FLIGHT PRICE PREDICTION PROJECT

Submitted by:

Anita Thapa

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my SME Swati Ma’am for giving me an opportunity to work on the flight price prediction project. During my research on this project internet sites like towardsdatascience, medium, easemytrip, yatra.com, and Stack Overflow helped me understand the main problem statement, also to collect data and in finding their solutions.

INTRODUCTION

- ## Business Problem Framing

  Model to predict flight price so that consumers book flight tickets at the best time and save the most by taking least risk.

- ## Conceptual Background of the Domain Problem

  Airline Industry has picked up a rapid pace in terms of revenue as soon as the covid-19 curbs uplifted from the country. To understand the dynamics of the flight fare changes a model needs to be built. The model will help to analyze the factors which are affecting the rise and fall of the flight fares. The prediction will be useful for consumers to book flight at the best time so that they save the most by taking the least risk.

- ## Review of Literature

  The research was done on Flight prices in various online websites like easemytrip, yatra.com etc. in order to collect the data using the selenium library. Using websites like medium, towardsdatascience and Kaggle helped in modeling the dataset and in predicting the airfare. Stack overflow website helped to resolve problems faced during data scraping and modeling.

- ## Motivation for the Problem Undertaken

  The main objective was to analyse the airfare changes and to model a flight price valuation prediction. In order to understand the airline industry and their price trends this project was taken up so that the consumers can book their tickets at the best time with more saving.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  The data was present in excel format so by using pandas the excel file was read. Then the dataset details were analysed using .shape() to find the number of rows and columns, using dtypes() we found the dataset datatype like object and numeric datatype. Dataset has 4749 Rows and 12 Columns; 2 columns are numeric and rest 10 columns are object type and the Target column is Price. Dataset is a Regression model.

```
#Importing Libraries:
import numpy as np
import pandas as pd
```

```
#Reading excel file and converting it in dataframe:
ds=pd.read_excel("C:\\Users\\Asus\\Desktop\\flights_data.xlsx")
df=pd.DataFrame(ds)
df.head()
```

	Unnamed: 0	Airline	Journey Date	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Flight_no	Price
0	0	GO FIRST	26/02/2022	Delhi	Mumbai	DEL → BOM	19:50	22:10	02h 20m	non-stop	G8- 323	7740
1	1	Indigo	26/02/2022	Delhi	Mumbai	DEL → BOM	20:00	22:15	02h 15m	non-stop	6E-6261	7845
2	2	Vistara	26/02/2022	Delhi	Mumbai	DEL → BOM	19:50	22:00	02h 10m	non-stop	UK-985	8790
3	3	Air India	26/02/2022	Delhi	Mumbai	DEL → BOM	20:00	22:10	02h 10m	non-stop	AI-805	8790
4	4	Indigo	26/02/2022	Delhi	Mumbai	DEL → BOM	21:15	23:30	02h 15m	non-stop	6E-6722	9945

```
# Rows & Columns in  Train dataset:
df.shape
```

```
(4749, 12)
```

```
4749 rows and 12 columns
```

```
# Datatype of dataset

df.dtypes
```

```
Unnamed: 0         int64
Airline           object
Journey Date      object
Source            object
Destination       object
Route             object
Dep_Time          object
Arrival_Time      object
Duration          object
Total_Stops       object
Flight_no         object
Price              int64
dtype: object
```

- Data Sources and their formats

  Data was scraped from websites like easemytrip and yatra.com using Selenium library in Python and saved as excel file. Then data was imported in python using pandas . Information of dataset is found

using datasetname.info(). As per the information each column has count of 4749 rows as shown in below figure.

```
# Information about Dataset:

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4749 entries, 0 to 4748
Data columns (total 12 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Unnamed: 0     4749 non-null   int64 
 1   Airline        4749 non-null   object
 2   Journey Date   4749 non-null   object
 3   Source         4749 non-null   object
 4   Destination    4749 non-null   object
 5   Route          4749 non-null   object
 6   Dep_Time       4749 non-null   object
 7   Arrival_Time   4749 non-null   object
 8   Duration       4749 non-null   object
 9   Total_Stops    4749 non-null   object
 10  Flight_no      4749 non-null   object
 11  Price          4749 non-null   int64 
dtypes: int64(2), object(10)
memory usage: 445.3+ KB
```

- Data Pre-processing Done

Dataset had 12 columns and 4749 rows. Journey Date converted to day and month. Columns named Dep_Time, Arrival_Time and Duration converted into hours and minutes. Repeated values replaced in Airline, Total_Stops and Destination column.

Converting object to datetime datatypes

```
# day separated from date
df["Journey_day"] = pd.to_datetime(df["Journey Date"], format="%d/%m/%Y").dt.day
```

```
# month separated from date
df["Journey_month"] = pd.to_datetime(df["Journey Date"], format ="%d/%m/%Y").dt.month
```

```
df['Journey_month'].value_counts()
```

```
3    2653
4    1185
2     911
Name: Journey_month, dtype: int64
```

DEP_TIME TO HOURS AND MINUTE

```
#departure hour extracted from  departure time.
df["Depart_Hour"] = pd.to_datetime(df["Dep_Time"]).dt.hour

#departure minute extracted from  departure time.
df["Depart_Min"] = pd.to_datetime(df["Dep_Time"]).dt.minute
```

```
# Arrival time is separated into arrival hour and arrival minute:

df["Arrival_hour"] = pd.to_datetime(df.Arrival_Time).dt.hour

df["Arrival_min"] = pd.to_datetime(df.Arrival_Time).dt.minute
```

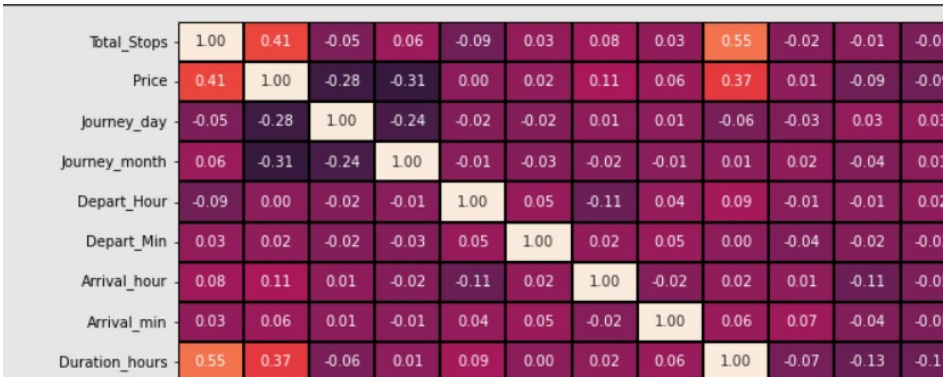
```
# Separating hours and minutes from duration column from the dataset.
duration = list(df["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))
```

```
#replacing repeated values:
df['Airline'].replace({"IndiGo": "Indigo", "GO FIRST":"Go First", "Air Asia": "AirAsia"}, inplace = True)
```

```
df['Destination'].replace({"New Delhi": "Delhi"}, inplace = True)
```

- Data Inputs- Logic- Output Relationships

  Output Column is Price column and rest other columns are Input Columns. Using Correlation we can find out the relationship between Input and output columns. Total_Stops column is positively correlated with Price column.



- State the set of assumptions (if any) related to the problem under consideration

  In Total_Stops column 2stops,3 stops and 4 stops were assumed as 2+ stops

- Hardware and Software Requirements and Tools Used

  Libraries used were:

  1. Numpy : It is a Numerical Python library used for numerical computation like arrays in Python.
  2. Panda: It was used for reading and converting excel/csv file into dataframe.
  3. matplotlib: this library was used for plotting the graph in the EDA.
  4. Seaborn: This library is also used for visualization as it has helped to plot different types of plots like countplot, displot, boxplot and heatmap in the notebook.
  5. Label encoder: It helped to encode the object datatype into numeric datatype as the Machine learning algorithm works on numeric datatype only.

6. sklearn.preprocessing and power transform: It was used to remove the skewness from the dataset.
7. Standard Scaler : It was used to scale the feature column of the dataset before model selection.
8. sklearn: This library was used to import train_test_split, metrics like accuracy score, classification report and also importing Algorithms likes Linear Regression, Decision Tree Regressor, SVR, Random Forest Regressor and K-Neighbors Regressor..
9. pickle: This library was used to save the final model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The Standard Scaled data was used in Model building. As the model is Regressor so Linear Regressor library along with r2 score, cross validation score, means square error metrics were imported. Then 4 Algorithms were used which were Decision Tree Regressor, SVR, Random Forest Regressor and K-Neighbors Regressor. Then after selecting best model, then the final model was deployed and the accuracy of model was 81.9% as it predicted values approximately equal to actual values.

Model Building:

```
# Model selection:

from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

```
for i in range(0,100):
    train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=i)
    lr.fit(train_x,train_y)
    pred_train=lr.predict(train_x)
    pred_test=lr.predict(test_x)
    print(f"At random state {i},the training accuracy is:- {r2_score(train_y,pred_train)}")
    print(f"At random state {i},the testing accuracy is:- {r2_score(test_y,pred_test)}")
    print("\n")
```

- Testing of Identified Approaches (Algorithms)

  Algorithms used for the training and testing are:
  1. XGBoost Regressor
  2. Decision Tree Regressor
  3. SVR
  4. Random Forest Regressor
  5. K-Neighbors Regressor

- Run and Evaluate selected models

  Out of all the 5 algorithm used the best algorithm used is Random Forest Regressor as the accuracy score is more after using hyperparameter.

3. Random forest regressor

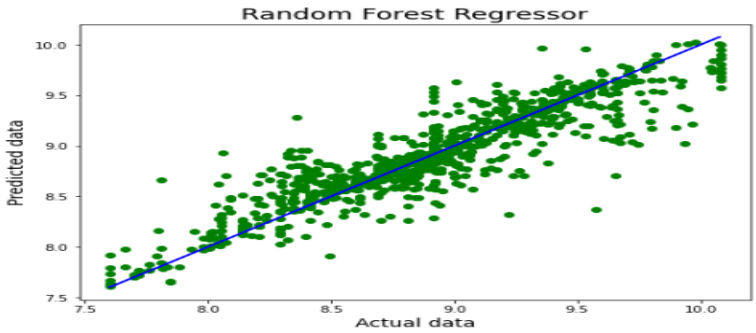
```
# Importing Libraries and Hyper parameter tuning:
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameters = {'criterion':['mse', 'mae'],'bootstrap': [True, False], 'max_depth': [5, 10, None], 'max_features': ['auto'
rf =RandomForestRegressor()
clf = GridSearchCV(rf,parameters,cv =6, n_jobs = -1, verbose = 2,refit = False)
clf.fit(train_x,train_y)

print(clf.best_params_)
```

```
Fitting 6 folds for each of 240 candidates, totalling 1440 fits
{'bootstrap': True, 'criterion': 'mse', 'max_depth': None, 'max_features': 'auto', 'n_estimators': 15}
```

```
rf= RandomForestRegressor(criterion="mse",max_features="auto",max_depth=None,bootstrap=True,n_estimators=15)
rf.fit(train_x,train_y)
rf.score(train_x,train_y)
predrf = rf.predict(test_x)
print('rf score',rf.score(train_x,train_y))
rfs = r2_score(test_y,predrf)
print('R2 Score:',rfs*100)

rfscore = cross_val_score(rf,x,y,cv=6)
rfc = rfscore.mean()
print('Cross Val Score:',rfc*100)
print("Mean squared error:",mean_squared_error(test_y,predrf))
print("Mean absolute error:",mean_absolute_error(test_y,predrf))
print('RMSE:', np.sqrt(mean_squared_error(test_y, predrf)))
```

```
rf score 0.9719211732040074
R2 Score: 81.9609580888492
Cross Val Score: 30.15636213785588
Mean squared error: 0.04764347710058702
Mean absolute error: 0.14673604760568396
RMSE: 0.21827385803294683
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=test_y, y=predrf, color='g')
plt.plot(test_y,test_y, color='b')
plt.xlabel('Actual data',fontsize=14)
plt.ylabel('Predicted data',fontsize=14)
plt.title('Random Forest Regressor',fontsize=18)
plt.show()
```

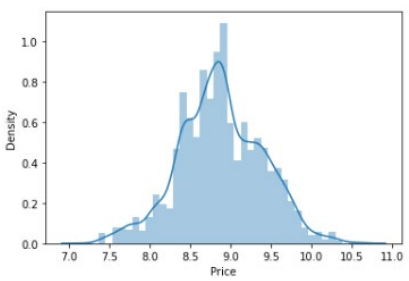


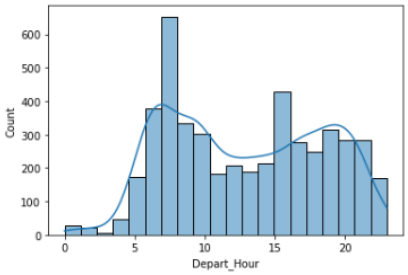
- Key Metrics for success in solving problem under consideration

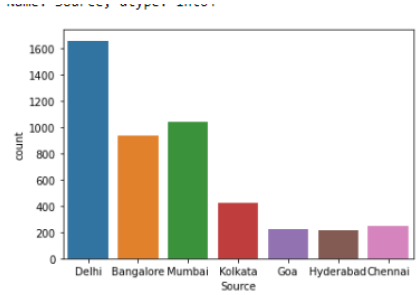
  The main metric used was to find out the accuracy of the model which helped to determine the best model is comparing r2 score with the cross validation score of the algorithm. For model selection the algorithm must have small difference between cross validation score and r2 score then that algorithm is considered to be the best model.

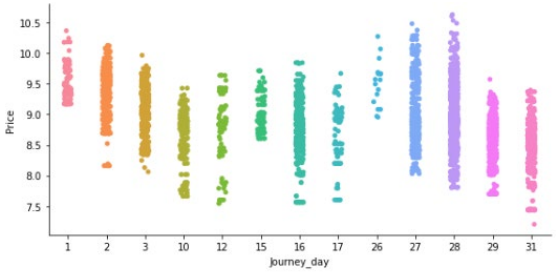
- Visualizations

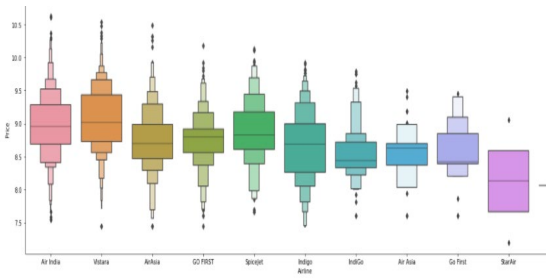
  Different plots were used in the problem like displot, countplot, histplot, heatmap, boxplot and pyplot using matplotlib and seaborn library as shown in the images below.

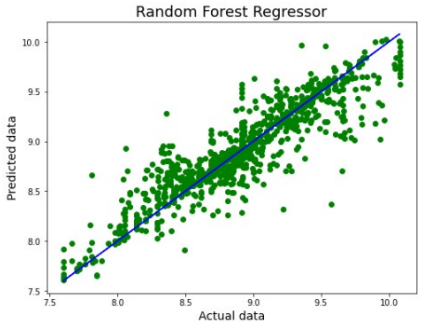


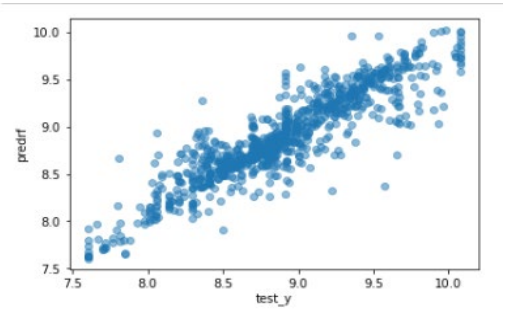












- Interpretation of the Results

  From the visualizations, preprocessing and modeling of the data it was interpreted that morning flights are expensive and the Airline Air India and Vistara are expensive. If stops are more than prices also more. Consumers should purchase tickets during mid of month as they are at less price. As per the model the data can be predicted with 81.9% accuracy.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  Key Findings of the project are:

  1. Flight fare increases with increase in the number of stops.
  2. Flights with duration 5 hours and above have higher prices.
  3. Vistara and Air India are expensive airlines.
  4. Flight fares are high during start of the month but fare is less in the mid and end last two-three days of the month.
  5. Delhi source flights and Goa destination flights have higher prices.

- ## Learning Outcomes of the Study in respect of Data Science

  Using Visualization libraries like matplotlib and seaborn it was easy to find the correlation between the dataset columns and the plotting of each column along the target column helped to analyse the problem pretty well. As visualization helped to get better insight of the data like skewness present in the data.

- ## Limitations of this work and Scope for Future Work

  As the model accuracy is 81.9% so it can approximately predict the correct value as sometimes it might predict wrong values but chances are less.