FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

Dinesh Sharma

Introduction

Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices.

Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.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.

Data Collection Phase

The data has been collected from Yatra.com website for different location across the India. These columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price

• Which variables are important to predict the price of variable?

How do these variables describe the price of the air ticket?

We can observe that data have 1233 rows and 11 columns.

There are 2 numeric columns and 9 categorical columns. With the first look, we can see that there are no missing values in the data.

'Price'column/feature is going to be the target column or dependent feature for this project.

Loading the required libraries

```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,mean_absolute_error
import warnings
warnings.filterwarnings('ignore')
```

```
#Uploading the the data set
data_train=pd.read_csv('Final_flight.csv')
ds=pd.DataFrame(data=data_train)
ds
```


```
#Uploading the the data set
data_train=pd.read_csv('Final_flight.csv')
ds=pd.DataFrame(data=data_train)
ds
```


	Airlines	Date_of_Journey	Source	Destination	Dep_times	Arrival_time	Duration	Stops	Price
0	SpiceJet	10/11/2021	New Delhi	Mumbai	7:20	9:35	2h 15m	Non Stop	2998
1	SpiceJet	10/11/2021	New Delhi	Mumbai	6:20	8:40	2h 20m	Non Stop	2998
2	SpiceJet	10/11/2021	New Delhi	Mumbai	19:45	22:05	2h 20m	Non Stop	2998
3	SpiceJet	10/11/2021	New Delhi	Mumbai	18:55	21:05	2h 10m	Non Stop	3177
4	Air India	10/11/2021	New Delhi	Mumbai	7:00	9:05	2h 05m	Non Stop	4931
...	...	...	...	...	...	...	...	...	...
1225	IndiGo	27/11/2021	New Delhi	Hyderabad	11:20	17:35	6h 15m	1 Stop	16718
1226	Vistara	27/11/2021	New Delhi	Hyderabad	8:05	20:25	12h 20m	1 Stop	11310
1227	Air India	27/11/2021	New Delhi	Hyderabad	9:45	19:00	9h 15m	2 Stop(s)	11678
1228	Air India	27/11/2021	New Delhi	Hyderabad	6:10	19:00	12h 50m	2 Stop(s)	11678
1229	Go First	27/11/2021	New Delhi	Hyderabad	10:45	23:20	12h 35m	1 Stop	15039

1230 rows × 9 columns

EDA

```
ds.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1230 entries, 0 to 1229
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airlines         1230 non-null   object
 1   Date_of_Journey  1230 non-null   object
 2   Source           1230 non-null   object
 3   Destination      1230 non-null   object
 4   Dep_times        1230 non-null   object
 5   Arrival_time     1230 non-null   object
 6   Duration         1230 non-null   object
 7   Stops            1230 non-null   object
 8   Price            1230 non-null   int64 
dtypes: int64(1), object(8)
memory usage: 86.6+ KB
```

here are 1203 rows and 9 columns.All are object type varibale. Target varibale is continous and interger type.

In [82]:
```
ds.isnull().sum()
```

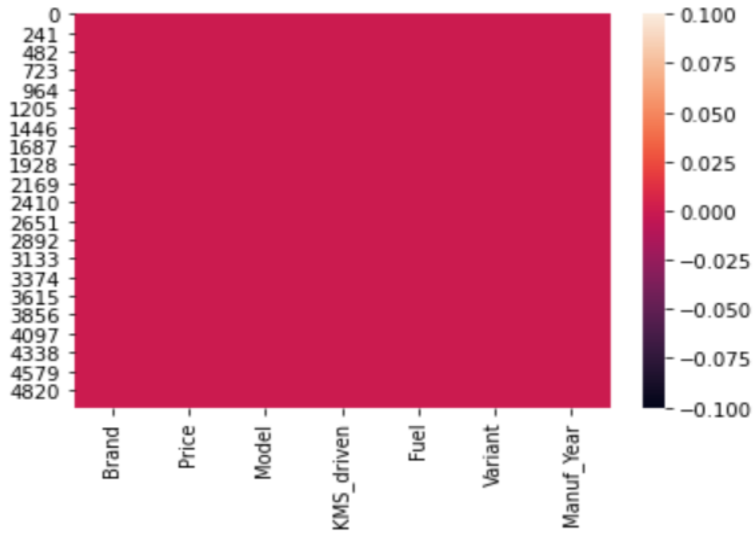
Out[82]:
```
Unnamed: 0    0
Brand         0
Price         0
Model         0
KMS_driven    0
Fuel          0
Variant       0
dtype: int64
```

There are no missing values in the data set

Since our data raw and messed up. First lets make it bit meanigfull before we do visualization.

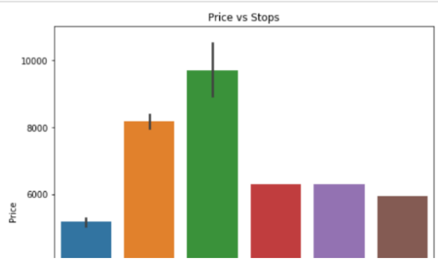
In [87]:
```
sns.heatmap(ds.isnull())
```

Out[87]: <AxesSubplot:>

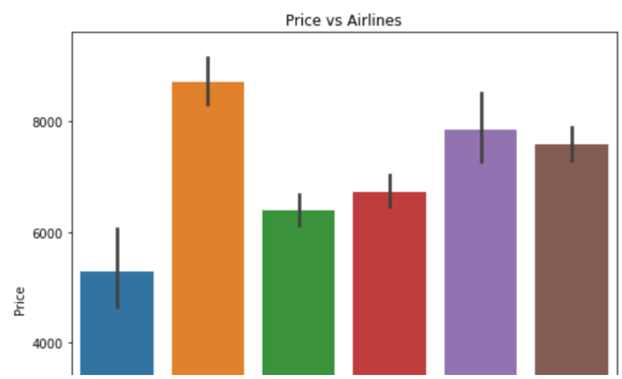


So there is no missing valuses in the data set

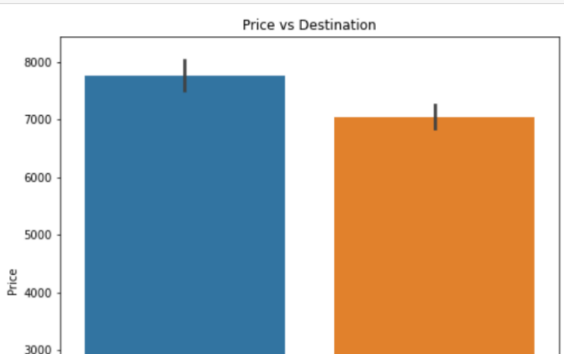
```
plt.figure(figsize=(8,8))
sns.barplot(x='Stops',y='Price',data=ds)
plt.title("Price vs Stops")
plt.xticks(rotation=45)
plt.show()
```




```
plt.figure(figsize=(8,8))
sns.barplot(x='Airlines',y='Price',data=ds)
plt.title("Price vs Airlines")
plt.xticks(rotation=45)
plt.show()
```

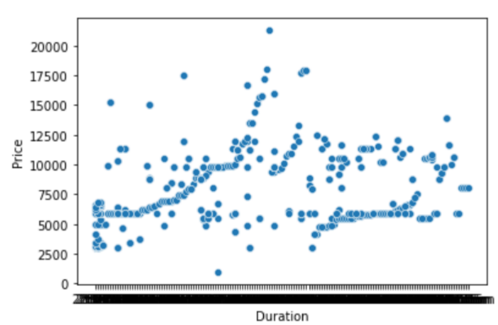



```
plt.figure(figsize=(8,8))
sns.barplot(x='Destination',y='Price',data=ds)
plt.title("Price vs Destination")
plt.xticks(rotation=45)
plt.show()
```




```
#duration v/s AveragePrice
sns.scatterplot(data=ds, x='Duration', y='Price')
```

<AxesSubplot:xlabel='Duration', ylabel='Price'>



- We know that duration is important and plays a major role in affecting air ticket prices but we see no such pattern here, there must be other significant factors affecting air fare like type of airline, destination of flight, date of journey of flight.
- The flights with destination Mumbai has the highest flight price then Hyderbad ,New delhi respectively.
- As the number of stops increases the price of flight also increases.

Data Wrangling

```
#firstly lets convert Date_of_journey,Arrival_Time and Dep_Time variables into date and time for proper predicion
def change_into_datetime(col):
    ds[col]=pd.to_datetime(ds[col])
```

```
for i in ['Date_of_Journey','Dep_times', 'Arrival_time']:
    change_into_datetime(i)
```

```
# Now we extract day and month from Date_of_journey and stored in 2 other columns
ds['journey-day']=ds['Date_of_Journey'].dt.day
ds['journey-month']=ds['Date_of_Journey'].dt.month
```

```
#now can drop 'Date_of_Journey' column
ds.drop('Date_of_Journey', axis=1, inplace=True)
```

```
# function for extracting hour and minutes From Arrival_time and Dept_time
def extract_hour(ds,col):
```

```
# duration column,Separate hours and minute from duration
duration=list(ds['Duration'])
for i in range(len(duration)):
    if len(duration[i].split(' '))==2:
        pass
    else:
        if 'h' in duration[i]: # Check if duration contains only hour
            duration[i]=duration[i] + ' 0m' # Adds 0 minute
        else:
            duration[i]='0h '+ duration[i]
```

```
ds['Duration']=duration
ds.head()
```

	Airlines	Source	Destination	Duration	Stops	Price	journey-day	journey-month	Dep_times_hour	Dep_times_min	Arrival_time_hour	Ai
0	SpiceJet	New Delhi	Mumbai	2h 15m	Non Stop	2998	11	10	7	20	9	
1	SpiceJet	New Delhi	Mumbai	2h 20m	Non Stop	2998	11	10	6	20	8	
2	SpiceJet	New Delhi	Mumbai	2h 20m	Non Stop	2998	11	10	19	45	22	

```python
# Correlation Matrix ---Pearson Method
dfcor=ds.corr()
plt.figure(figsize=(10,6))
sns.heatmap(dfcor,cmap="YlOrRd_r",annot=True)
```

```
<AxesSubplot:>
```

Creating feature and target dataframe

```python
x=dsnew.drop(columns=['Price'])
y=dsnew['Price']
```

```python
x.shape
```

```
(1203, 12)
```

```python
y.shape
```

```
(1203,)
```

```python
# lets bring all feature into common scale
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X=sc.fit_transform(x)
```

```python
# To find the best random state using Linear Regressor model

from sklearn.metrics import r2_score
```

```
# Sending the data for train and test using Train_test_Split
# 30 % data will go for testing and 70% data will go for training the model
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=.30,random_state=maxRS)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(842, 12)
(361, 12)
(842,)
(361,)
```

70% of the data (842 rows) will be available for training the model & 30% (361 rows) will be available for testing the model

Model Building & Evaluation

Since the target variable as continuous values we can build the regression models. Therefore oue evalution criteria will be: Evaluation Matrics : MAE,MSE,RMSE and R2 Score

Model Building

We built LinearRegression, XGBoost, and RandomForest as machine learning models and two deep learning models one having a small network and another having a large network.We built base models of LinearRegression, XGBoost, and RandomForest so there is not much to show about these models but we can see the model summary and how they converge with deep learning models that we built.

```
# Decision Tree Regression Model
dc=DecisionTreeRegressor()
dc.fit(x_train,y_train)
dc.score(x_train,y_train)
```

```
0.9743936232512141
```

```
from sklearn.metrics import r2_score
pred=dc.predict(x_test)
print('Coefficient of determination',r2_score(y_test,pred))
print('mean absolute error',mean_absolute_error(y_test,pred))
print('mean squarred error',mean_squared_error(y_test,pred))
print('Root mean square error',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Coefficient of determination 0.7160431903269577
mean absolute error 469.3753462603878
mean squarred error 1938124.0644044322
Root mean square error 1392.1652432108885
```

```
# KNeighbors Regression Model
kn=KNeighborsRegressor()
kn.fit(x_train,y_train)
kn.score(x_train,y_train)
```

```
0.6974127999842219
```

```python
from sklearn.metrics import r2_score
pred=kn.predict(x_test)
print('Coefficient of determination',r2_score(y_test,pred))
print('mean absolute error',mean_absolute_error(y_test,pred))
print('mean squarred error',mean_squared_error(y_test,pred))
print('Root mean square error',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Coefficient of determination 0.5277430273792334
mean absolute error 1326.4581717451524
mean squarred error 3223351.4817728535
Root mean square error 1795.3694555084905
```

```python
# Random Forest Regression Model
rf=RandomForestRegressor()
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
```

```
0.958454230420262
```

```python
from sklearn.metrics import r2_score
pred=rf.predict(x_test)
print('Coefficient of determination',r2_score(y_test,pred))
print('mean absolute error',mean_absolute_error(y_test,pred))
print('mean squarred error',mean_squared_error(y_test,pred))
```

```python
from sklearn.metrics import r2_score
pred=rf.predict(x_test)
print('Coefficient of determination',r2_score(y_test,pred))
print('mean absolute error',mean_absolute_error(y_test,pred))
print('mean squarred error',mean_squared_error(y_test,pred))
print('Root mean square error',np.sqrt(mean_squared_error(y_test,pred)))
```

```
Coefficient of determination 0.8483442709909098
mean absolute error 522.8996741524866
mean squarred error 1035113.8197240402
Root mean square error 1017.4054352734903
```

Based on the results of above models, and capomaring the R2 score and other evalution matrics result of MAE,MSE and RMSE.We can find the Random Forest Regression model is best model to predict the . Since the Random Forest model has the second hishest score(0.98) and R2 score(0.92) and lowest values of MAE, MSE,RMSE among other four models build above, it is the best model among the above five models

Improving the model accuracy using cross Validation¶

```
from sklearn.model_selection import cross_val_score
lmscores =cross_val_score(lm,x,y,cv=5)
print(lmscores)
print(lmscores.mean(),lmscores.std())
```

```
[0.34335308 0.23635697 0.53728487 0.20729467 0.26216924]
0.3172917664688602 0.11896592158596964
```

```
from sklearn.model_selection import cross_val_score
dcscores =cross_val_score(dc,x,y,cv=5)
print(dcscores)
print(dcscores.mean(),dcscores.std())
```

```
[ 0.86122973  0.81716691  0.53320681  0.24115754 -0.31824385]
0.4269034281096612 0.43404869243524363
```

```
from sklearn.model_selection import cross_val_score
knnscores =cross_val_score(kn,x,y,cv=5)
```

HyperParameter Tuning

```
from sklearn.model_selection import GridSearchCV
parameter={'criterion':['mse', 'mae'],'max_features':['auto', 'sqrt']}
```

```
GCV=GridSearchCV(RandomForestRegressor(),parameter,cv=5)
```

```
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse', 'mae'],
                         'max_features': ['auto', 'sqrt']})
```

```
GCV.best_params_
```

```
{'criterion': 'mae', 'max_features': 'auto'}
```

```
rf_final=RandomForestRegressor(criterion= 'mae', max_depth= 30, max_features= 'auto')
rf_final.fit(x_train,y_train)
rf_final.score(x_train,y_train)
```

```
0.9532751764878631
```

```
In [67]: from sklearn.metrics import r2_score
         pred=rf_final.predict(x_test)
         print('Coefficient of determination',r2_score(y_test,pred))
         print('mean absolute error',mean_absolute_error(y_test,pred))
         print('mean squarred error',mean_squared_error(y_test,pred))
         print('Root mean square error',np.sqrt(mean_squared_error(y_test,pred)))

         Coefficient of determination 0.8750229718232851
         mean absolute error 516.6959972299169
         mean squarred error 853020.521275554
         Root mean square error 923.5911006909681
```

Saving the best Model

```
In [68]: import joblib
         joblib.dump(rf_final,'Flight-Price_Preiction.obj')

Out[68]: ['Flight-Price_Preiction.obj']

In [ ]:
```

Conclusion

In this project, we tried predicting the flight price using the various parameters that were provided in the data about the flight. We build machine learning models to predict prices and saw that machine learning-based models performed well at this data.

By performing different ML models, we aim to get a better result or less error with max accuracy. Our purpose was to predict the price of the flights having 9 predictors and 12033 data entries. Initially, data cleaning is performed to remove the null values and outliers from the dataset then ML models are implemented to predict the price . Next, with the help of data visualization features were explored deeply. The relation between the features is examined.

From the below table, it can be concluded that Random Forest Regressor is the best model for the prediction for used flight prices. The regression model gave the best MSLE and RMSLE values.