FLIP ROBO

Flight Price Prediction

Submitted by: Poonam Kaur (Internship 29)

SME Name: Shwetank Mishra

ACKNOWLEDGMENT

I would like to convey my heartfelt gratitude to Flip Robo Technologies for providing me with this wonderful opportunity to work on a Machine Learning project “Flight Price Prediction Model”.

I also want to thank my SME “Shwetank Mishra” for providing the dataset and directions to complete this project.

I would also like to thank my academic “Data Trained Education” and their team who has helped me to learn Machine Learning and how to work on it.

I am very grateful for the team as I learnt a lot during the completion of this project.

INTRODUCTION

- ## Business Problem Framing

We have to work on a project where we have to collect data of flight fares with other features and work to make a model to predict fares of flights.

This project contains three phases:

1) Data Collection Phase
2) Data Analysis
3) Model Building Phase

- ## Conceptual Background of the Domain Problem

Anyone who has booked a flight ticket knows how quickly the prices vary.

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 filing up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

- ## Review of Literature

We have to made flight price prediction model. This project contains three phases:

a) Data Collection phase: Scrapped 1501 rows of data from website Yatra.com. We have fetched data for different airlines.

b) Data Analysis: After cleaning the data, we have to do some analysis on the data.

c) Model Building Phase: After collecting the data, built a machine learning model. Before model building have done all data pre-processing steps. Tried different models with different hyper parameters and selected the best model. Followed the complete life cycle of data science

   Include all steps like:
   1) Data Cleaning
   2) Exploratory Data Analysis
   3) Data Pre-processing
   4) Model Building
   5) Model Evaluation
   6) Selecting the best Model

- Motivation for the Problem Undertaken

  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 flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive pruchases.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem
  1) Scrapped data from Yatra.com
  2) Used Panda's Library to save data into csv file
  3) Cleaned the data
  4) Extracted hidden features
  5) Descriptive Statistics
  6) Detected Skewness
  7) Scaled data
  8) Removed Multicollinearity

- Data Sources and their formats

  Scraped Data from websites: Yatra.com and used panda's library to save data in csv file: flight_price_prediction

- Data Preprocessing Done
-

  a) Checked total number of rows and columns

```
flight.shape
```

```
(1500, 9)
```

  b) Checked all columns name

```
flight.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', '
       'Arrival_Time', 'Duration', 'Total_Stops'
      dtype='object')
```

c) Checked Data type of All Data

```
flight.dtypes
```

```
Airlire            object
Date_of_Journey    object
Source             object
Destiration        object
Dep_Time           object
Arrival_Time       object
Duration           object
Total_Stops        object
Price              object
dtype: object
```

d) Checked for Null Values

```
flight.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Price              0
dtype: int64
```

e) Checking if "-" values present in dataset or not

```
(flight=='-').sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Price              0
dtype: int64
```

f) Checked total number of unique values

```
flight.nunique()
```

```
Airline             6
Date_of_Journey     5
Source              4
Destination         5
Dep_Time           55
Arrival_Time       30
Duration           19
Total_Stops         1
Price              55
dtype: int64
```

g) Informing about Data

```
flight.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          1500 non-null   object
 1   Date_of_Journey  1500 non-null   object
 2   Source           1500 non-null   object
 3   Destination      1500 non-null   object
 4   Dep_Time         1500 non-null   object
 5   Arrival_Time     1500 non-null   object
 6   Duration         1500 non-null   object
 7   Total_Stops      1500 non-null   object
 8   Price            1500 non-null   object
dtypes: object(9)
memory usage: 105.6+ KB
```

h) Extracting “Day”, “Date” and “Month” from Column

‘Date_of_Journey’

```
#converting into list for extraction
Journey_Date= flight['Date_of_Journey'].tolist()
```

```
#creating empty list
Day= []
date = []
Month = []
Date = []
```

```
#fetching data from 'date'
for i in date:
    Date.append(i.split(" ")[2])
    Month.append(i.split(" ")[1])
```

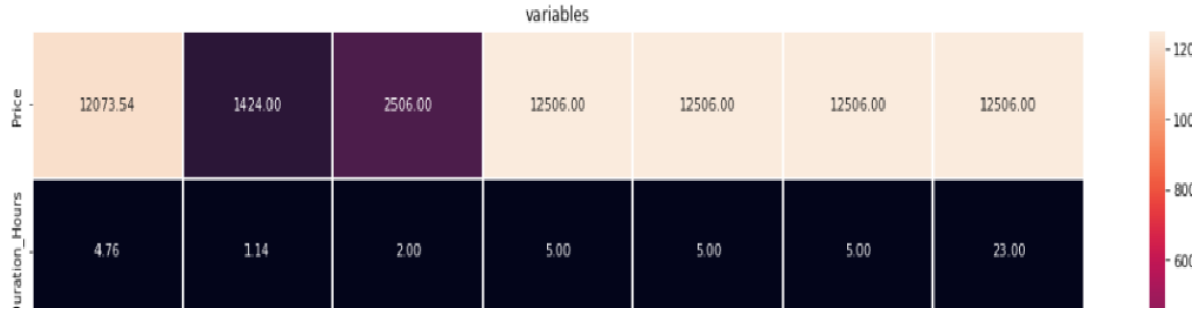
- Data Inputs- Logic- Output Relationships
  1) Descriptive Statistics

```
# Description of flight Dataset : works only on continuous column
flight.describe()
```

	Price	Duration_Hours	Duration_Minutes
count	1500.000000	1500.000000	1500.000000
mean	12073.541333	4.760000	28.433333
std	1424.001218	1.135749	5.182998
min	2506.000000	2.000000	0.000000
25%	12506.000000	5.000000	30.000000
50%	12506.000000	5.000000	30.000000
75%	12506.000000	5.000000	30.000000
max	12506.000000	23.000000	55.000000

Checking Description through heatmap

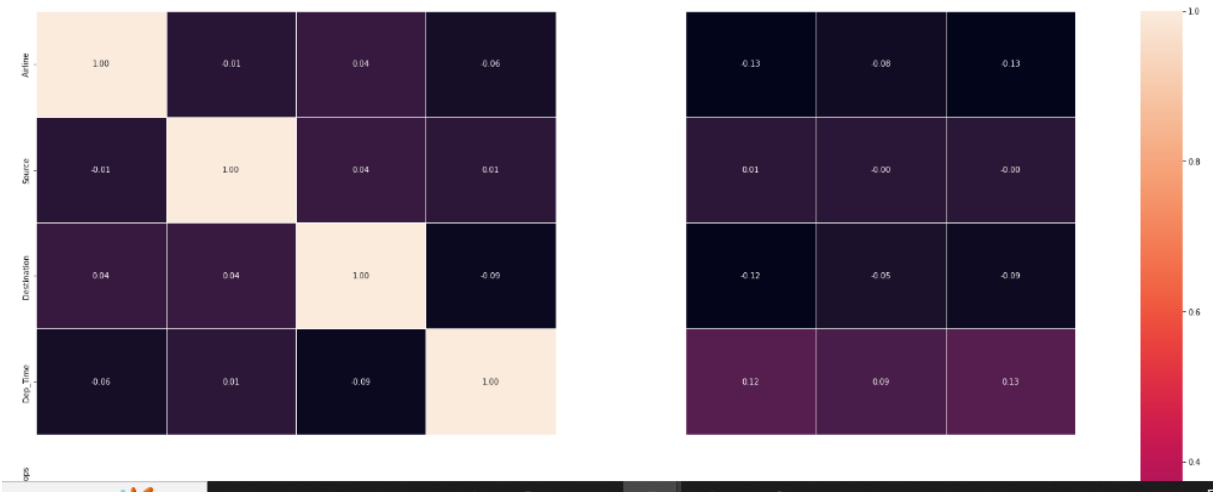
```
plt.figure(figsize=(20,5))
sns.heatmap(round(flight.describe()[1:].transpose(),2),linewidth=2,annot=True,fmt='.2f')
plt.xticks(fontsize=18)
plt.xticks(fontsize=12)
plt.title('variables')
plt.show()
```



Checking Correlation with HeatMap

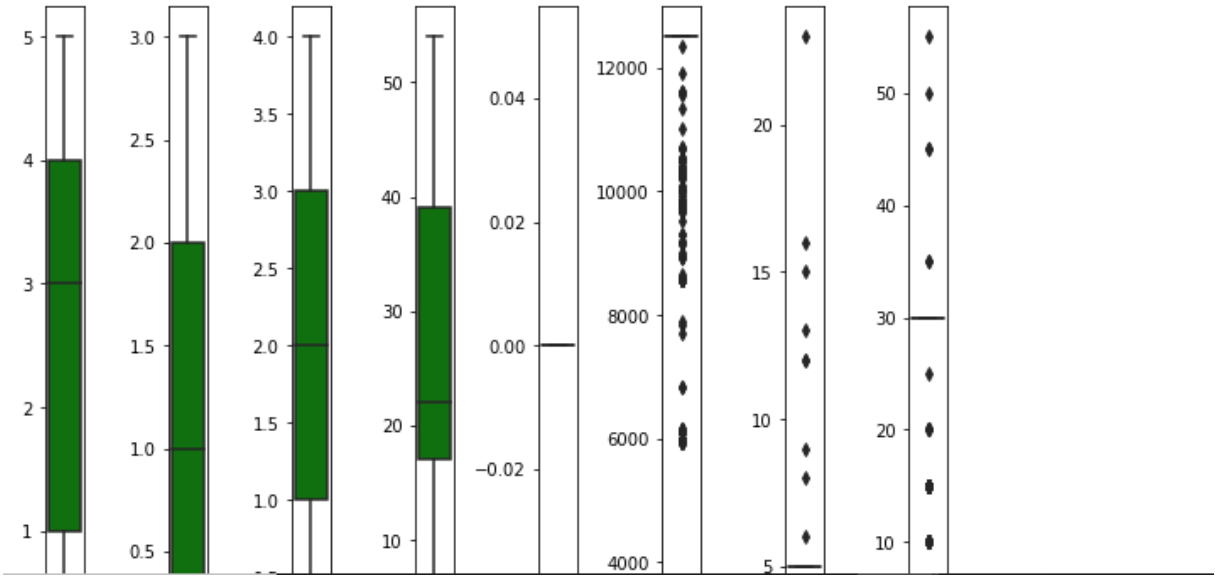
```
plt.figure(figsize=(30,20))
sns.heatmap(flight.corr(),annot=True,annot_kws= {"size": 10}, linewidth=0.5, linecolor='white', fmt='.2f')
```

```
<AxesSubplot:>
```



Checking Outliers

```
collist= flight.columns.values
ncol=14
nrows=7
plt.figure(figsize=(ncol,3*ncol))
for i in range(0,len(collist)):
    plt.subplot(nrows,ncol,i+1)
    sns.boxplot(data=flight[collist[i]],color='green',orient='v')
    plt.tight_layout()
```



Variance Threshold Method

```
var_threshold = VarianceThreshold(threshold=0)
var_threshold.fit(x)
```

```
VarianceThreshold(threshold=0)
```

```
var_threshold.get_support()
```

```
array([ True,  True,  True,  True, False,  True,  True])
```

```
x.columns[var_threshold.get_support()]
```

```
Index(['Airline', 'Source', 'Destination', 'Dep_Time', 'Duration_Hours',
       'Duration_Minutes'],
      dtype='object')
```

```
# taking  out  all the  constant  columns
cons_columns = [column for column in x.columns
                if column not in x.columns[var_threshold.get_support()]]
print(len(cons_columns))
```

```
1
```

Selecting Kbest Features

```
bestfeat = SelectKBest(score_func = f_classif, k = 'all')
fit = bestfeat.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
```

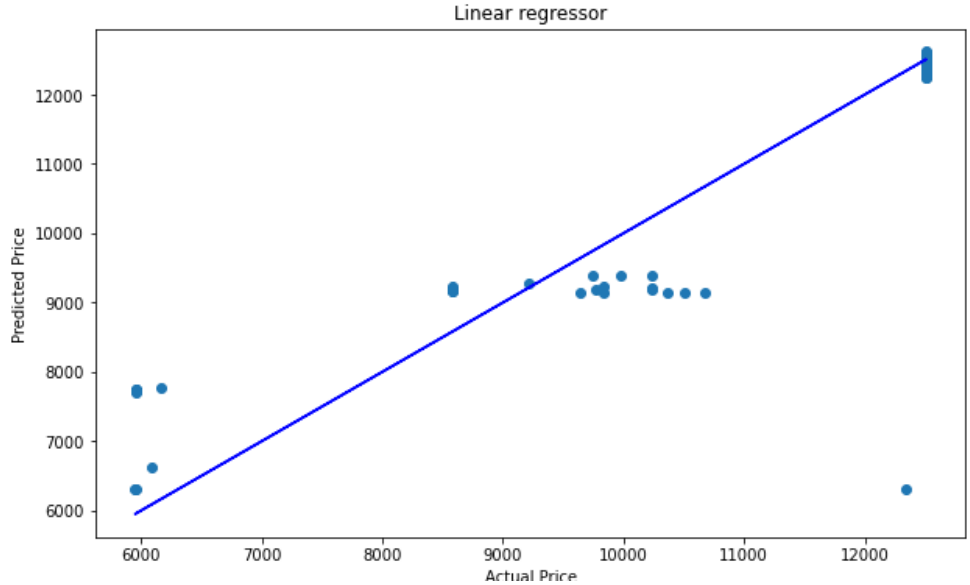
```
fit = bestfeat.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
dfcolumns.head()
featureScores = pd.concat([dfcolumns,dfscores],axis = 1)
featureScores.columns = ['Feature', 'Score']
print(featureScores.nlargest(40,'Score'))
```

```
             Feature       Score
6  Duration_Minutes  246.239769
5    Duration_Hours  226.876543
0           Airline    2.526158
2       Destination    1.381302
1            Source    0.835614
3          Dep_Time    0.660808
```

```
#checking again
bestfeat = SelectKBest(score_func = f_classif, k = 'all')
fit = bestfeat.fit(x,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(x.columns)
```

Checking the Performance of the model by graph

```
plt.figure(figsize=(10,6))
plt.scatter(x=y_test,y=predLR,cmap='set1')
plt.plot(y_test,y_test,color='b')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Linear regressor")
plt.show()
```



Cross Validation Score for all the model

```
#CV Score for Linear Regression
print('CV score for Linear Regression: ',cross_val_score(LR,x,y,cv=5).mean())

#CV Score for Random Forest Regression
print('CV score for Random forest Regression: ',cross_val_score(RFR,x,y,cv=5).mean())

#CV Score for KNN Regression
print('CV score for KNN Regression: ',cross_val_score(knn,x,y,cv=5).mean())

#CV Score for Gradient Boosting Regression
print('CV score for Gradient Boosting Regression: ',cross_val_score(Gb,x,y,cv=5).mean())

#CV Score for Decision Tree Regression
print('CV score for Decision Tree Regression: ',cross_val_score(DTR,x,y,cv=5).mean())
```

```
CV score for Linear Regression:  -0.1476413871776014
CV score for Random forest Regression:  0.053167979923595565
CV score for KNN Regression:  0.05315624100672674
CV score for Gradient Boosting Regression:  -0.14704886078642768
CV score for Decision Tree Regression:  0.45315624100672663
```

Saving The Predictive Model

```
#saving the model at local file system
filename='flight_price_prediction.pickle'
pickle.dump(CV_GBR,open(filename,'wb'))
#prediction using the saved model
loaded_model = pickle.load(open(filename, 'rb'))
loaded_model.predict(x_test)
```

```
array([12496.98526587,  9731.14059416, 12480.86982193, 12485.03109331,
       12496.99114206, 12476.62220721, 12495.33847687,  8085.36177757,
       12485.03109331, 12499.93758076, 12496.24802507, 12493.22295998,
       12493.22295998, 12497.76685031, 12498.19607534,  9254.35183468,
        5978.33520635, 12496.24802507, 12499.15599633, 12476.62220721,
       12479.30501329, 12499.32922116, 12486.19449868, 12479.30501329,
       12493.22295998, 12500.07781001, 12494.43964564, 12494.00454441,
       12494.00454441, 12478.18701585, 12498.55351291, 12486.19449868,
       12493.22883617, 12498.97765978,  6557.14606876, 12494.43376945,
       12496.99114206, 12470.63037456, 12494.00454441, 12494.43964564,
       12486.59590195, 12500.54003063, 12498.19607534, 12493.22295998,
       12462.49976874, 12498.97765978, 12485.03109331, 12493.22883617,
        9336.29697033, 12499.75844619, 12470.50725157, 12499.29622557,
       12493.22883617, 12480.86982193, 12480.86982193, 12499.29622557,
       12486.59590195, 12498.19607534, 12496.24802507, 12494.55689243,
       12496.99114206, 12498.97765978,  9692.72431975, 12485.03109331,
       12493.22883617, 12466.38845772, 12494.43964564, 12491.78371211,
       12493.22883617,  8144.18335751, 12491.78371211, 12485.03109331,
       12496.25390126, 12496.24802507, 12496.11367201, 12493.22883617,
       12479.30501329, 12486.59590195, 12493.22883617, 12496.88938026,
       12496.24802507, 12498.19607534, 12484.62969004, 12493.22883617,
       12473.75495214, 12462.49976874, 12500.07781001, 12486.59590195,
```

Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

from scipy.stats import zscore
from sklearn.preprocessing  import power_transform, StandardScaler, LabelEncoder
from sklearn.feature_selection import VarianceThreshold, SelectKBest, f_classif
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import train_test_split, GridSearchCV,cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import roc_curve, auc, roc_auc_score, plot_roc_curve, r2_score, classification_report, mean_absolute_error,
from sklearn.metrics import confusion_matrix, mean_absolute_error, mean_squared_error
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR

import pickle
```

Model/s Development and Evaluation

1) Identification of possible problem-solving approaches (methods)

- Checked total number of rows and column
- Checked All column name
- Checked datatype of all data
- Checked for null values
- Checked for special character present in dataset or not
- Checked total number of unique values
- Information about data
- Checked description of data and Data set
- Extracted hidden feature
- Checked skewness
- Scaled data
- Checked Multicollinearity
- Used feature selection Method: Variance threshold method and Select kbest Features

2) Testing of Identified Approaches (Algorithms)

Linear Regression

Random Forest Regressor

KNN Regressor

Gradient Boosting Regressor

Decision Tree Regressor