PRML(CSL2030) Bonus Project
Predict the Flight Ticket Price

Khushi Parikh
(B20EE029)

Abstract

The paper reports my experience with building a regressor model that helps to predict the price of an airplane ticket. Various regression models are used and their results are compared in this report. A web application has also been made where you can customize the inputs and predict the price of the flight.

Introduction

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. The goal of this project is to use the given data to predict the price of a plane ticket with custom inputs. The inputs taken are date of departure and arrival, source, destination, number of stops and which airline you want to travel in. This can be used by us in case we want to predict the price of a journey in the upcoming future. It can also help us compare the differences in prices when changing the parameters, for example for the same places and dates, we can change the airways used.

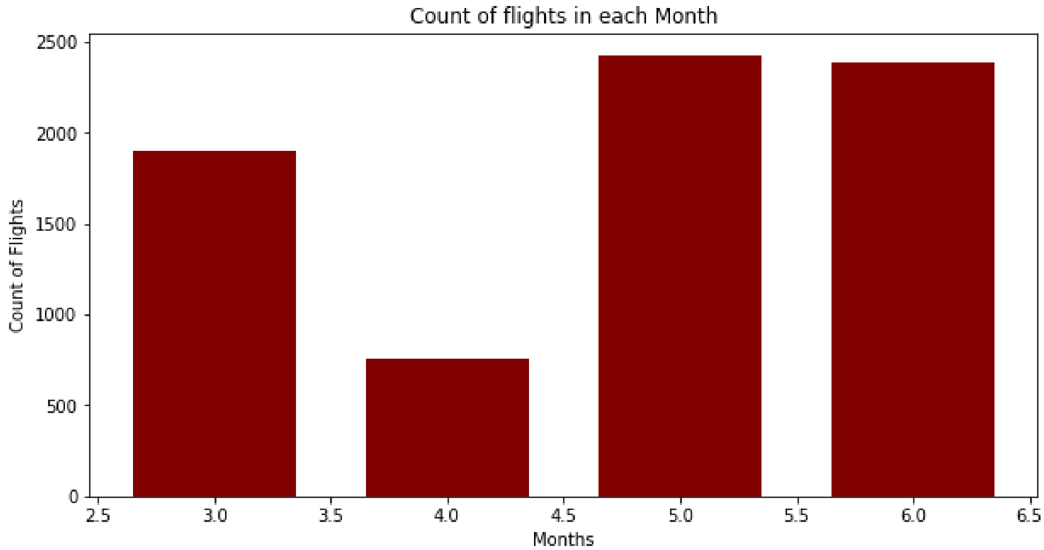
Pre-processing

- Head of the dataframe -

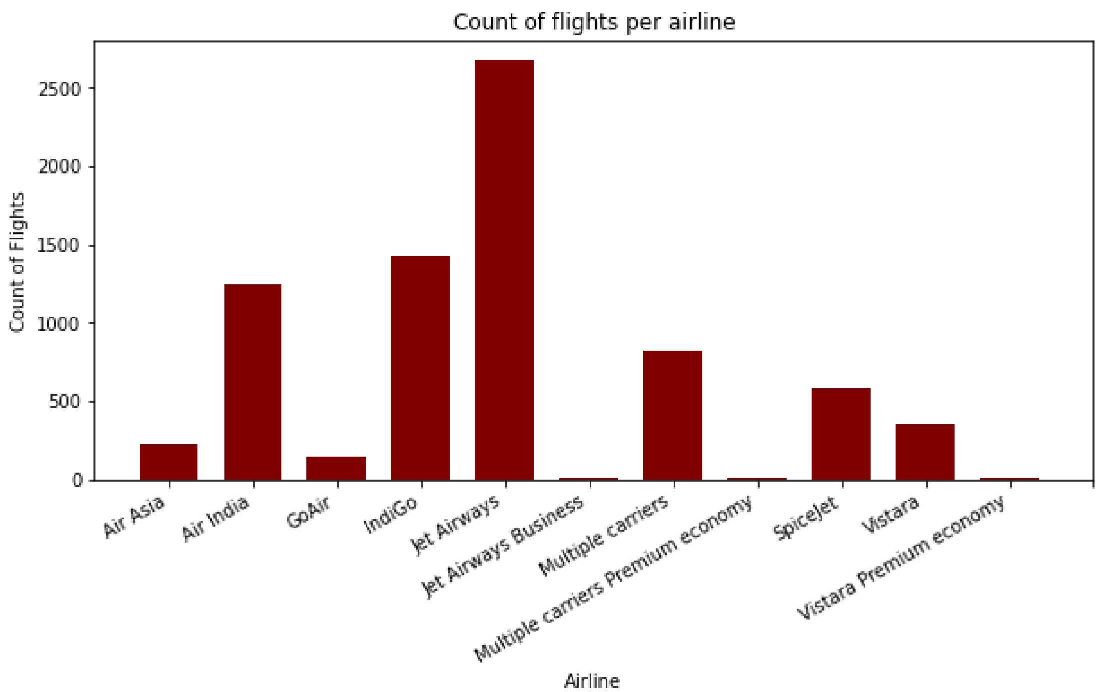
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

- The data frame consists of 10683 rows and 11 columns.
- Rows with null values were dropped as there was only one row with two missing values. Columns such as date of journey, departure time and arrival time were split into their respective two columns to give integer values. “Total stops” was encoded based on the number of stops. The duration in hours and minutes was converted to minutes only.
- Most of the values of “Additional Info” had no information and hence the column was dropped. The “Route” variable was a mixture of source, destination and number of stops and hence was dropped.
- Airline, source and destination were label encoded as integers are easier to interpret and visualize.

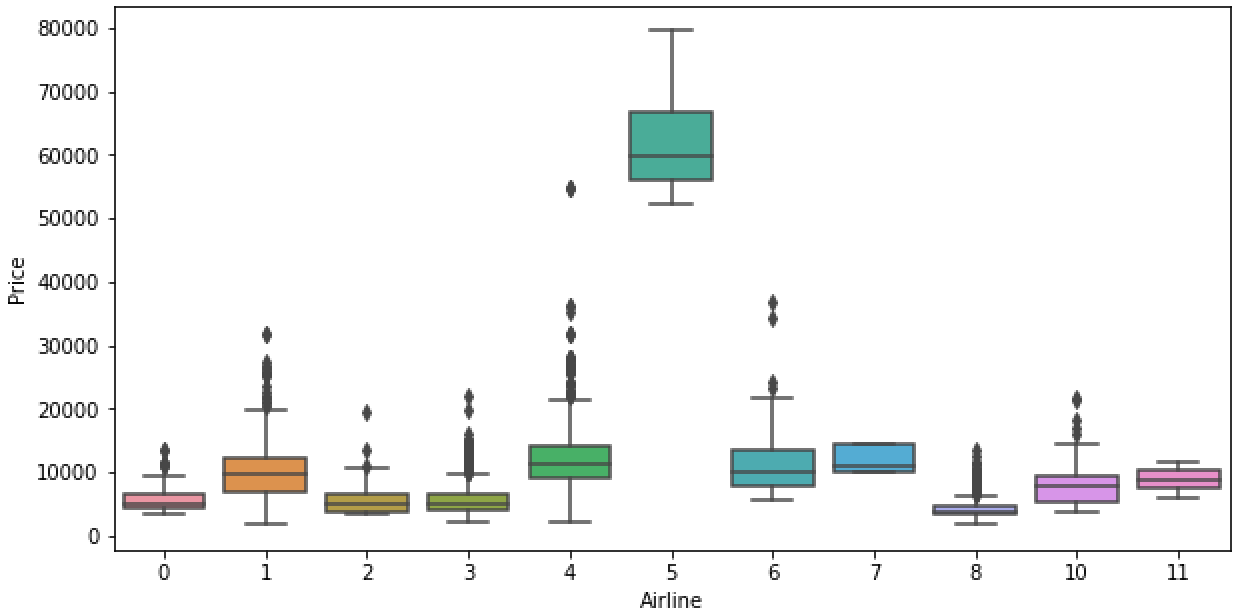
- Maximum flights took place in the latter months out of the months concerned.



- Flights flown were mainly of Jet Airways -



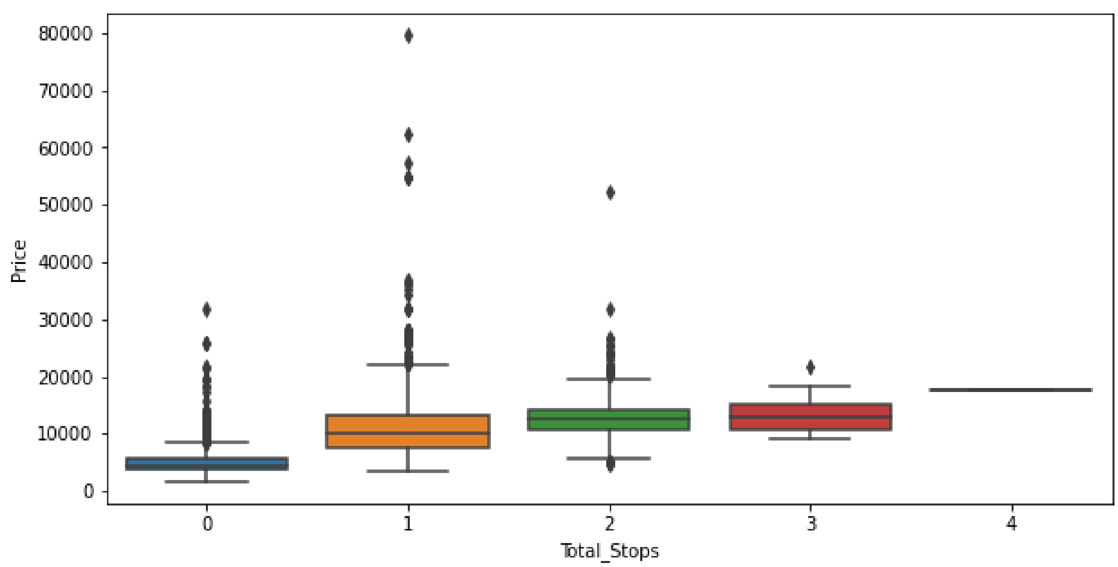
- Price of Jet Airways Business was very high compared to other airlines -



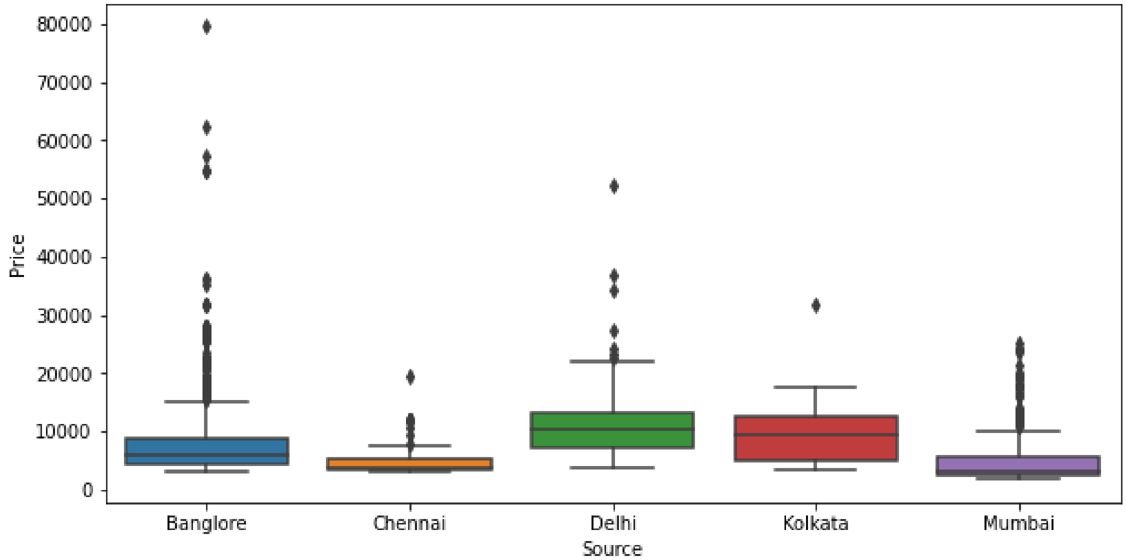

```
encoded_var[0][5]
```

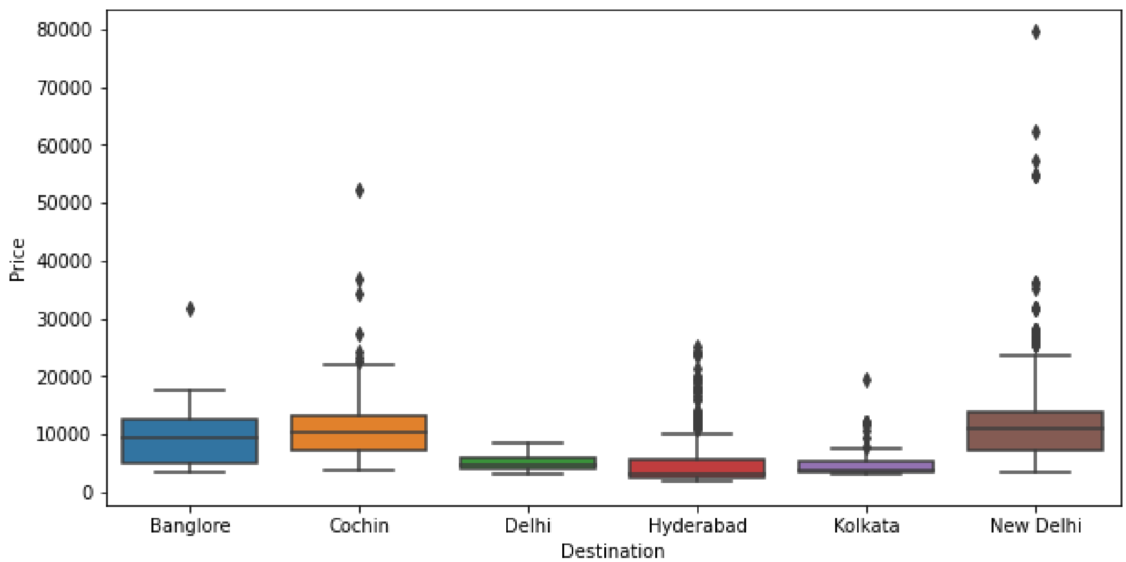
```
'Jet Airways Business'
```

- As the number of stops increased, the mean price increased -

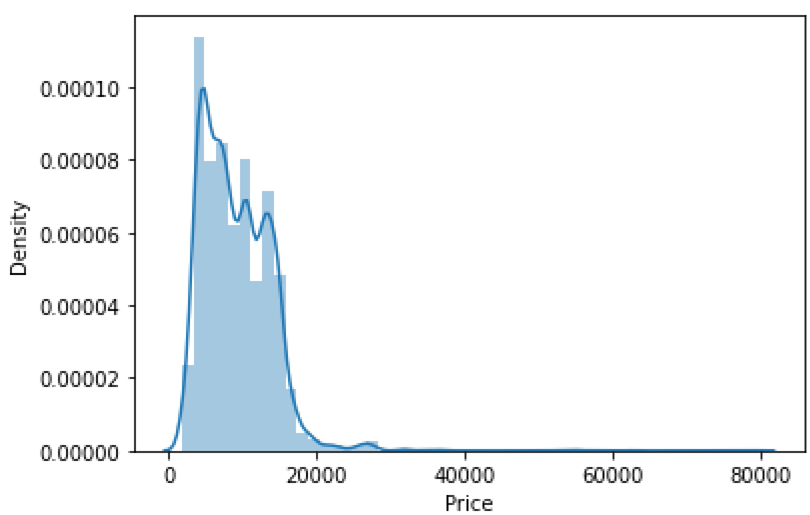


- Price varied with source and destination as follows -





- Variation of Price variable was as follows -



Model Training and Evaluation

Regression models(without hyperparameter tuning) were trained and evaluated based on

- **mean square error(MSE)**
- **mean absolute error(MAE)**
- **r2 score.**

Models -

1) **K-Nearest Neighbors**

```
Train Results for KNN Regressor Model:
Root mean squared error:  2561.2001160151035
Mean absolute error:  1471.676260532299
R-squared:  0.6993038157602993

--------------------------------------------------
Test Results for KNN Regressor Model:
Root mean squared error:  3038.8492221021475
Mean absolute error:  1865.077940717629
R-squared:  0.5377826280586413
```

2) **Decision Tree Regressor**

```
Train Results for Decision Tree Regressor Model:
Root mean squared error:  1353.5637064565267
Mean absolute error:  842.742559313752
R-squared:  0.9160156639204329

--------------------------------------------------
Test Results for Decision Tree Regressor Model:
Root mean squared error:  2258.522945485507
Mean absolute error:  1309.4443381158821
R-squared:  0.7446846221064345
```

3) **Random Forest Regressor**

```
Train Results for Random Forest Regressor Model:
Root mean squared error:  995.7027666859785
Mean absolute error:  549.5638501302336
R-squared:  0.9545535128827408

--------------------------------------------------
Test Results for Random Forest Regressor Model:
Root mean squared error:  2084.626150176648
Mean absolute error:  1222.2300741726553
R-squared:  0.7824874488955754
```

4) XGBoost Regressor

```
Train Results for XGBoost Regressor Model:
Root mean squared error:  1515.474399207406
Mean absolute error:  1051.2206340329092
R-squared:  0.8947218850956267
--------------------------------------------------
Test Results for XGBoost Regressor Model:
Root mean squared error:  1851.2682405788864
Mean absolute error:  1268.7985933919779
R-squared:  0.8284595043418619
```

5) Neural Network Regressor

```
Train Results for NN Regressor Model:
Root mean squared error:  3637.595973223196
Mean absolute error:  2466.557413556787
R-squared:  0.3934455234823193
--------------------------------------------------
Test Results for NN Regressor Model:
Root mean squared error:  3477.7719319633484
Mean absolute error:  2445.397856394782
R-squared:  0.3946170915896242
```

Evaluation/Comparison of models

Comparing the scores on these models, we can see that Random Forest and XGBoost work the best -

	Score	KNN_train	KNN_test	DecTree_train	DecTree_test	RF_train	RF_test	XGB_train	XGB_test	NN_train	NN_test
0	MSE	2561.200116	3038.849222	1353.544370	2337.268604	993.820613	2080.734755	3406.557993	3266.126749	3637.595973	3477.771932
1	MAE	1471.676261	1865.077941	842.686066	1329.123886	550.018774	1221.471517	2368.585645	2370.835908	2466.557414	2445.397856
2	R2 Score	0.699304	0.537783	0.916018	0.726571	0.954725	0.783299	0.468048	0.466058	0.393446	0.394617

Hyperparameter tuning using GridSearchCV

1) **Random Forest Regressor -**

Following parameters were chosen -

```
params = {'n_estimators' : [100, 200, 300, 400, 500, 800, 1000],
          'max_depth' : [i for i in range(10,110,10)],
          'min_samples_leaf' : [1,2,4],
          'min_samples_split': [2, 5, 10]
          }
```

Best fit -

```
RandomForestRegressor(max_depth=70, min_samples_leaf=2, min_samples_split=5,
                      n_estimators=800)
```

Predictions -

```
Train Results for Random Forest Regressor Model:
Root mean squared error:  1359.8611960632154
Mean absolute error:  734.280833566593
R-squared:  0.9152323674722829

--------------------------------------------------
Test Results for Random Forest Regressor Model:
Root mean squared error:  2009.263468390968
Mean absolute error:  1187.8700673736578
R-squared:  0.7979300492597732
```

2) **XGBoost Regressor**

Following parameters were chosen -

```
tuned_params = {'max_depth': [1, 2, 3, 4, 5, 6, 8, 10, 12],
                'learning_rate': [0.01, 0.05, 0.1],
                'n_estimators': [100, 200, 300, 400, 500],
                'reg_lambda': [0.001, 0.1, 1.0, 10.0, 100.0]
                }
```

Best fit -

```
XGBRegressor(learning_rate=0.01, max_depth=12, n_estimators=500, reg_lambda=1.0)
```

Predictions -

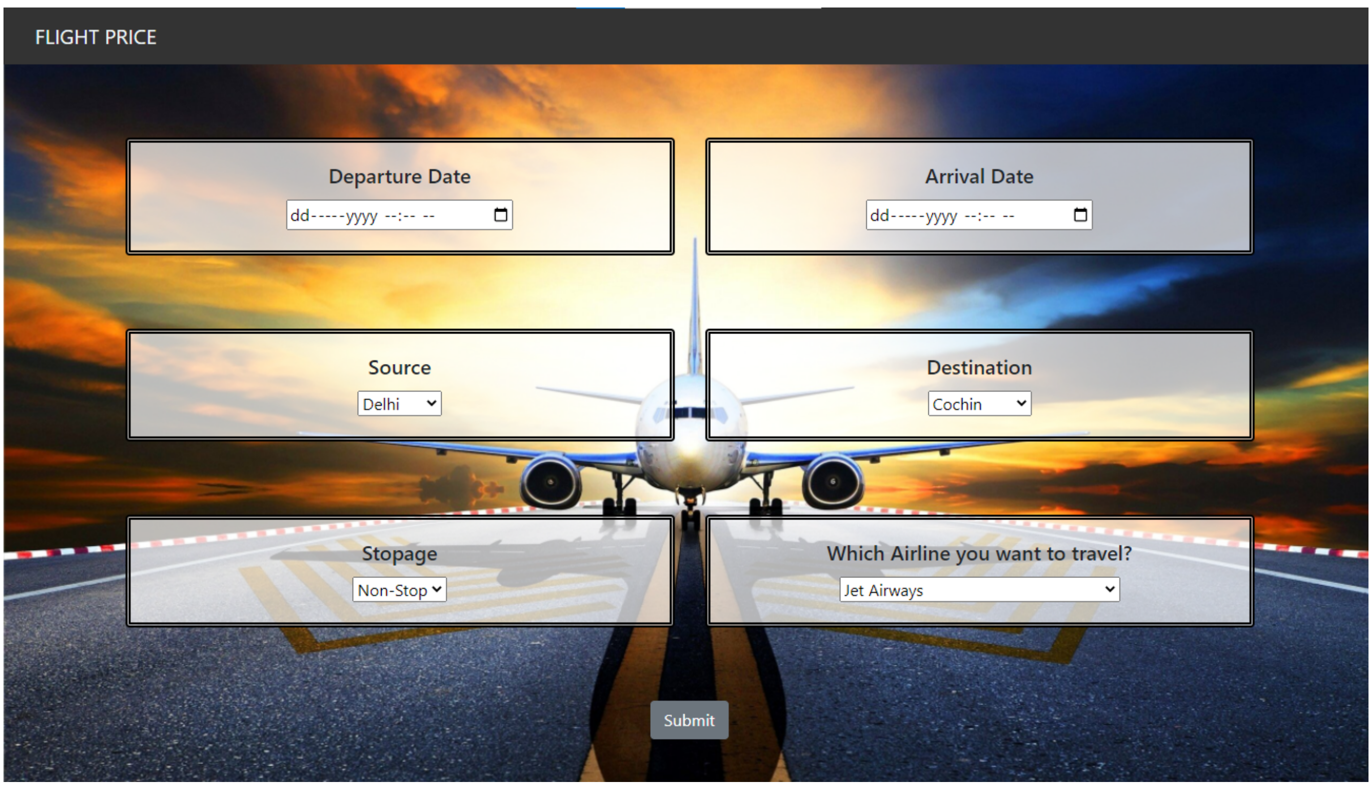
```
Train Results for XGBoost Regressor Model:
Root mean squared error:  961.1592041496634
Mean absolute error:  602.273569888413
R-squared:  0.9576521320987835
--------------------------------------------------
Test Results for XGBoost Regressor Model:
Root mean squared error:  1869.8398220542344
Mean absolute error:  1157.1595379000707
R-squared:  0.8250005157508882
```

Results and Analysis

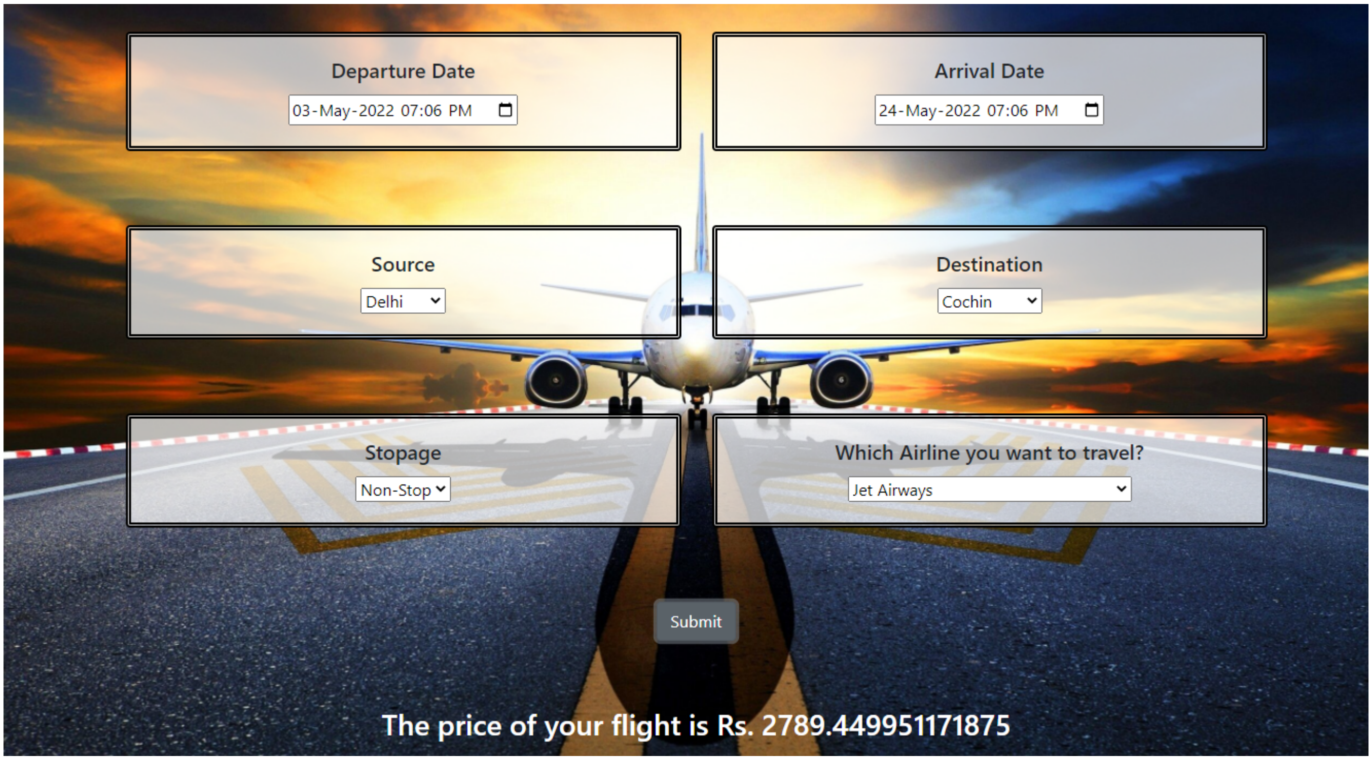
XGBoost model has a very high training accuracy and a higher testing accuracy compared to other models. The root mean square error and the mean absolute error are lesser compared to other models, which means that it performs better. The r-squared score shows the proportion of the variation in the dependent variable that is predictable from the independent variable. For example, if the r-squared score is 0.50, then 50% of the variability of the output can be explained by the model. Higher the r-squared score, better the model performs.

Bonus Web App

A **web application** was also deployed on Heroku using Flask as follows -



You may **customize the inputs** and the price will be predicted after clicking submit as follows -



The model was exported to a web application using pickle library. The inputs of the form were then and pre-processed sent to this model. Preprocessing of each of the fields was done so that the inputs match the columns of the training data. The inputs were then predicted by the model and output was shown on the screen. It is deployed at :
https://flight-price-predictor-prml.herokuapp.com/

Contribution

This project was done individually by Khushi Parikh (B20EE029) . This includes making an end-to-end machine learning pipeline, making a report, learning and deploying the web app using Flask and Heroku. Various references were taken from the internet.