FLIGHT PRICE PREDICTION PROJECT

Submitted by:

ATUL DAHIMA

INTRODUCTION

- ***Business Problem Framing***

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.

STEPS

1. Data Collection

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 areairline 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.

2. Data Analysis

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?

3. Model Building

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

Analytical Problem Framing

- *Mathematical/ Analytical Modeling of the Problem*

❖ **DATA SCRAPPING**

In this project we first we have to scrap the data about flight prices using web scraping.

```
In [1]: #import all required libraries
        import selenium
        from selenium import webdriver
        import pandas as pd

        import warnings
        warnings.filterwarnings("ignore")

In [2]: #let's first connect to web driver
        driver=webdriver.Chrome(r"C:\Users\ankus\Downloads\chromedriver_win32 (2)\chromedriver.exe")

In [3]: #create some empty list
        flight=[]
        airline=[]
        duration=[]
        departure=[]
        arrival=[]
        price=[]
        arrival_time=[]
        departure_time=[]
        city=[]

In [4]: #scrap details from yatra.com
        url="https://flight.yatra.com/air-search-ui/dom2/trigger?type=O&viewName=normal&flexi=0&noOfSegments=1&origin=DEL&originCountry=
        driver.get(url)
```

First we import the required libraries for web scraping like selenium,webdriver ,pandas etc and install the webdriver using driver.

Now create some empty list to store the scraped the data and fetch the web page using **driver.get(url)**

```
In [5]: #flights from new delhi to mumbai
        #airline
        airline_tag=driver.find_elements_by_xpath('//span[@class="i-b text ellipsis"]')
        for i in airline_tag:
            airline.append(i.text)
        len(airline)

        #price
        price_tag=driver.find_elements_by_xpath('//div[@class="i-b tipsy fare-summary-tooltip fs-18"]')
        for i in price_tag:
            price.append(i.text)
        len(price)

        #duration
        duration_tag=driver.find_elements_by_xpath('//p[@class="fs-12 bold du mb-2"]')
        for i in duration_tag:
            duration.append(i.text)
        len(duration)


        #cities
        city_tag=driver.find_elements_by_xpath('//p[@class="fs-10 font-lightgrey no-wrap city ellipsis"]')
        for i in city_tag:
            city.append(i.text)


        departure=city[1::2]
        len(departure)


        arrival=city[0::2]
        len(arrival)
```

```
#time
#arrival_time
arr_time_tag=driver.find_elements_by_xpath('//div[@class="i-b pr"]')
for i in arr_time_tag:
    arrival_time.append(i.text)
len(arrival_time)

#departure_time
dep_time_tag=driver.find_elements_by_xpath('//p[@class="bold fs-15 mb-2 pr time"]')
for i in dep_time_tag:
    departure_time.append(i.text)
len(departure_time)


#flight
#flight
flight_tag=driver.find_elements_by_xpath('//p[@class="normal fs-11 font-lightestgrey no-wrap fl-no"]/span')
for i in flight_tag:
    flight.append(i.text)
len(flight)
```

Above is the code that how we scrap the data using absolute xpath with the help of class and tag name and them append the data to empty list which we

created earlier

```
In [6]: data=pd.DataFrame()
data["Airline"]=airline
data["Flight"]=flight
data["Duration"]=duration
data["Arrival"]=arrival
data["Departure"]=departure
data["Price"]=price
data["Arrival_time"]=arrival_time
data["Departure_time"]=departure_time
data
```

Out[6]:

	Airline	Flight	Duration	Arrival	Departure	Price	Arrival_time	Departure_time
0	IndiGo	6E-5023	2h 05m	New Delhi	Mumbai	7,319	06:30	08:35
1	IndiGo	6E-6814	2h 05m	New Delhi	Mumbai	7,319	07:20	09:25
2	IndiGo	6E-5318	2h 05m	New Delhi	Mumbai	7,319	17:00	19:05
3	Go First	G8-328	2h 05m	New Delhi	Mumbai	7,319	20:50	22:55
4	IndiGo	6E-2519	2h 05m	New Delhi	Mumbai	7,319	23:35	01:40\n+ 1 day
...	...	...	...	...	...	...	...	...
154	Vistara	UK-847/842	8h 50m	New Delhi	Mumbai	16,718	11:10	20:00
155	SpiceJet	SG-8373/945	6h 20m	New Delhi	Mumbai	18,014	09:50	16:10
156	SpiceJet	SG-8963/945	6h 45m	New Delhi	Mumbai	19,168	09:25	16:10
157	SpiceJet	SG-473/945	9h 45m	New Delhi	Mumbai	19,798	06:25	16:10
158	Go First	G8-2403/2412	6h 55m	New Delhi	Mumbai	24,932	10:30	17:25

After scraping the data make the data frame for the data and start the model building.

Model Building

```
In [15]: #import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

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

```
In [16]: data.head()
```

Out[16]:

	Airline	Flight	Duration	Arrival	Departure	Price	Arrival_time	Departure_time
0	IndiGo	6E-5023	2h 05m	New Delhi	Mumbai	7,319	06:30	08:35
1	IndiGo	6E-6814	2h 05m	New Delhi	Mumbai	7,319	07:20	09:25
2	IndiGo	6E-5318	2h 05m	New Delhi	Mumbai	7,319	17:00	19:05
3	Go First	G8-328	2h 05m	New Delhi	Mumbai	7,319	20:50	22:55
4	IndiGo	6E-2519	2h 05m	New Delhi	Mumbai	7,319	23:35	01:40\n+ 1 day

```
In [17]: data.isnull().sum()
```

```
Out[17]: Airline          0
Flight           0
Duration         0
Arrival          0
Departure        0
Price            0
Arrival_time     0
Departure_time   0
dtype: int64
```

First we will import some required libraries and then import the dataset. We have to make prediction on the dataset

Few libraries are used initially for different purpose like:-

- *Numpy*:- for mathematical operation
- *Pandas*:- for data preprocessing
- *Matplotlib and Seaborn*:- for data visualization
- *Warnings*:- for some warnings

➢ **data.isnull().sum()** *shows that there is no null values in the data*

❖ ***LABEL ENCODING:-***

As most of the columns are having text or symbol so we will do label encoding first and then we will check the decription of the dataset.

LABEL ENCODING

In [18]:
```
from sklearn.preprocessing import LabelEncoder

data= data.copy()
label_encoder = LabelEncoder()
for column in data.columns:
    data[column] = label_encoder.fit_transform(data[column])
```

In [19]:
```
data.head()
```

Out[19]:

	Airline	Flight	Duration	Arrival	Departure	Price	Arrival_time	Departure_time
0	3	35	77	1	0	70	23	41
1	3	62	77	1	0	70	30	54
2	3	41	77	1	0	70	92	135
3	2	148	77	1	0	70	131	171
4	3	22	77	1	0	70	147	16

now we are good to go

In [20]:
```
data.describe()
```

Out[20]:

	Airline	Flight	Duration	Arrival	Departure	Price	Arrival_time	Departure_time
count	332.000000	332.000000	332.000000	332.000000	332.000000	332.000000	332.000000	332.000000
mean	3.075301	165.500000	80.478916	0.478916	0.521084	61.105422	72.900602	97.653614
std	1.686195	95.984374	39.246716	0.500309	0.500309	25.809277	42.698115	55.883110
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	82.750000	53.750000	0.000000	0.000000	45.000000	34.000000	48.750000

data.describe() *shows some calculation about the data like:-mean,median,mode,std and some other calculation*

On the top of the data description we can see that all the columns have same value which also defines that there is no null value

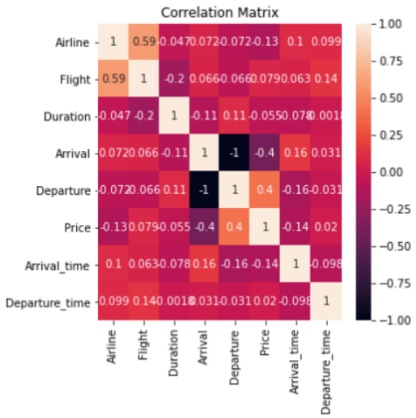
❖ ***Correlation :-***

check correlation

In [24]:
```
corr_mat=data.corr()

#size of the canvas
plt.figure(figsize=[5,5])

#plot the correlation matrix
sns.heatmap(corr_mat,annot=True)
plt.title("Correlation Matrix")
plt.show()
```



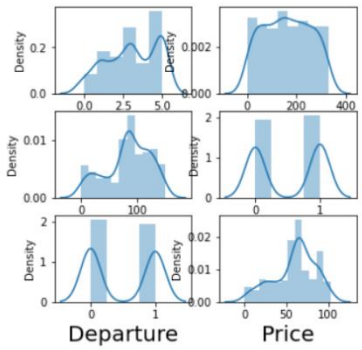
Above heatmap shows the correlation among all the columns.

If any columns are highly correlated to each other we can remove one of them

❖ ***DATA DISTRIBUTION:-***

In [25]:
```
#let's see how data is distributed for every column
plt.figure(figsize=(5,5))
plotnumber=1

for column in data:
    if plotnumber<=6:
        ax=plt.subplot(3,2,plotnumber)
        sns.distplot(data[column])
        plt.xlabel(column,fontsize=20)
    plotnumber+=1
plt.show()
```



Above code shows the data distribution among all the columns , let go ahead with checking skewness in the data distribution.

- ***Data Preprocessing Done***

Separate target and variables

seperate label and features

```
In [21]: x=data.drop(['Price'],axis=1)
         y=data['Price']

In [22]: x
```

Out[22]:

	Airline	Flight	Duration	Arrival	Departure	Arrival_time	Departure_time
0	3	35	77	1	0	23	41
1	3	62	77	1	0	30	54
2	3	41	77	1	0	92	135
3	2	148	77	1	0	131	171
4	3	22	77	1	0	147	16
...	...	...	...	...	...	...	...
327	0	189	101	0	1	41	93
328	5	242	119	0	1	45	110
329	4	203	107	0	1	66	134
330	2	144	135	0	1	100	19
331	4	199	20	0	1	19	128

332 rows × 7 columns

```
In [23]: y
```

```
Out[23]: 0      70
         1      70
         2      70
         3      70
         4      70
```

Check skewness:-

check for skewness

```
In [26]: #checking skewness
         x.skew().sort_values()
```

```
Out[26]: Duration         -0.485655
         Airline          -0.281694
         Departure_time   -0.122711
         Arrival          -0.056440
         Flight            0.014556
         Departure         0.056440
         Arrival_time      0.057130
         dtype: float64
```

data distribution is all fine

Data is perfectly fine so we are good to go.

DATA SCALING:-

data scaling

In [28]:
```
#data scaling using standard scaler
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
```

In [29]:
```
x_scaled=scaler.fit_transform(x)
x_scaled
```

Out[29]:
```
array([[-0.03611495, -1.34801058, -0.09296224, ..., -0.97229378,
        -1.15886662, -0.99194589],
       [-0.03611495, -1.06167898, -0.09296224, ..., -0.97229378,
        -0.9999203 , -0.75594314],
       [-0.03611495, -1.28438133, -0.09296224, ..., -0.97229378,
         0.49871639,  0.67822745],
       ...,
       [ 1.18835399,  0.82598863,  1.00593039, ...,  1.02849573,
        -0.59120121,  0.18806788],
       [ 0.57611952,  0.42300341,  0.69926268, ...,  1.02849573,
        -0.11436226,  0.64191934],
       [ 0.57611952,  0.38058391, -1.65185644, ...,  1.02849573,
        -1.24969309,  0.53299499]])
```

Now check the vif score for scaled data

vif score

In [30]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif"]=[variance_inflation_factor(x_scaled, i) for i in range(x_scaled.shape[1])]

vif["Features"]=x.columns

#let's check the values
vif
```

Out[30]:

	vif	Features
0	1.546269	Airline
1	1.619440	Flight
2	1.070697	Duration
3	inf	Arrival
4	inf	Departure
5	1.035128	Arrival_time
6	1.028025	Departure_time

all the values are less than 5 so it is good

As the value of all the columns is less than 5 that means there is no overfitting of the data

Training Process:-

Import the required libraries for the training process and split the data into training and testing process

training process

In [31]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import r2_score,confusion_matrix,classification_report
from sklearn.model_selection import train_test_split
```

In [32]:
```
#finding  best random state
maxAccu=0     #maximum accuracy
maxRS=0       #best random state value for which max accuracy is achieved

for i in range(1,100):
    x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=i)
    lr.fit(x_train,y_train)
    pred=lr.predict(x_test)
    acc=r2_score(y_test,pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best Accuracy is :",maxAccu,"on Random_state",maxRS)
```

```
Best Accuracy is : 0.4406070296260963 on Random_state 77
```

In [33]:
```
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.25,random_state=77)
```

Now trained the model using linear regression and check the r2 score

In [38]:
```
lr.fit(x_train,y_train)
```

Out[38]:
```
LinearRegression()
```

In [39]:
```
#Ajusted r2 score
lr.score(x_train,y_train)*100
```

Out[39]:
```
21.0773535933965
```

let's check how well it fits the test data

In [40]:
```
lr.score(x_test,y_test)*100
```

Out[40]:
```
44.06070296260963
```

CROSS VALIDATON :-

cross validation of the model

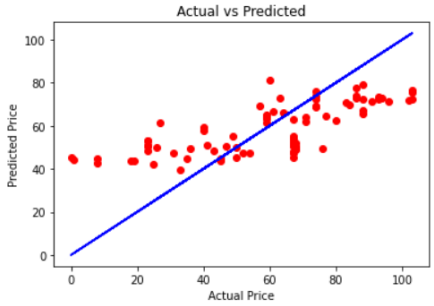
In [41]:
```
train_accuracy=lr.score(x_train,y_train)
test_accuracy=lr.score(x_test,y_test)
from sklearn.model_selection import cross_val_score,GridSearchCV
cv_score=cross_val_score(lr,x,y,cv=3)
cv_mean=cv_score.mean()
cv_mean
```

Out[41]:
```
-0.5895893495346044
```

In [42]:
```
y_pred=lr.predict(x_test)
y_pred
```

Out[42]:
```
array([54.33261596, 75.34627712, 49.91826783, 50.32755302, 47.93484447,
       73.70165857, 69.40131687, 65.19728771, 43.7817859 , 63.97317533,
       71.22717831, 72.64253845, 76.73826967, 66.77499459, 44.61089247,
       51.63464629, 47.17199126, 72.87519623, 64.39587928, 71.21322042,
       47.4850401 , 55.39326699, 43.93313554, 51.06617394, 39.53670347,
       72.9593006 , 63.89044528, 77.58340088, 69.83888258, 61.32793838,
       52.93541206, 63.45519541, 50.73090027, 45.46336513, 55.34842741,
       52.72949142, 71.98161823, 64.71386237, 44.55725716, 47.93484447,
       45.16984668, 63.66900558, 50.74625866, 43.72079749, 72.74973397,
       66.93696081, 62.58226273, 47.54634971, 68.67587673, 64.72020207,
       49.59516324, 70.75966873, 61.82879947, 74.13793852, 63.18203989,
       81.37922834, 54.22384786, 79.24240382, 51.82344964, 65.45810084,
       48.56998272, 42.11866076, 50.11470848, 75.83614954, 44.65982223,
       50.73568151, 72.6016121 , 45.4199198 , 51.25704615, 47.4189518 ,
       59.52741185, 72.44307194, 53.01496451, 49.51496672, 49.67852153,
       66.14523835, 61.40374707, 69.37439931, 51.18671281, 49.07318857,
       44.0601924 , 42.79815176, 49.57863563, 48.57292557, 73.30545909,
       52.27474136, 53.99110586, 57.84197767, 72.39381162])
```

In [43]:
```
plt.scatter(y_test,y_pred,color='r')
plt.plot(y_test,y_test,color='b')
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted")
plt.show()
```



Above we do the cross validation of the model and then plot the graph for actual and predicted price of the flights

HYPER PARAMETER TUNNING:-

ensemble technique

In [44]:
```
from sklearn.ensemble import RandomForestRegressor


parameters={'criterion':['mse','mae'],'max_features':["auto","sqrt","log2"]}
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(x_train,y_train)

print(clf.best_params_)
```

```
{'criterion': 'mse', 'max_features': 'sqrt'}
```

In [45]:
```
rf=RandomForestRegressor(criterion="mse",max_features="auto")
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
pred_decision=rf.predict(x_test)

rfs=r2_score(y_test,pred_decision)
print('R2_score :',rfs*100)

rfscore=cross_val_score(rf,x,y,cv=3)
rfc=rfscore.mean()
print('Cross_val_score :',rfc*100)
```

```
R2_score : 57.32290146492136
Cross_val_score : -90.64983111694994
```

Perform the hyperparameter tunning for the model using RandomForestRegressor and and the find the best parameter using GridSearchCV for final r2 Score

CONCLUSION

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn and matplotlib to do the visualizations. During the data pre-processing part. Afterwards we started training model using Linear regression, Lasso regression, Ridge regression, and applied cross validation on it. Then we do the Hyper Parameter tunning of the model and get the final accuracy score for the model. Later we save the model for later use for prediction