FLIP ROBO

Flight Price Prediction Project

Submitted By:
SYED MUGERA BILAL
syedbilalarmaan@gmail.com

ACKNOWLEDGMENT

Working on this project ***"Flight Price Prediction Project."*** Was a source of immense knowledge to me. We would like to express my sincere gratitude to FlipRobo Technologies for guidance and valuable support throughout this project work.

We acknowledge with a deep sense of gratitude, the encouragement and inspiration received from our SME/Mentor. We would also like to thank our parents for their love and support.

–SYED MUGERA BILAL

CONTENTS

About the project:

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on –

1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases) So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

Business Goal

Data Collection Phase

You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data better the model

In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable

price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

Data Analysis Phase

After cleaning the data, you have to do some analysis on the data.

Do airfares change frequently?

Do they move in small increments or in large jumps?

Do they tend to go up or down over time?

What is the best time to buy so that the consumer can save the most by taking the least risk?

Does price increase as we get near to departure date?

Is Indigo cheaper than Jet Airways?

Are morning flights expensive?

Model Building Phase

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like.

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

Data source and their format:

We collected the data from website https://www.en.kayak.sa/flights/.The data is scrapped using Web scraping technique and the framework used is Selenium.

We scrapped nearly 1848 rows of the data and saved as csv file.

Hardware, Software and Tools:

- ❖ For doing this project, we require laptop with high configuration and specification with a stable Internet connection.

- ❖ Microsoft office, Anaconda distribution as software.

- ❖ Python 3.x as programming language.

- ❖ Jupyter Notebook as Editor which is in Anaconda navigator.

- ❖ Some tools or libraries required like Numpy – used to numerical calculations. Pandas – used to data manipulation. Matplotlib and Seaborn – used to data visualization.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings('ignore')

In [2]: # Loading Dataset
        data = pd.read_csv('df_flight.csv')
        data
```

Data Analysis:

```
In [5]: # Checking dimension of dataset
        data.shape

Out[5]: (1848, 7)
```

The dataset has 1848 rows and 7 columns.

```
In [6]: # Checking missing values
        data.isnull().sum()

Out[6]: Airline        0
        Source         0
        Destination    0
        Duration       0
        Total stops    0
        Price          0
        Date           0
        dtype: int64
```

We have seen that, there is no missing value.

Data Cleaning:

- ❖ Removing word from object data and converting them into int data type.
- ❖ Removing ‘,’ , ‘ ’ replacing missing data .
- ❖ Removing unnecessary features.

```
In [11]: data['Price'] = data['Price'].str.replace('SAR','',regex=True)
         data['Price'] = data['Price'].str.replace(',','',regex=True)

In [12]: data['Price'] = data['Price'].astype(int)
```

```
In [14]: data['Date'] = pd.to_datetime(data['Date'])
```

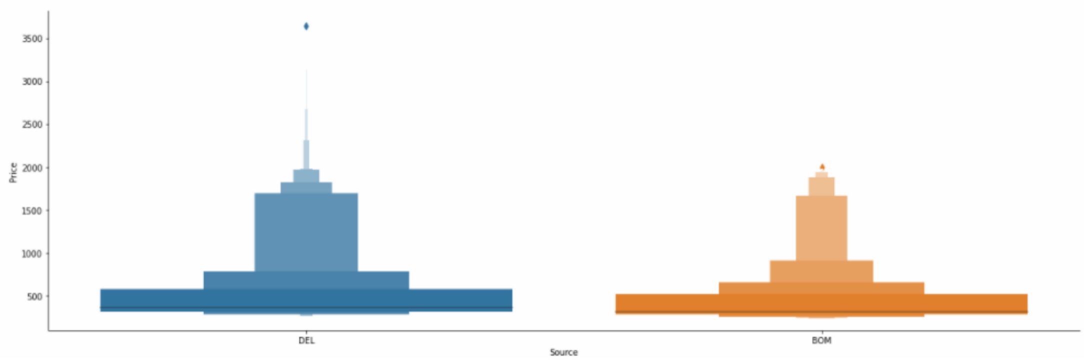
```
In [15]: data.dtypes
```

```
Out[15]: Airline                object
         Source                 object
         Destination            object
         Duration               object
         Total stops            object
         Price                   int32
         Date           datetime64[ns]
         dtype: object
```

Exploratory Data Analysis (EDA):

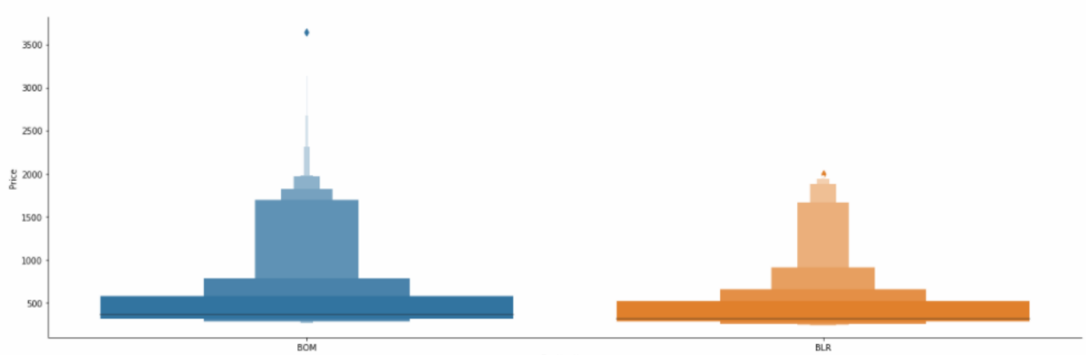
```
In [16]: # source vs price
         sns.catplot(y = "Price", x= "Source", data = data.sort_values("Price", ascending = False), kind="boxen", height = 6, aspect = 3)
```

```
Out[16]: <seaborn.axisgrid.FacetGrid at 0x172ff867608>
```




```
In [17]: # source vs price
         sns.catplot(y = "Price", x= "Destination", data = data.sort_values("Price", ascending = False), kind="boxen", height = 6, aspect
```

```
Out[17]: <seaborn.axisgrid.FacetGrid at 0x1728131b488>
```



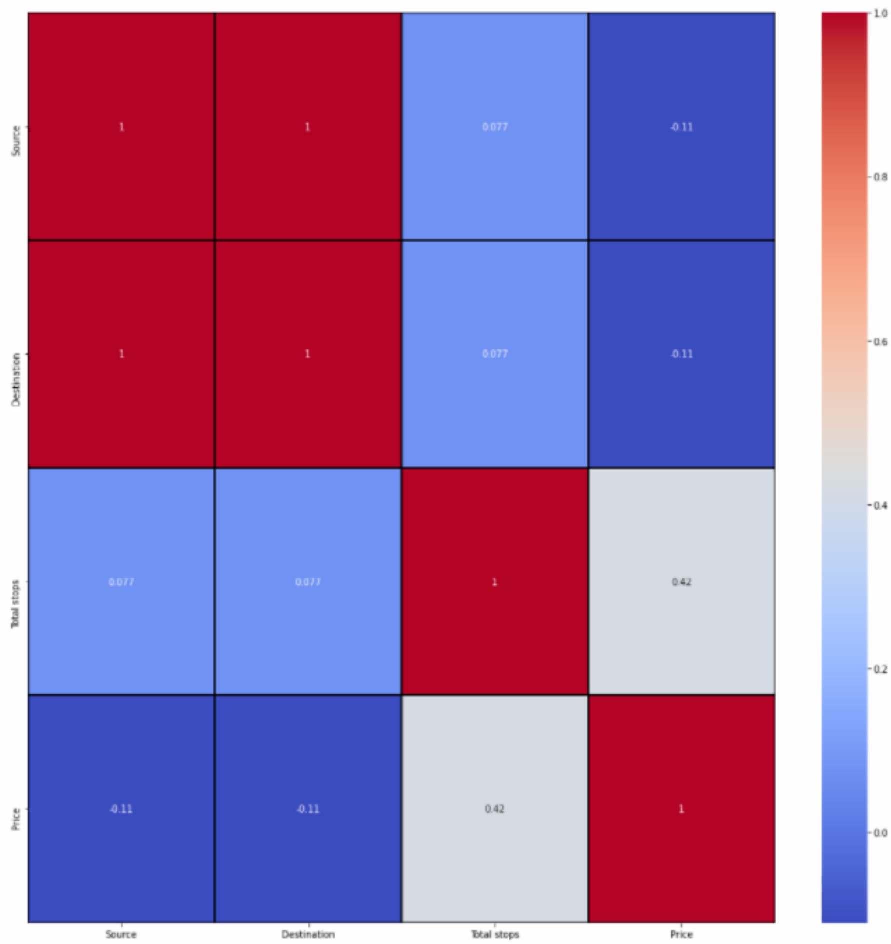
Multivariate Analysis

In [22]:
```
# Correlation of dataset
corr = data.corr()
corr
```

Out[22]:

	Source	Destination	Total stops	Price
Source	1.000000	1.000000	0.077416	-0.110924
Destination	1.000000	1.000000	0.077416	-0.110924
Total stops	0.077416	0.077416	1.000000	0.417396
Price	-0.110924	-0.110924	0.417396	1.000000

In [23]:
```
# Heat map
plt.figure(figsize = (18,18))
sns.heatmap(data.corr(),annot= True, cmap = "coolwarm", linewidths=2, linecolor='black')
plt.show()
```



Splitting the data:

Using scikit learn library of python we have imported train test split to divide data into train data and test data. We have use test size is 0.2 (20%), rest of 0.8 (80%) data as train data. Random state provides us a seed to the random number generator by using following codes.

from sklearn.model_selection import train_train_split
x_train, x_test, y_train, y_test = train_train_split (x, y, test_size=0.2, random_state=45)

```
In [31]: # Creating train_test_split using best random_state
         x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=45,test_size=.20)
```

Building Machine Learning Algorithms:

The following classifier algorithms we used are:

- Linear Regression,
- Lasso Regression
- Ridge Regression
- ElasticNet
- Decision Tree Regressor
- KNeighbors Regressor
- Random Forest Regressor,
- AdaBoost Regressor
- Gradient Boosting Regressor

```
In [32]: #Importing the algorithms and other parameters
         from sklearn.linear_model import LinearRegression,Lasso,Ridge,ElasticNet
         from sklearn.svm import SVR
         from sklearn.neighbors import KNeighborsRegressor
         from sklearn.tree import DecisionTreeRegressor
```

```
In [33]: LR=LinearRegression()
         l=Lasso()
         en=ElasticNet()
         rd=Ridge()
         svr=SVR()
         dtr=DecisionTreeRegressor()
         knr=KNeighborsRegressor()
```

Evaluation of ML Models:

- As you can see , I had called the algorithms, then I called the empty list with the name models [ ], and calling all the model one by one and storing the result in that.

```
In [34]: models= []
         models.append(('Linear Regression',LR))
         models.append(('Lasso Regression',l))
         models.append(('Elastic Net Regression',en))
         models.append(('Ridge Regression',rd))
         models.append(('Support Vector Regressor',svr))
         models.append(('Decision Tree Regressor',dtr))
         models.append(('KNeighbors Regressor',knr))
```

```
In [36]: #Finding the required metrices for all models together using a for loop
         Model=[]
         score=[]
         cvs=[]
         sd=[]
         mae=[]
         mse=[]
         rmse=[]
         for name,model in models:
             print('******************************',name,'**************************')
             print('\n')
             Model.append(name)
             model.fit(x_train,y_train)
             print(model)
             pre=model.predict(x_test)
             print('\n')
             AS=r2_score(y_test,pre)
             print('r2_score: ',AS)
             score.append(AS*100)
             print('\n')
             sc=cross_val_score(model,x,y,cv=5,scoring='r2').mean()
             print('cross_val_score: ',sc)
             cvs.append(sc*100)
             print('\n')
             std=cross_val_score(model,x,y,cv=5,scoring='r2').std()
             print('Standard Deviation: ',std)
             sd.append(std)
             print('\n')
             MAE=mean_absolute_error(y_test,pre)
             print('Mean Absolute Error: ',MAE)
             mae.append(MAE)
             print('\n')
             MSE=mean_squared_error(y_test,pre)
             print('Mean Squared Error: ',MSE)
             mse.append(MSE)
             print('\n')
             RMSE=np.sqrt(mean_squared_error(y_test,pre))
             print('Root Mean Squared Error: ',RMSE)
             rmse.append(RMSE)
             print('\n\n')
```

As you can observe, I made a for loop and called all the algorithms one by one and appending their result to models.

Let me show the output so that we can glance the result in more appropriate way.

Random Forest Regressor

In [38]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor(random_state=45)   #Using the best random state we obtained
parameters={'n_estimators':[10,50,100,500]}
grid=GridSearchCV(rfr,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)  #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'n_estimators': 500}
0.17860490876552895
```

In [39]:
```
#Using the best parameters obtained
RF=RandomForestRegressor(random_state=45, n_estimators=500)
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(RF,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(RF,x,y,cv=5,scoring='r2').std())
print('\n')
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:  24.00320074815504
Cross validation score:  18.526219902836218
Standard deviation:  0.029231520728342533


Mean absolute error:  189.64271752303043
Mean squared error:  127384.65526241872
Root Mean squared error:  356.9098699425651
```

AdaBoost Regressor

In [40]:
```
from sklearn.ensemble import AdaBoostRegressor
adr=AdaBoostRegressor(random_state=45)   #Using the best random state we obtained
parameters={'n_estimators':[10,50,100,500,1000],'learning_rate':[0.001,0.01,0.1,1],'loss':['linear','square']}
grid=GridSearchCV(adr,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)  #Printing the best parameters obtained
print(grid.best_score_) #Mean cross-validated score of best_estimator
```

```
{'learning_rate': 0.001, 'loss': 'square', 'n_estimators': 10}
0.17969291732861664
```

In [41]:
```
#Using the best parameters obtained
adr=AdaBoostRegressor(random_state=45, n_estimators=10, learning_rate=0.001, loss='square')
adr.fit(x_train,y_train)
pred=adr.predict(x_test)
print("r2_score: ",r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(adr,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(adr,x,y,cv=5,scoring='r2').std())
print('\n')
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:  24.154324757773715
Cross validation score:  18.327966451471916
Standard deviation:  0.030664556146517235


Mean absolute error:  189.35788847630187
Mean squared error:  127131.34354328494
Root Mean squared error:  356.55482543822757
```

Gradient Boosting Regressor

```
In [42]: from sklearn.ensemble import GradientBoostingRegressor
         gbr=GradientBoostingRegressor(random_state=45)   #Using the best random state we obtained
         parameters={'n_estimators':[10,50,100,500,1000]}
         grid=GridSearchCV(gbr,parameters,cv=5,scoring='r2')
         grid.fit(x_train,y_train)
         print(grid.best_params_)  #Printing the best parameters obtained
         print(grid.best_score_) #Mean cross-validated score of best_estimator

         {'n_estimators': 50}
         0.17906121566431718
```

```
In [43]: #Using the best parameters obtained
         gbr=GradientBoostingRegressor(random_state=45, n_estimators=50)
         gbr.fit(x_train,y_train)
         pred=gbr.predict(x_test)
         print("r2_score: ",r2_score(y_test,pred)*100)
         print('Cross validation score: ',cross_val_score(gbr,x,y,cv=5,scoring='r2').mean()*100)
         print('Standard deviation: ',cross_val_score(gbr,x,y,cv=5,scoring='r2').std())
         print('\n')
         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:  23.9744123114883
         Cross validation score:  18.53543091791307
         Standard deviation:  0.02944319735328972


         Mean absolute error:  189.67122771887713
         Mean squared error:  127432.90999309735
         Root Mean squared error:  356.9774642650392
```

Key Metrics for success in solving problem under consideration:

- r2_score
- Mean Absolute Error (MAE)
- Mean Square Error (MSE)
- Root Mean Square Error (RMSE)
- Cross-validation
- Hyperparameter Tuning using GridSearchCV

Result of ML Models:

We saved result of ML Models in DataFrame.

```
In [37]: #Finalizing the result
result=pd.DataFrame({'Model':Model, 'r2_score': score, 'Cross_val_score':cvs, 'Standard_deviation':sd,
                     'Mean_absolute_error':mae, 'Mean_squared_error':mse, 'Root_Mean_Squared_error':rmse})
result
```

Out[37]:

	Model	r2_score	Cross_val_score	Standard_deviation	Mean_absolute_error	Mean_squared_error	Root_Mean_Squared_error
0	Linear Regression	23.654321	17.565568	0.033797	194.821587	127969.441845	357.728168
1	Lasso Regression	23.593495	17.600782	0.032821	194.365116	128071.396966	357.870643
2	Elastic Net Regression	11.515149	9.264821	0.012982	231.232004	148316.934102	385.119376
3	Ridge Regression	23.634759	17.598175	0.033219	194.810450	128002.231897	357.773996
4	Support Vector Regressor	-3.382130	-4.287826	0.057896	191.100593	173287.522285	416.278179
5	Decision Tree Regressor	24.009448	18.528796	0.029293	189.520256	127374.184435	356.895201
6	KNeighbors Regressor	25.462091	-26.090340	0.509888	192.296216	124939.285838	353.467517

We can see that Random Forest Regressor and KNeighbors Regressor are performing well compared to other algorithms. Now we will try Hyperparameter Tuning to find out the best parameters and try to increase the scores.

Hyperparameter Tuning:

In order to increase the accuracy score of the model we use hyperparameter tuning of the best model in order to find best parameters by using GridSearchCV() .

Hyperparameter Tuning

```
In [44]: # Creating parameter list to pass in GridSearchCV
parameters={'criterion':['mse','mae'],'n_estimators':[50,100,500],'max_features':['auto','sqrt','log2']}
```

```
In [45]: # Using GridSearchCV to run the parameters and checking final accuracy
rf=RandomForestRegressor()
grid=GridSearchCV(rf,parameters,cv=5,scoring='r2')
grid.fit(x_train,y_train)
print(grid.best_params_)
print(grid.best_score_)

{'criterion': 'mse', 'max_features': 'auto', 'n_estimators': 50}
0.17945916883561358
```

```
In [46]: # Using the best parameters obtained
RF=RandomForestRegressor(random_state=45, n_estimators=50, criterion='mse', max_features='auto')
RF.fit(x_train,y_train)
pred=RF.predict(x_test)
print('r2_score: ',r2_score(y_test,pred)*100)
print('Cross validation score: ',cross_val_score(RF,x,y,cv=5,scoring='r2').mean()*100)
print('Standard deviation: ',cross_val_score(RF,x,y,cv=5,scoring='r2').std())
print('\n')
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:  23.959960836089834
Cross validation score:  18.46870003501898
Standard deviation:  0.028778886504331264


Mean absolute error:  189.6600444912149
Mean squared error:  127457.13333184042
Root Mean squared error:  357.0113910393342
```

After applying Hyperparameter Tuning, we can see that RandomForestRegressor improve acuuracy r2_score slightly decreased. Now we will finalize the model.

Saving the model

In order to dump the model which we have developed so that we can use it to make predictions in future, we have saved or dumped the best model.

Finalizing the model

```
In [47]: rf_prediction=RF.predict(x)
         print('Predictions of Random Forest Regressor: ',rf_prediction)

         Predictions of Random Forest Regressor:  [359.67618147 359.67618147 359.67618147 ... 544.06711797 544.06711797
          544.06711797]

In [48]: # Saving the model
         import pickle
         filename='Flight_Price_Project.pkl'   #Specifying the filename
         pickle.dump(RF,open(filename,'wb'))

In [49]: # Saving the predicted values
         results=pd.DataFrame(rf_prediction)
         results.to_csv('Flight_Price_Prediction_Results.csv')
```

Conclusion:

- After web scraping from *https://www.en.kayak.sa/flights/* (using selenium) we making a dataset in csv format.
- The target variable column ['Price'] in SAR.
- First, we loaded the dataset and did the EDA process and other pre-processing techniques checking and filling the missing data, converting categorical data

into datetime format and numerical data, visualizing the of data, etc.

- Then we did the model training, building the model and finding out the best model on the basis of different metrics scores we got like r2_score, Mean Absolute Error (MAE), Mean Square Error (MSE), Root Mean Square Error (RMSE), Cross-validation

- We used algorithms like Linear Regression, Lasso Regression, Ridge Regression, ElasticNet, Decision Tree Regressor, KNeighbors Regressor, Random Forest Regressor, AdaBoost Regressor, Gradient Boosting Regressor .

- After performing the analysis, we got KNeighbors Regressor and Random Forest Regressor algorithm as the best algorithms among all. After that finding out the best parameter and improving the scores, we performed Hyperparameter Tuning.

- The problem while doing Hyperparameter Tuning is that it took nearly 5 hours to fetch the best parameters.
- After Tuning Random Forest Regressor is selected as the final model ,

We finalized the best model we obtained by saving the model in a pkl file.