log2',max_depth=20,n_estimators=300)
          flight_model.fit(x_train,y_train)

          pred = flight_model.predict(x_test)
          print('R2_Score: ',r2_score(y_test,pred))
          print('Mean absolute error: ',mean_absolute_error(y_test,pred))
          print('Mean squared error: ',mean_squared_error(y_test,pred))
          print('Root Mean squared error: ',np.sqrt(mean_squared_error(y_test,pred)))
```

```
R2_Score:  0.8149575169163876
Mean absolute error:  896.3206010910508
Mean squared error:  2501701.2924392736
Root Mean squared error:  1581.6767344938958
```

After Hyper Parameter Tuning, we have got a R2 score of 81.49 %.



Saving the final model and predicting the saved model

Now we shall save the best model.

In [114]:
```
import joblib
joblib.dump(flight_model,"Flight_Price_Prediction_Model.pkl")
```

Out[114]:
```
['Flight_Price_Prediction_Model.pkl']
```

In [115]:
```
# Loading the saved model
flight_price_model=joblib.load("Flight_Price_Prediction_Model.pkl")

# Prediction
prediction = flight_price_model.predict(x_test)
prediction
```

Out[115]:
```
array([ 6001.35333333,  8701.97666667,  7086.4       , ...,
       14868.03      ,  8612.97666667,  9047.88333333])
```

Putting the predicted & actual values in a dataframe.

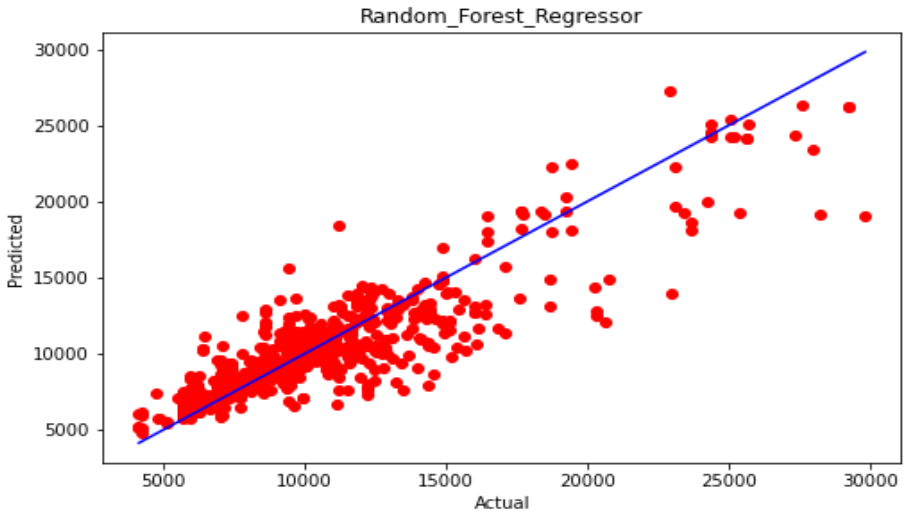
In [116]:
```
pd.DataFrame([flight_price_model.predict(x_test)[:],y_test[:]],index=["Predicted","Actual"])
```

Out[116]:

	0	1	2	3	4	5	6	7	8	9	...	1079	1080
Predicted	6001.353333	8701.976667	7086.4	9271.646667	13887.37	9417.62	5818.873333	13789.773333	9757.893333	8464.743067	...	8137.72	19092.042667
Actual	5943.000000	8567.000000	7056.0	9314.000000	11543.00	8839.00	5943.000000	13830.000000	9679.000000	9597.000000	...	8346.00	16480.000000

2 rows × 1089 columns

Graph of predicted and actual values.



Interpretation of the Results

The dataset was scrapped from makemytrip.com website. The dataset was very challenging to handle; it had 9 features with 3631 samples.

- Firstly, the datasets had 2 complete rows as nan values, so I have dropped those rows.
- And there was a huge number of unnecessary entries in all the features so I have used feature extraction to get the required format of variables.
- And proper plotting for the proper type of features will help us to get better insight on the data. I found both numerical columns and categorical columns in the dataset so I have chosen strip plot and bar plot to see the relation between target and features.
- I did not find any outliers or skewness in the dataset.
- I then encoded the data set with Label Encoder.
- Then scaling the dataset has a good impact like it will help the model not to get biased.Since we did not have outliers and skewness in the dataset so we have to choose Standardization.
- We used multiple algorithms while building a model using the dataset to get the best model out of it.
- And we have to use multiple metrics like mse, mae, rmse and r2_score which will help us to decide the best model.
- I found RandomForestRegressor to be the best model with 81.49% r2_score.
- Also I have improved the accuracy of the best model by running hyper parameter tuning.
- At last I have predicted the used flight price using the saved model. I was able to get the predictions near to actual values.

CONCLUSION

In this project, we have used machine learning algorithms to predict the flight prices. We have mentioned the step by step procedure to analyze the dataset and find the correlation between the features. Thus we can select the features which are correlated to each other and are independent in nature.

These feature sets were then given as an input to 8 algorithms and a hyper parameter tuning was done to the best model and the accuracy has been improved.

Thus, we calculated the performance of each model using different performance metrics and compared them based on those metrics. Then we have also saved the best model and predicted the flight price.

It was good that the predicted and actual values were almost the same.

Limitations of this work and Scope for Future Work

LIMITATIONS:

- First drawback is scraping the data as it is a fluctuating process.
- Followed by raw data which is not in format to analyze.
- Also, we have tried best to deal with improper format data and null values. So it looks quite good that we have achieved an accuracy of 81.49% even after dealing with all these drawbacks.
- Also, this study will not cover all Regression algorithms; instead, it is focused on the chosen algorithm.

FUTURE WORK:

- Future direction of research may consider incorporating additional used flight data from a larger economical background with more features.