Flight Price Prediction

Pavani Lakshmi Gunnam, Srikeerthana Reddy Bandi, Chandrika Sowmini Devabhakthuni
Indiana University, Bloomington

Abstract: Air travel has evolved into a popular, necessary, and quick mode of transportation owing to the ever-increasing connectivity of air routes around the globe. For airlines, predicting prices is a crucial yet difficult task because prices are always fluctuating and are known to depend on a wide range of factors. In this project, we employ a machine learning regression approach to forecast flight fares by using simple information such as the airline name, source, destination, number of stops, and departure and arrival times.

Key Words: Flight Price**,** Regression Model**,** Prediction Model, Random Forest

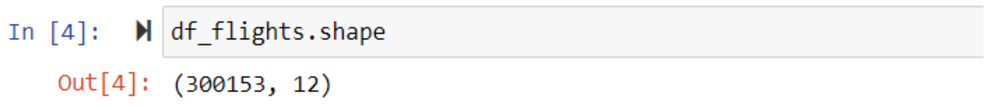
I. INTRODUCTION

This whole process gives the idea to predict about the airline tickets to make it easier for tourists to book their tickets in accordance with their demands. It's extremely difficult for a customer to purchase an airplane ticket airplane ticket at the lowest price due to the high complexity of the pricing models used by the airlines, as the price changes constantly. In this project, prediction of flights fares is done using regression models, a data of decent size is taken from Kaggle to train and test the models built. The models have been trained and tested using variety of figures of variables for validity. This project examines various regression techniques for forecasting flight fares using dataset of around 3 lakh records for testing and training. Three regression models have been trained and their performance has been evaluated using the R Squared Score and Root Mean Square Error followed by tuning of the hyperparameters of top model.

II. METHODOLOGY

Data collection:

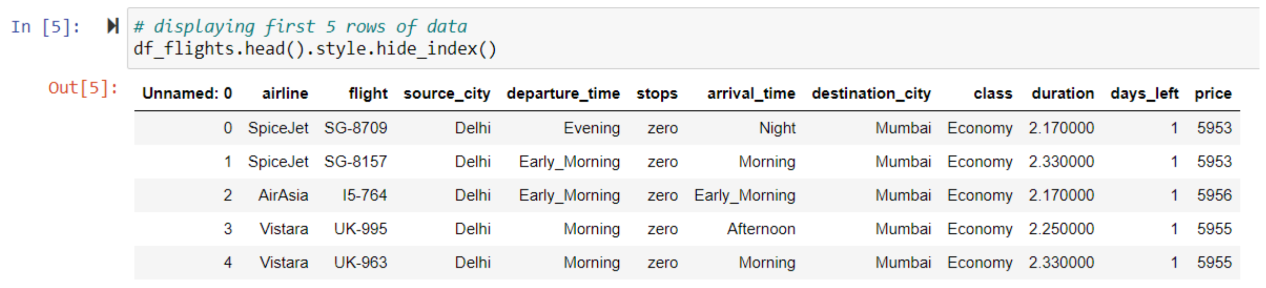
1. **Dataset**
   The Flight Price Prediction dataset have been extracted from Kaggle. The dataset contains 3,00,153 records about flight details between the 6 top metro cities in India which were taken from the website Easemytrip.



```
In [4]:  df_flights.shape
Out[4]:  (300153, 12)
```

Fig.1: Code snippet for shape function

This dataset contains 12 attributes among which 8 are categorical and 4 are numerical as shown below using head() function.



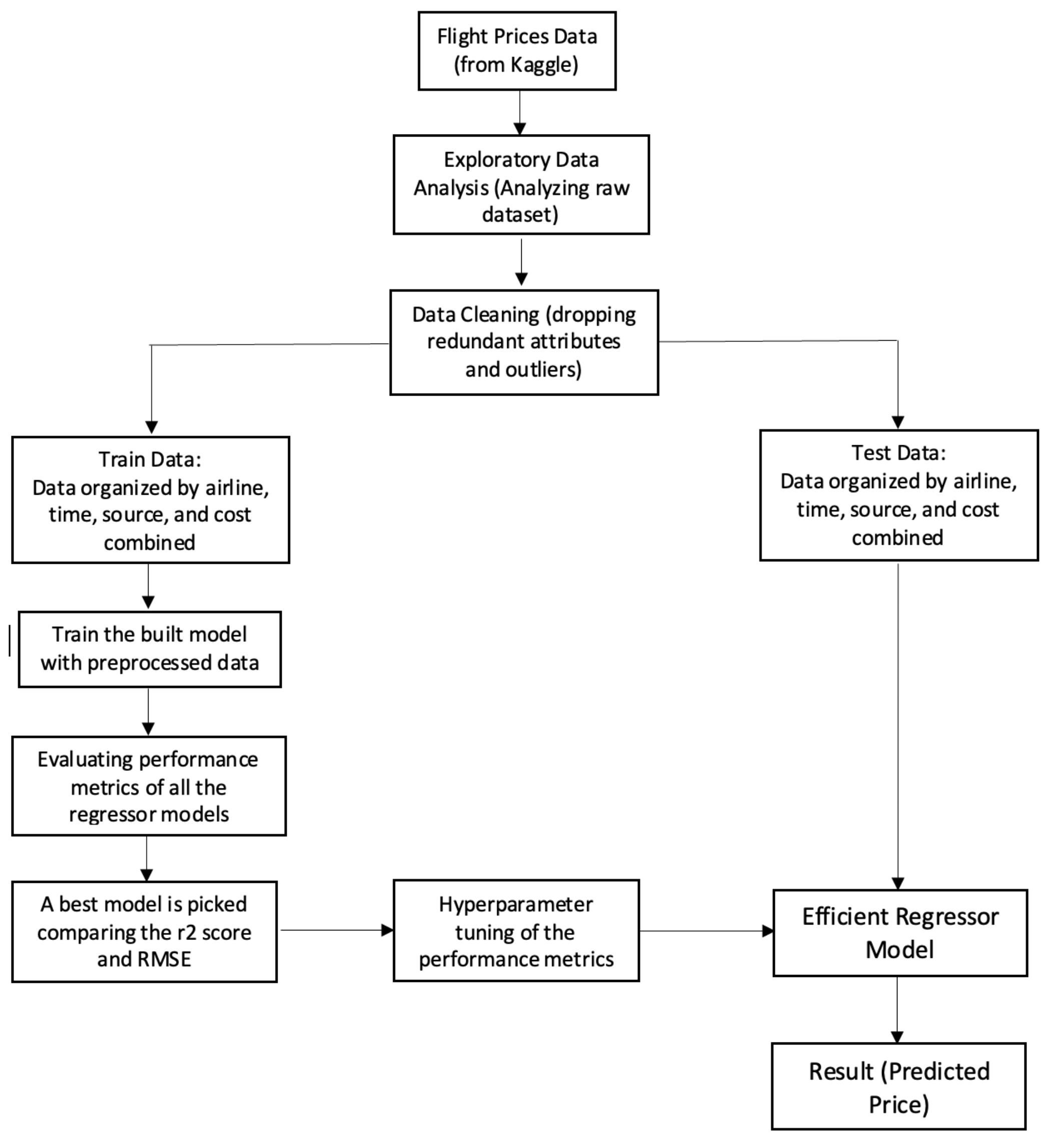
```
In [5]:  # displaying first 5 rows of data
         df_flights.head().style.hide_index()
```

Out[5]:

Unnamed: 0	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
0	SpiceJet	SG-8709	Delhi	Evening	zero	Night	Mumbai	Economy	2.170000	1	5953
1	SpiceJet	SG-8157	Delhi	Early_Morning	zero	Morning	Mumbai	Economy	2.330000	1	5953
2	AirAsia	I5-764	Delhi	Early_Morning	zero	Early_Morning	Mumbai	Economy	2.170000	1	5956
3	Vistara	UK-995	Delhi	Morning	zero	Afternoon	Mumbai	Economy	2.250000	1	5955
4	Vistara	UK-963	Delhi	Morning	zero	Morning	Mumbai	Economy	2.330000	1	5955

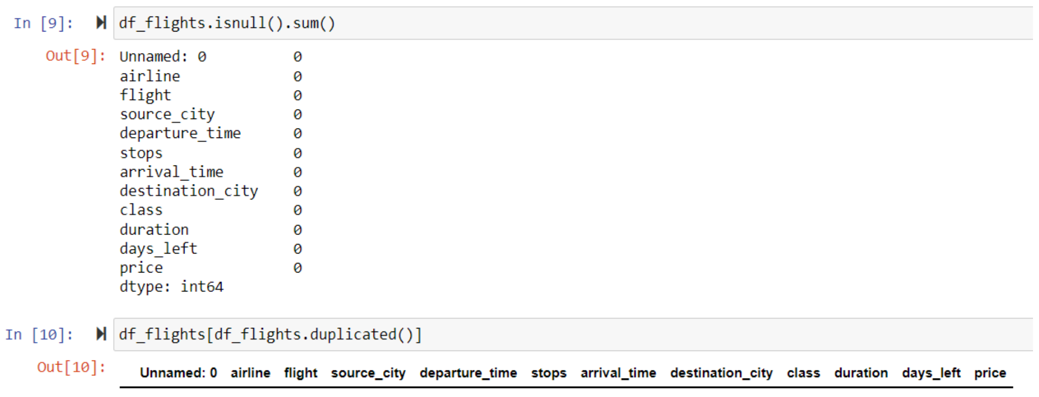
Fig.2: Code snippet for displaying the first 5 rows of the data set

Workflow of the Model


Flight Prices Data (from Kaggle)
Exploratory Data Analysis (Analyzing raw dataset)
Data Cleaning (dropping redundant attributes and outliers)
Train Data: Data organized by airline, time, source, and cost combined
Test Data: Data organized by airline, time, source, and cost combined
Train the built model with preprocessed data
Evaluating performance metrics of all the regressor models
A best model is picked comparing the r2 score and RMSE
Hyperparameter tuning of the performance metrics
Efficient Regressor Model
Result (Predicted Price)

2. Exploratory Data Analysis

Data modelling is made simpler by EDA, which makes it easy to understand a dataset's structure. The main objective of EDA is to make data "clean," which implies that it should be free of redundancies. This helps us to discover the patterns associated with the data.



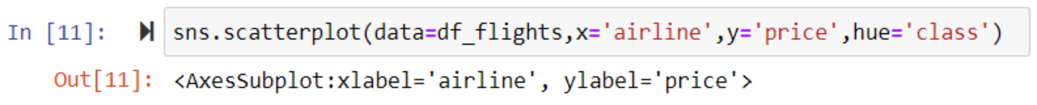
```
In [9]:  df_flights.isnull().sum()

Out[9]: Unnamed: 0          0
        airline             0
        flight              0
        source_city         0
        departure_time      0
        stops               0
        arrival_time        0
        destination_city    0
        class               0
        duration            0
        days_left           0
        price               0
        dtype: int64

In [10]:  df_flights[df_flights.duplicated()]

Out[10]:
   Unnamed: 0  airline  flight  source_city  departure_time  stops  arrival_time  destination_city  class  duration  days_left  price
```

Fig.3: Checking for null and duplicate values

As shown above, our dataset has zero null and duplicate values. In general, we will find few records containing null values which can be removed completely from out dataset or can be replaced with the mean of the values of that attribute. Fortunately, the dataset is free from such type of data.

a. Checking outliers

In the below scatterplot between airline and price, there are a few records which differ significantly from the rest of the data. These data points are not much important for our analysis. So, these are removed as shown in the later part of this report.



```
In [11]:  sns.scatterplot(data=df_flights,x='airline',y='price',hue='class')

Out[11]: <AxesSubplot:xlabel='airline', ylabel='price'>
```

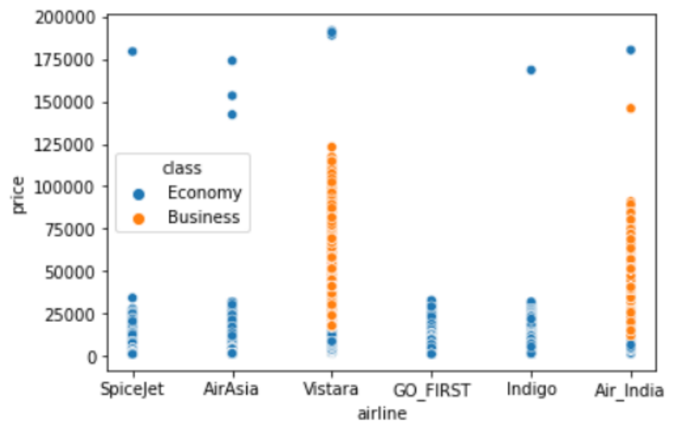


Fig.4: Scatter plot to visualize the outliers

b. Data Visualization

To understand the data patterns between the “target attribute - Price” and other attributes. Following graphs are plotted.

We have considered the attributes Airline, Class, Source_city and Destination_city to understand how price fluctuates.

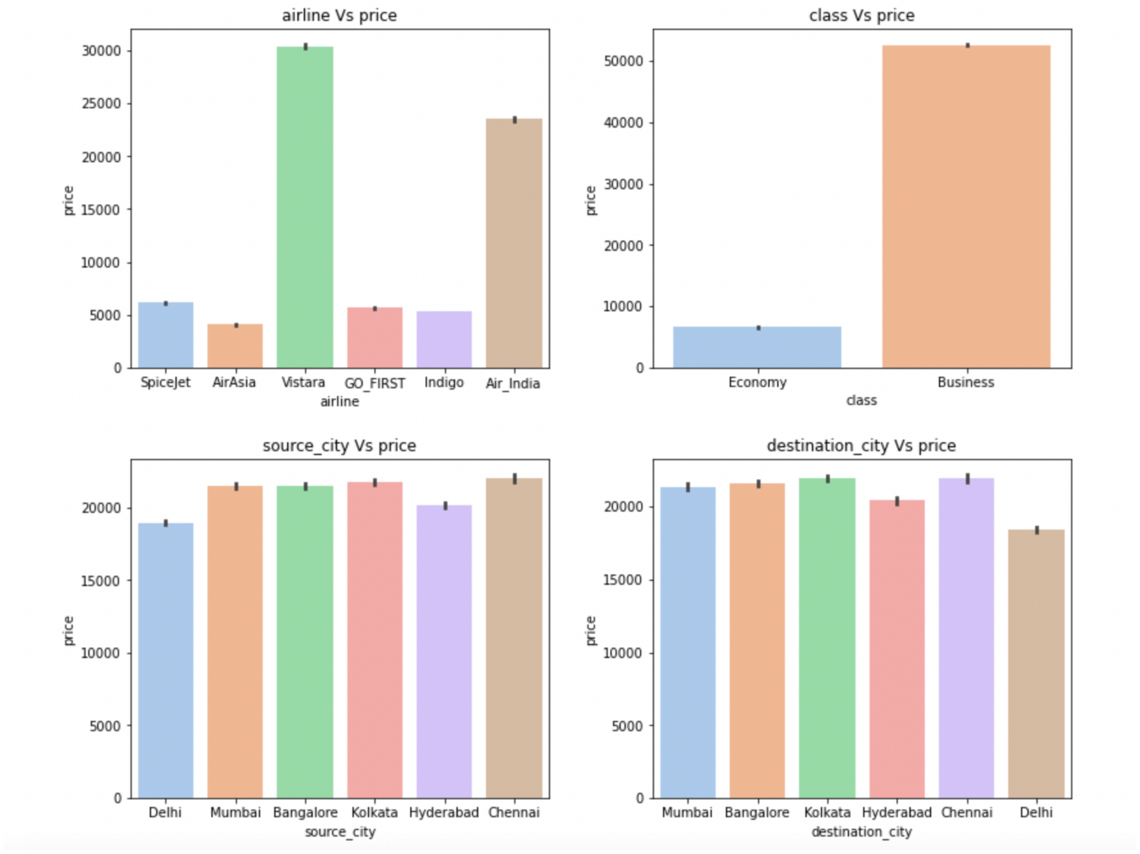


Fig.5: The above figure represents the subplots of bar graph of other attributes vs Price

Airline Vs Price: - From the subplot 1 (airline vs price), Vistara has the maximum price range followed by Air India when compared with the rest of the airlines.

Class Vs Price: - We have Economy and Business classes. The flight fare for economy is less than 10k whereas the fare for business class is reaching above 50k.

Source_city vs price: - Our dataset contains 6 major cities of India; among them the flight fare is maximum in Chennai followed by Kolkata. Delhi has the least Price Range when compared to the rest.

Destination_City vs Price: - Similar to subplot 3, the price at Chennai is maximum followed by Kolkata. Mumbai and Bangalore are a bit less when compared with the highest and Delhi has the least range.

3. Data Cleaning

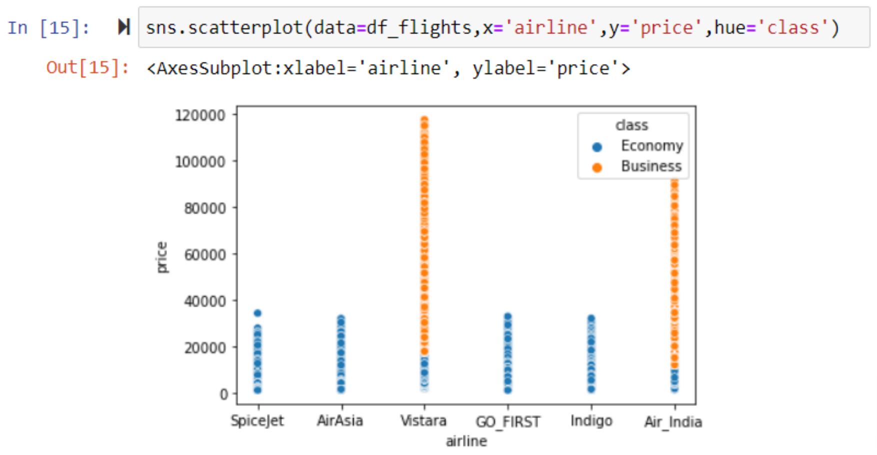
a. Dropping redundant columns

Out of the 12 attributes present in the dataset, the first column is Unnamed, it is removed from the dataset as it doesn't add any value in our analysis.

b. Dropping outliers

We have used the IQR (Interquartile range) technique to set the upper and lower bound for removing the outliers present in the price attribute. We have used 2.1IQR and 1.6IQR for setting the upper and lower bounds respectively. Even though for most datasets using ±1.5IQR is more reasonable for detecting outliers, we have used values higher than 1.5 as we want to make the range very inclusive so that the few outliers which were detected can be dropped and not the useful data.

The below scatterplot represents the price attribute free from all the outliers.



```
In [15]:  sns.scatterplot(data=df_flights,x='airline',y='price',hue='class')
Out[15]: <AxesSubplot:xlabel='airline', ylabel='price'>
```

Fig.6: Datatype conversion of 'stops' attribute

The values of 'stops' attribute represents the number of stops the flight takes during the whole journey. In the dataset, it is a categorical attribute, and its values are:

- zero
- one
- two or more

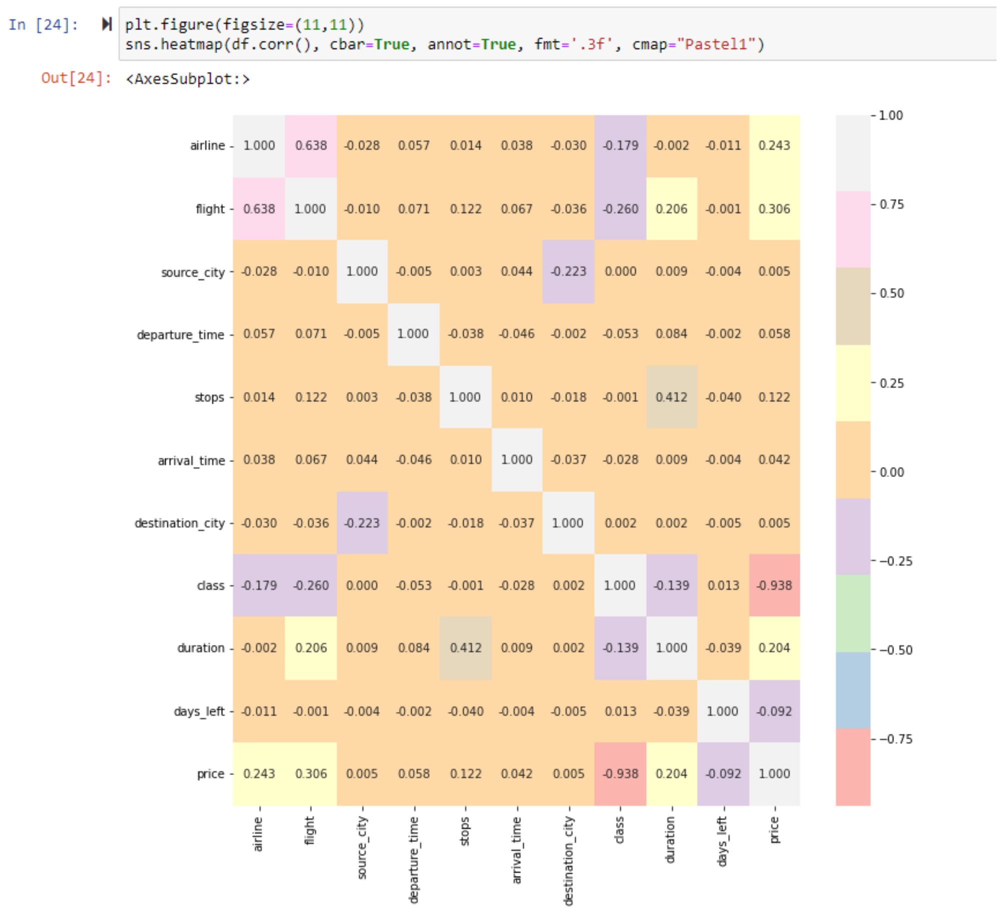
We want to closely understand the behavior of this attribute and how it affects the price of the ticket. So, we have implemented a code in python to assign value for stops based on the price range and converted its datatype to integer. Instead of implementing a new code, we can make use of the label encoder, however, it won't suffice our purpose of assigning the values based on the price range.

4. Data Analysis:

a. Correlation

The below heatmap for correlation matrix is plotted to understand how strongly the price attribute is affected by other features. The matrix depicts the relation between numerical attributes. We have used label encoder as there are 7 categorical features after the conversion of ‘stops’ attribute to int. The correlation coefficient's value fluctuates from +1 to -1 depending on how strong the association is.

From the heatmap we can infer that, price is highly negatively correlated with the class attribute. Economy and Business are the two categories under the class attribute. In general, the price of Business class is comparatively higher than Economy. As we have used label encoder, the string value of Business class is interpreted as 0 and Economy as 1. Hence, the negative correlation.



```
In [24]:  plt.figure(figsize=(11,11))
          sns.heatmap(df.corr(), cbar=True, annot=True, fmt='.3f', cmap="Pastel1")

Out[24]: <AxesSubplot:>
```

	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price
airline	1.000	0.638	-0.028	0.057	0.014	0.038	-0.030	-0.179	-0.002	-0.011	0.243
flight	0.638	1.000	-0.010	0.071	0.122	0.067	-0.036	-0.260	0.206	-0.001	0.306
source_city	-0.028	-0.010	1.000	-0.005	0.003	0.044	-0.223	0.000	0.009	-0.004	0.005
departure_time	0.057	0.071	-0.005	1.000	-0.038	-0.046	-0.002	-0.053	0.084	-0.002	0.058
stops	0.014	0.122	0.003	-0.038	1.000	0.010	-0.018	-0.001	0.412	-0.040	0.122
arrival_time	0.038	0.067	0.044	-0.046	0.010	1.000	-0.037	-0.028	0.009	-0.004	0.042
destination_city	-0.030	-0.036	-0.223	-0.002	-0.018	-0.037	1.000	0.002	0.002	-0.005	0.005
class	-0.179	-0.260	0.000	-0.053	-0.001	-0.028	0.002	1.000	-0.139	0.013	-0.938
duration	-0.002	0.206	0.009	0.084	0.412	0.009	0.002	-0.139	1.000	-0.039	0.204
days_left	-0.011	-0.001	-0.004	-0.002	-0.040	-0.004	-0.005	0.013	-0.039	1.000	-0.092
price	0.243	0.306	0.005	0.058	0.122	0.042	0.005	-0.938	0.204	-0.092	1.000

Fig.7: Heatmap of correlation matrix

5. Modeling

The dataset is split into train and test data. The model is trained using the train data, and it is then tested using the remaining dataset's data. After splitting, the dataset contains 67.7% of train data and the rest of 1/3rd is the test data.

a. Fitting the Regression Models

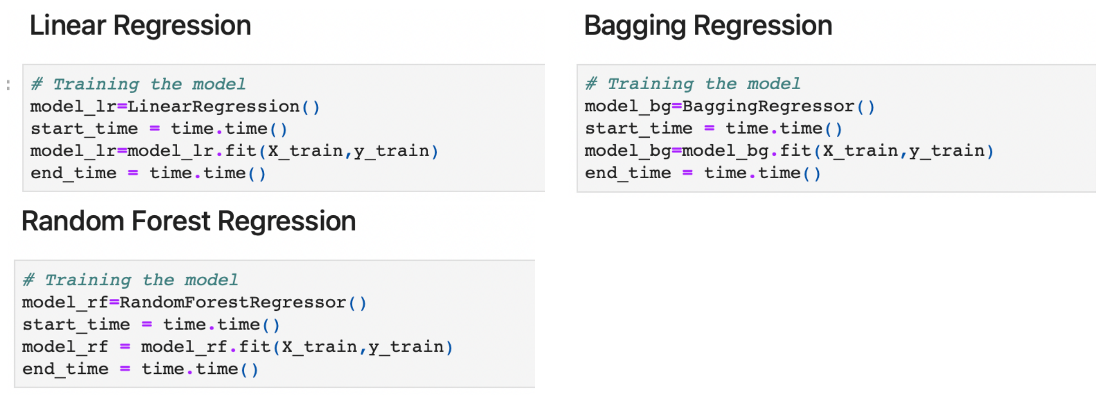
We now proceed to the main step of our machine learning, fitting the model and predicting the outputs. The data is fit into multiple models to compare the results and to select the best model. We have used Linear Regressor, Random Forest Regressor and Bagging Regressor to train the model.

To compare the results of the model, we are mainly focusing on 2 metrics, they are:

1. R2 Score: - R2 score means the proportion of the variance in the dependent variable that is predictable from the independent variables.
2. RMSE: - Root mean square error or root mean square deviation is one of the most used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

b. Models:

We use the code snippets listed below to train the data into ML models.



Linear Regression

```
# Training the model
model_lr=LinearRegression()
start_time = time.time()
model_lr=model_lr.fit(X_train,y_train)
end_time = time.time()
```

Bagging Regression

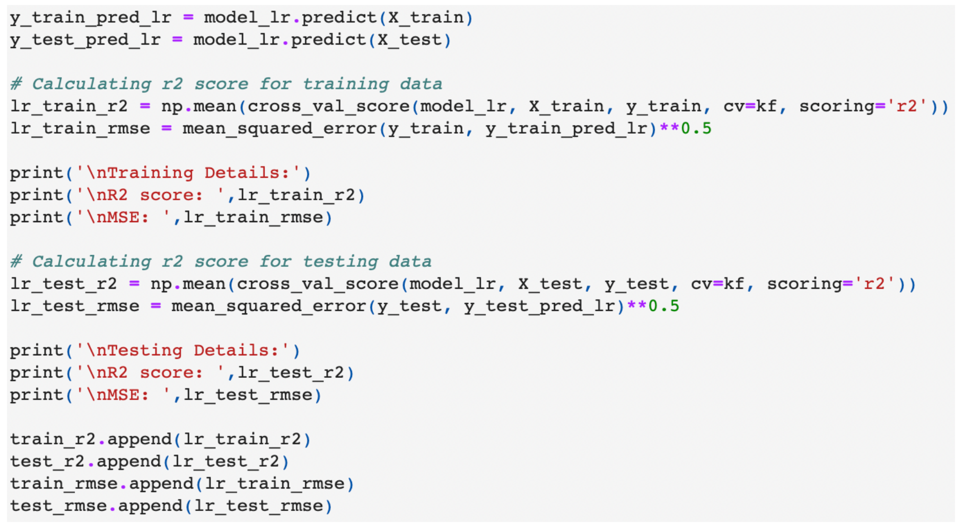
```
# Training the model
model_bg=BaggingRegressor()
start_time = time.time()
model_bg=model_bg.fit(X_train,y_train)
end_time = time.time()
```

Random Forest Regression

```
# Training the model
model_rf=RandomForestRegressor()
start_time = time.time()
model_rf = model_rf.fit(X_train,y_train)
end_time = time.time()
```

Fig.8 Code Snippets for fitting the base models

The next step in modeling is to fit the data into our models to predict the output. The below snippet of code is used for fitting the data.



```
y_train_pred_lr = model_lr.predict(X_train)
y_test_pred_lr = model_lr.predict(X_test)

# Calculating r2 score for training data
lr_train_r2 = np.mean(cross_val_score(model_lr, X_train, y_train, cv=kf, scoring='r2'))
lr_train_rmse = mean_squared_error(y_train, y_train_pred_lr)**0.5

print('\nTraining Details:')
print('\nR2 score: ',lr_train_r2)
print('\nMSE: ',lr_train_rmse)

# Calculating r2 score for testing data
lr_test_r2 = np.mean(cross_val_score(model_lr, X_test, y_test, cv=kf, scoring='r2'))
lr_test_rmse = mean_squared_error(y_test, y_test_pred_lr)**0.5

print('\nTesting Details:')
print('\nR2 score: ',lr_test_r2)
print('\nMSE: ',lr_test_rmse)

train_r2.append(lr_train_r2)
test_r2.append(lr_test_r2)
train_rmse.append(lr_train_rmse)
test_rmse.append(lr_test_rmse)
```

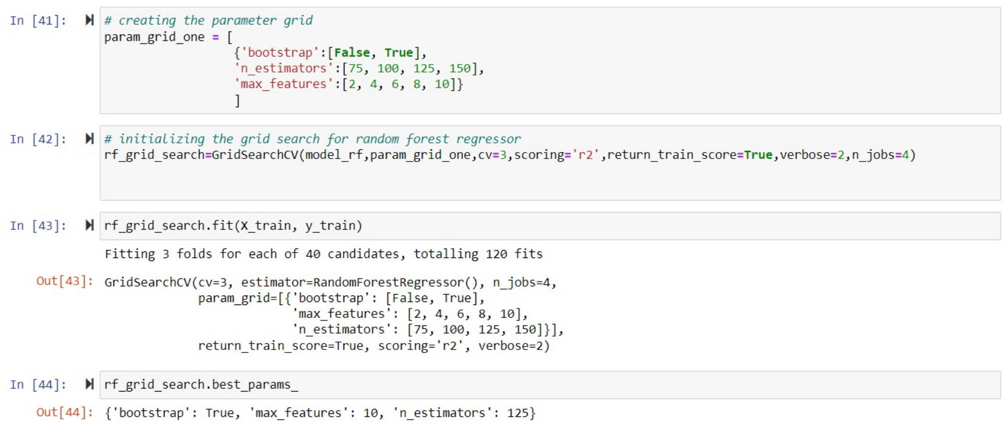
Fig.9: Code Snippet for training the base models

Among the three models, we achieve the best score using Random Forest Regressor. Random Forest Regressor provides an R2 score of 0.9885 (98.85%) and the root mean square value is 894.93 which is very less when compared with the other models.

Now, we will try to tune our model with Hyperparameters to check if the R2 score can be further improved for our models.

c. Hyperparameter Tuning:

We have applied Gridsearch CV hyper parameter tuning technique to validate the model with different parameter combinations. A grid of parameters will be created and after trying all the combinations, the Gridsearch CV will give us the parameters which can further improve the model. The below code snippet is used to apply the grid search on all the three models.



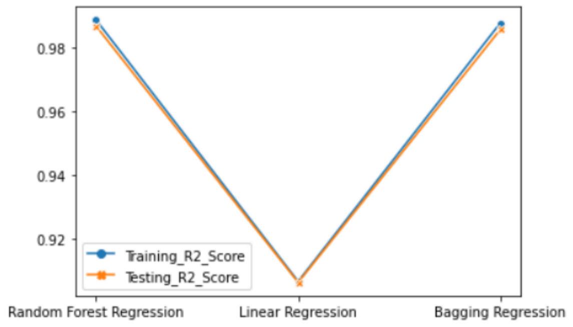
```
In [41]:  # creating the parameter grid
          param_grid_one = [
                            {'bootstrap':[False, True],
                            'n_estimators':[75, 100, 125, 150],
                            'max_features':[2, 4, 6, 8, 10]}
                            ]

In [42]:  # initializing the grid search for random forest regressor
          rf_grid_search=GridSearchCV(model_rf,param_grid_one,cv=3,scoring='r2',return_train_score=True,verbose=2,n_jobs=4)

In [43]:  rf_grid_search.fit(X_train, y_train)

          Fitting 3 folds for each of 40 candidates, totalling 120 fits

Out[43]:  GridSearchCV(cv=3, estimator=RandomForestRegressor(), n_jobs=4,
                       param_grid=[{'bootstrap': [False, True],
                                    'max_features': [2, 4, 6, 8, 10],
                                    'n_estimators': [75, 100, 125, 150]}],
                       return_train_score=True, scoring='r2', verbose=2)

In [44]:  rf_grid_search.best_params_

Out[44]:  {'bootstrap': True, 'max_features': 10, 'n_estimators': 125}
```

Fig.10: Code Snippet for Hyperparameter tuning with GridSearchCV

Gridsearch CV gave max_features as 10 and n_estimators as 125 as the best combination to improve the model.

III. RESULTS

1. Experimental Results and Performance Analysis:



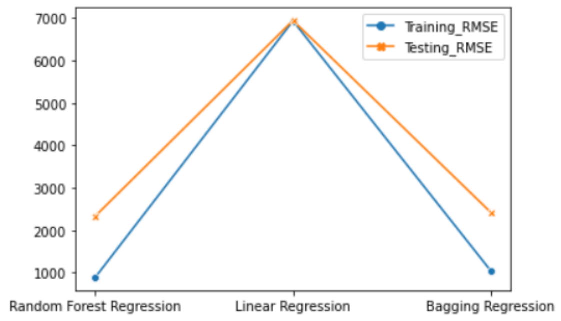


Fig.11: The above plots represent the performance metrics of the three baseline models

The above figures are plotted to visualize the performance metrics obtained through all the three baseline models. In the R2 score figure, the testing and training R2 values of both the Random Forest and Bagging are almost similar. The variation in the value of Linear regression with other models is clearly visible. In the RMSE figure, linear regression model gives a high root mean squared error which makes it the most unfit model for our dataset.

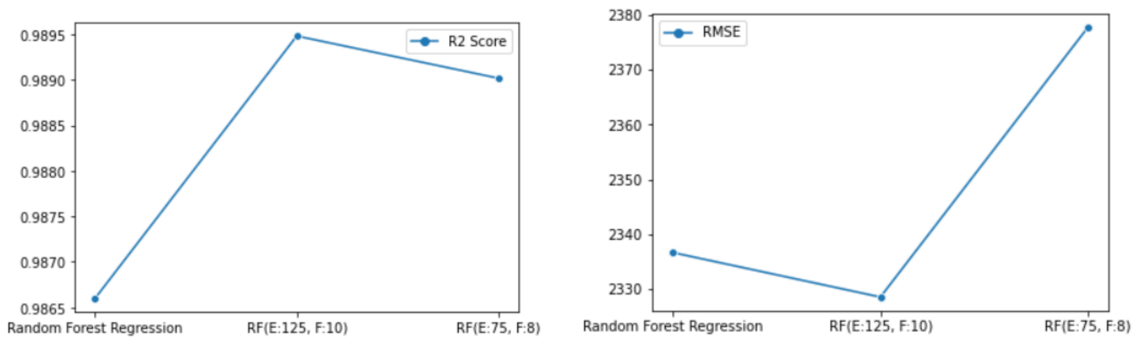


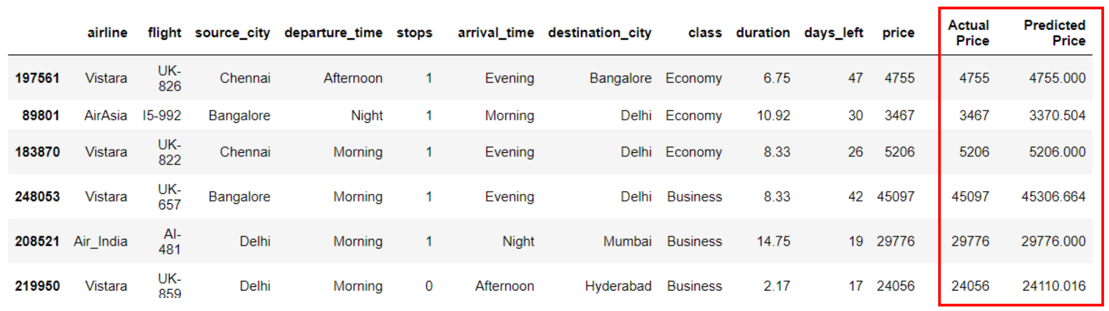
Fig.12: The above plots represent the R2 Score of RF baseline model along with the tuned RF model with different estimators and max_features

The above figures are plotted to visualize the performance metrics obtained through the baseline and hyperparameter tuned random forest models. In the R2 score figure, the value is far better for model with 125 estimators and 10 features as compared with the baseline model. Also, the RMSE is least for the tuned model. So, the tuning of baseline random forest model has improved the performance metrics and it makes it the best fitted model.

Model Name	Training		Testing	
	R2 Score	RMSE	R2 Score	RMSE
Random Forest	0.988519	894.934061	0.986600	2336.642827
Bagging	0.987559	1049.308118	0.985516	2424.548686
Linear	0.906749	6931.158116	0.906435	6941.098790
RF (E:125, F:10)	-	-	0.989472	2328.533757
RF (E:75, F:8)	-	-	0.989023	2377.726833

Table 1: The table consists of the results of the three baseline models along with the Random Forest Hyperparameter Tuned models

2. **Price Prediction with Random Forest:**
The highlighted area shows the actual and predicted ticket price on the test data. As we can see the values are almost similar and we have used the tuned random forest model (n_estimators = 125, max_features = 10) for this prediction as it is the efficient model.

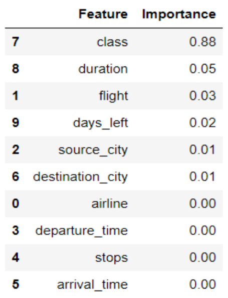


	airline	flight	source_city	departure_time	stops	arrival_time	destination_city	class	duration	days_left	price	Actual Price	Predicted Price
197561	Vistara	UK-826	Chennai	Afternoon	1	Evening	Bangalore	Economy	6.75	47	4755	4755	4755.000
89801	AirAsia	I5-992	Bangalore	Night	1	Morning	Delhi	Economy	10.92	30	3467	3467	3370.504
183870	Vistara	UK-822	Chennai	Morning	1	Evening	Delhi	Economy	8.33	26	5206	5206	5206.000
248053	Vistara	UK-657	Bangalore	Morning	1	Evening	Delhi	Business	8.33	42	45097	45097	45306.664
208521	Air_India	AI-481	Delhi	Morning	1	Night	Mumbai	Business	14.75	19	29776	29776	29776.000
219950	Vistara	UK-859	Delhi	Morning	0	Afternoon	Hyderabad	Business	2.17	17	24056	24056	24110.016

Fig.13: The above fig talks about the comparison of the actual price and predicted price of the flight

3. **Feature Importance:**

The feature importance is calculated for the tuned random forest model with 125 estimators and 10 features. The class attribute has more importance with a value of 0.88 as compared with other features. Also, the 'stops' attribute on which we have performed some analysis like converting the data type and assigning the values doesn't have much importance in our analysis. This can be understood from its value of 0.00.



	Feature	Importance
7	class	0.88
8	duration	0.05
1	flight	0.03
9	days_left	0.02
2	source_city	0.01
6	destination_city	0.01
0	airline	0.00
3	departure_time	0.00
4	stops	0.00
5	arrival_time	0.00

Fig.14: The above fig informs about the importance of the features which helps in predicting the price of the flight

IV. CONCLUSION

In comparison to the hyperparameter tuned Random Forest Regressors, the baseline model is the least effective because it only accounts for 98.66% (R2 Score) of the variation in flight price. Hence, we can infer that Random Forest Regressor gives a high R2 score with 125 estimators and 10 features (Hyperparameter tuning) out of all the models. However, if we consider the fitting time of the models, the Bagging regressor with 13.36 seconds is far better than the Random Forest model with 133.79 seconds. Also, there is only a difference of 0.003956 in the R2 scores of both the regressors and the RMSE is slightly higher for Bagging Regressor. If we want a model that fits fast and makes predictions like the best model, we can choose Bagging else tuned Random Forest model is preferred.

REFERENCES

1. R. R. Subramanian, M. S. Murali, B. Deepak, P. Deepak, H. N. Reddy and R. R. Sudharsan, "Airline Fare Prediction Using Machine Learning Algorithms," *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2022, pp. 877-884, doi: 10.1109/ICSSIT53264.2022.9716563.
2. K. Tziridis, T. Kalampokas, G. A. Papakostas and K. I. Diamantaras, "Airfare prices prediction using machine learning techniques," *2017 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 1036-1039, doi: 10.23919/EUSIPCO.2017.8081365.
3. T. Wang *et al*., "A Framework for Airfare Price Prediction: A Machine Learning Approach," *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*, 2019, pp. 200-207, doi: 10.1109/IRI.2019.00041.
4. Wang, Tianyi & Pouyanfar, Samira & Tian, Haiman & Tao, Yudong & Alonso, Miguel & Luis, Steven & Chen, Shu-Ching. (2019). A Framework for Airfare Price Prediction: A Machine Learning Approach. 200-207. 10.1109/IRI.2019.00041.
5. Y. S. Can, K. Büyükoğuz, E. B. Giritli, M. Şişik and F. Alagöz, "Predicting Airfare Price Using Machine Learning Techniques: A Case Study for Turkish Touristic Cities," *2022 30th Signal Processing and Communications Applications Conference (SIU)*, 2022, pp. 1-4, doi: 10.1109/SIU55565.2022.9864692.

6. S. N. Prasath, S. Kumar M and S. Eliyas, "A Prediction of Flight Fare Using K-Nearest Neighbors," 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), 2022, pp. 1347-1351, doi: 10.1109/ICACITE53722.2022.9823876.