FLIGHT TICKET PRICE PREDICTION

SUBMITTED BY

Gyan Prakash Tripathi

AGENDA:

- Data Scraping
- Data Preparation
- Explanatory Data Analysis
- Data Cleaning & Imputation
- Remove Outliers
- Feature engineering
- Model selection
- Model Prediction
- Conclusion

DATA SCRAPING

- Flight ticket prices are scraped from online websites like Yatra.com
- Data is scraped for major source cities like Delhi, Bangalore, Mumbai, Ahmedabad,, Kolkata, etc.
- Different features like Airline, Source, Destination, Arrival Time, Duration, Total Stops, Duration, Total Stops, Additional_Info, and Price were scraped .
- Around 1948 data were scraped for 9 different features.
- Some data cleaning was done.
- The data frame was made and changed into CSV file.
- The final dataset has 1948 rows and 9 features which was further loaded for EDA and Machine learning.

DATA PREPROCESSING

- Checking the shape of Datasets
- Checking the columns
- Checking the Data types Of independent features
- Checking the null values
- Checking and dropping the unwanted columns
- Checking Categorical columns and numerical columns

CHECKING THE DATATYPE

```
flight_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1948 entries, 0 to 1947
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          1948 non-null   object
 1   Source           1948 non-null   object
 2   Destination      1948 non-null   object
 3   Dep_Time         1948 non-null   object
 4   Arrival_Time     1948 non-null   object
 5   Duration         1948 non-null   object
 6   Total_Stops      1948 non-null   object
 7   Additional_Info  1769 non-null   object
 8   Price (in ₹)     1948 non-null   object
dtypes: object(9)
memory usage: 137.1+ KB
```

- Dataset all the variables as object datatypes.
- Target column is also object type. I converted the price column to the integer type.
- We will do regression analysis.

ADDRESS NULL VALUE FIELDS

- Additional_Info feature have 179 null values.
- There are many rows in Additional_info which are having no info. So we will convert them also as nan value.

```
#To check percent of missing data in column Additional info
flight_df['Additional_Info']. isnull(). sum() * 100 / len(flight_df['Additional_Info'])
```

```
78.64476386036961
```

- We can see more than 78% of data in Additional info is null. So we can drop the info column.

```
flight_df.drop('Additional_Info', axis=1, inplace=True)
flight_df.head()
```

	Airline	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Price (in ₹)
0	Air India	New Delhi	Mumbai	07:00	09:05	2h 05m	Non Stop	4,065
1	Air India	New Delhi	Mumbai	08:00	10:10	2h 10m	Non Stop	4,065

CREATING HOUR AND MINS FEATURES SEPARATELY

Departure Hrs and Mins

```
# Departure time is when a plane leaves the gate.

# Extracting Hours
flight_df["Dep_hour"] = pd.to_datetime(flight_df["Dep_Time"]).dt.hour

# Extracting Minutes
flight_df["Dep_min"] = pd.to_datetime(flight_df["Dep_Time"]).dt.minute

# Now we can drop Dep_Time as it is of no use
flight_df.drop(["Dep_Time"], axis = 1, inplace = True)
```

Arrival Hrs and Mins

```
# Arrival time is when the plane pulls up to the gate.

# Extracting Hours
flight_df["Arrival_hour"] = pd.to_datetime(flight_df['Arrival_Time']).dt.hour

# Extracting Minutes
flight_df["Arrival_min"] = pd.to_datetime(flight_df['Arrival_Time']).dt.minute

# Now we can drop Arrival_Time as it is of no use
flight_df.drop(["Arrival_Time"], axis = 1, inplace = True)
```

EXTRACTING THE HOURS AND MIN FROM THE DURATION COLUMN

```
duration = list(flight_df["Duration"])
for i in range(len(duration)):
    if len(duration[i].split()) !=2:
        if "h" in duration[i]:
             duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]
duration_hrs = []
duration_min = []

for i in range(len(duration)):
    duration_hrs.append(int(duration[i].split("h")[0]))
    duration_min.append(int(duration[i].split("m")[0].split()[-1]))
```

```
flight_df["Duration_hours"] = duration_hrs
flight_df["Duration_Min"] = duration_hrs
flight_df.drop("Duration",axis = 1,inplace = True)
```

CHANGING NUMBER OF STOPS AND PRICE TO NUMERICAL FORM

```python
# Replacing Total_Stops
flight_df.replace({"Non Stop": 0, "1 Stop": 1, "2 Stop(s)": 2}, inplace = True)
```

```python
flight_df.head(2)
```

	Airline	Source	Destination	Total_Stops	Price (in ₹)	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_Min
0	Air India	New Delhi	Mumbai	0	4,065	7	0	9	5	2	2
1	Air India	New Delhi	Mumbai	0	4,065	8	0	10	10	2	2

```python
def convert_price(flight_df):
    flight_df['Price (in ₹)'] = flight_df['Price (in ₹)'].str.replace(',', '')
    flight_df['Price (in ₹)'] = flight_df['Price (in ₹)'].astype('int64') # con
    return flight_df
```

```python
print(convert_price(flight_df))
```

```
     Airline      Source Destination  Total_Stops  Price (in ₹)  Dep_hour  \
0    Air India  New Delhi      Mumbai            0          4065         7
1    Air India  New Delhi      Mumbai            0          4065         8
2    Air India  New Delhi      Mumbai            0          4065         9
3    Air India  New Delhi      Mumbai            0          4065        14
4    Air India  New Delhi      Mumbai            0          4065        21
```

Univariate Analysis

Univariate data requires to analyse each variable separately. It doesn't deal with causes or relationships among the features.

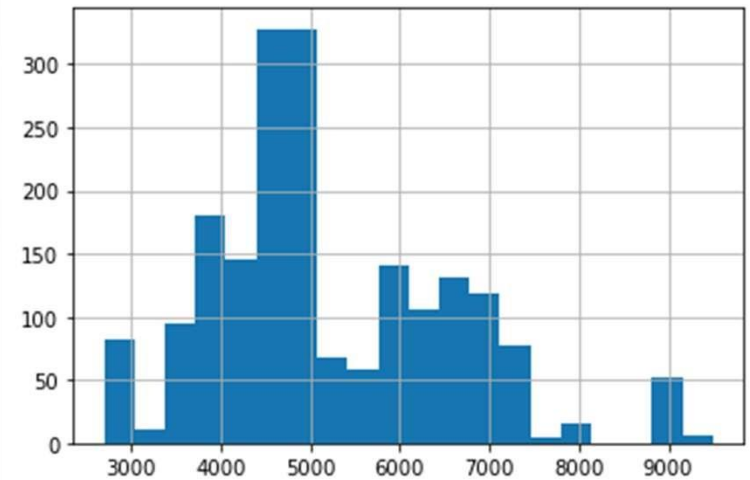
ANALYZING TARGET COLUMN-SELLING PRICE

```
flight_df["Price (in ₹)"].describe()
```

```
count    1948.000000
mean     5199.128337
std      1355.415178
min      2692.000000
25%      4352.000000
50%      4768.000000
75%      6269.750000
max      9500.000000
Name: Price (in ₹), dtype: float64
```

```
#histogram
flight_df['Price (in ₹)'].hist(bins = 20)
```

```
<AxesSubplot:>
```

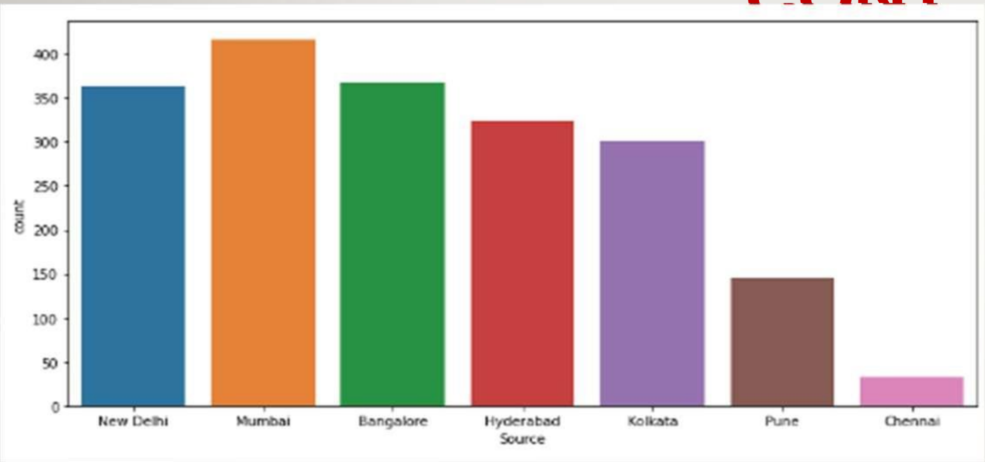



```
#skewness & kurtosis
print("Skewness: %f" % flight_df['Price (in ₹)'].skew())
print("Kurtosis: %f" % flight_df['Price (in ₹)'].kurt())
```

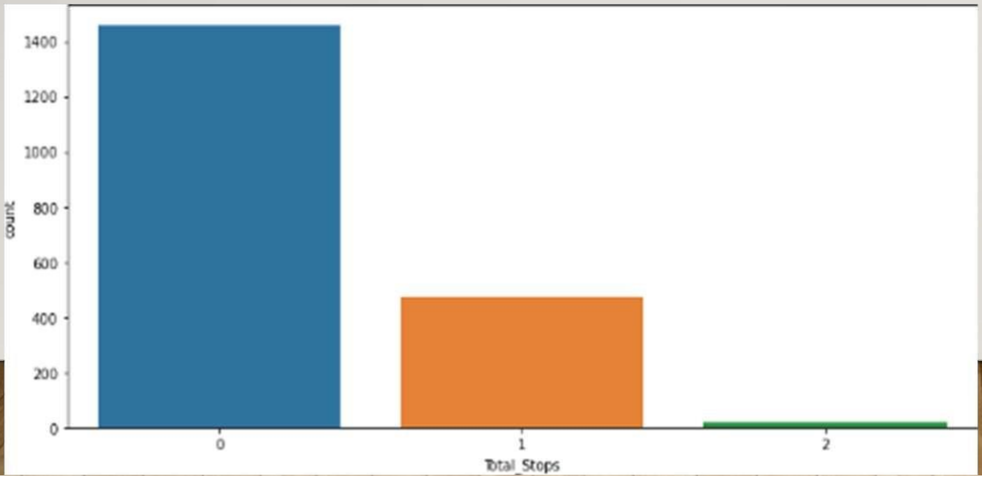
```
Skewness: 0.789319
Kurtosis: 0.465031
```

Price column has some outliers. Minimum Price is Rs 2692 and maximum price is Rs 9500.Also the data is not much skewed.

ANALYZING CATEGORICAL COLUMNS CONT

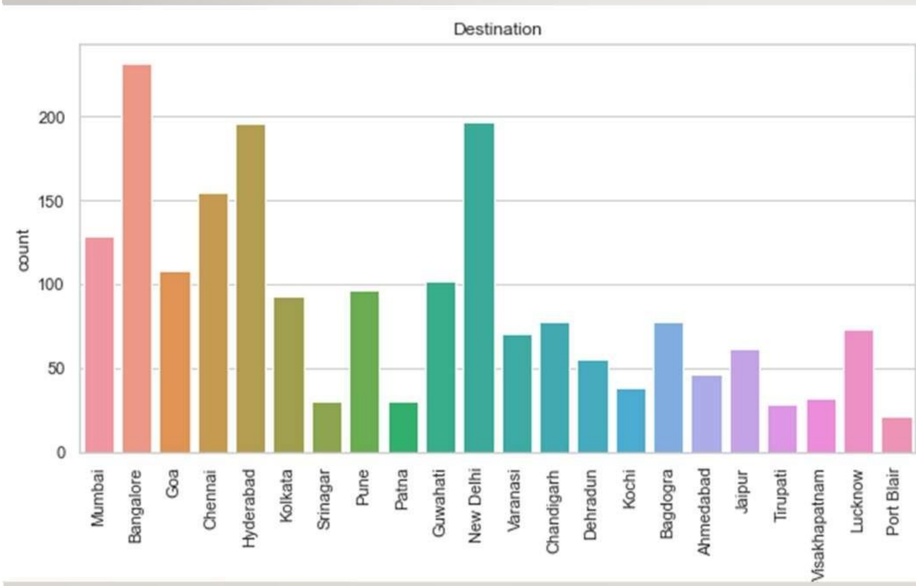


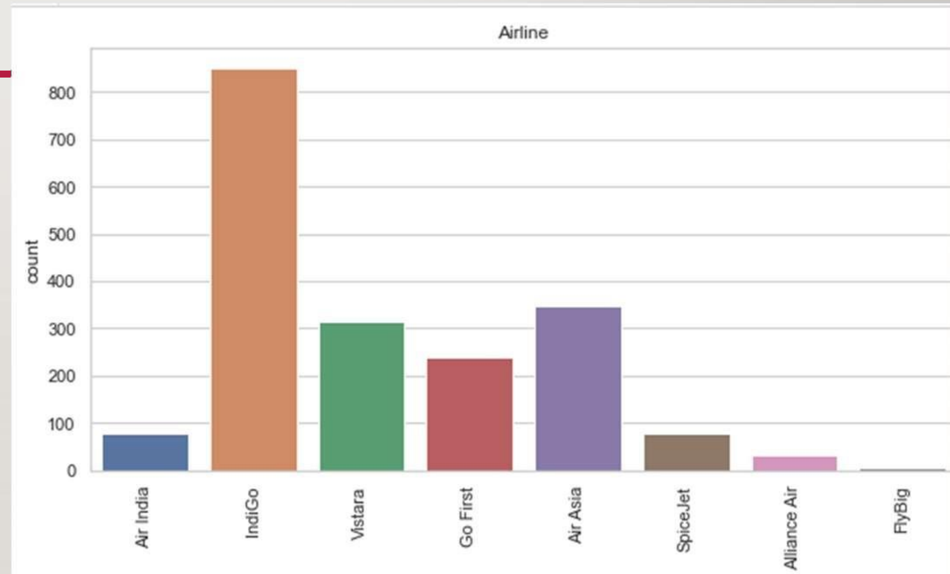
Most of the flight's source is Mumbai, followed by Bangalore and New Delhi.



Maximum flights are having 0 stop only followed by one stop.

ANALYSING CATEGORICAL FEATURES



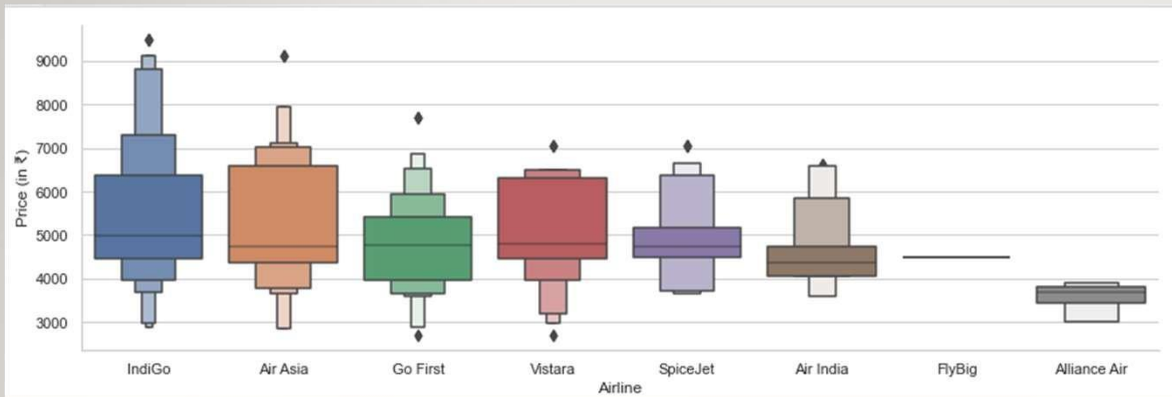


maximum flights have destination as Bangalore followed by New Delhi and Hyderabad.

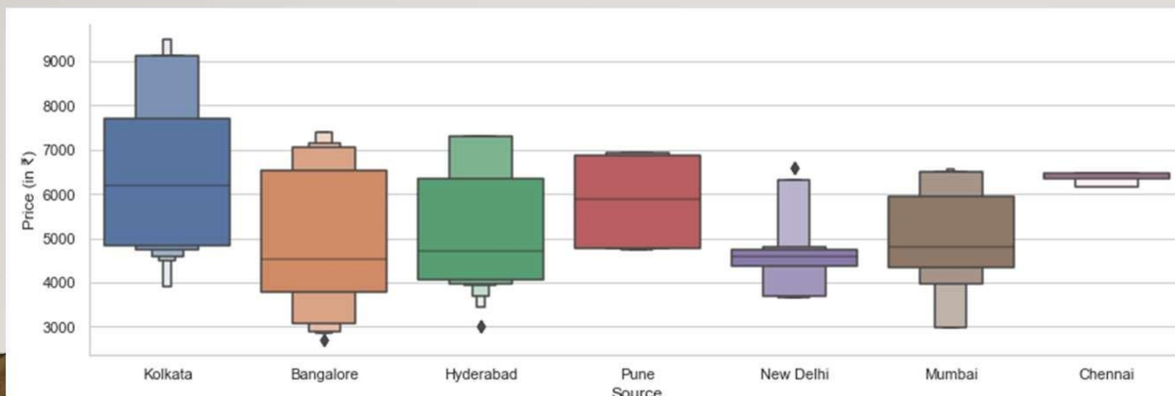
Indigo has the highest number of flights.

BIVARIATE ANALYSIS

Bivariate analysis is finding some kind of empirical relationship between two variables. Specifically, the dependent vs independent Variables

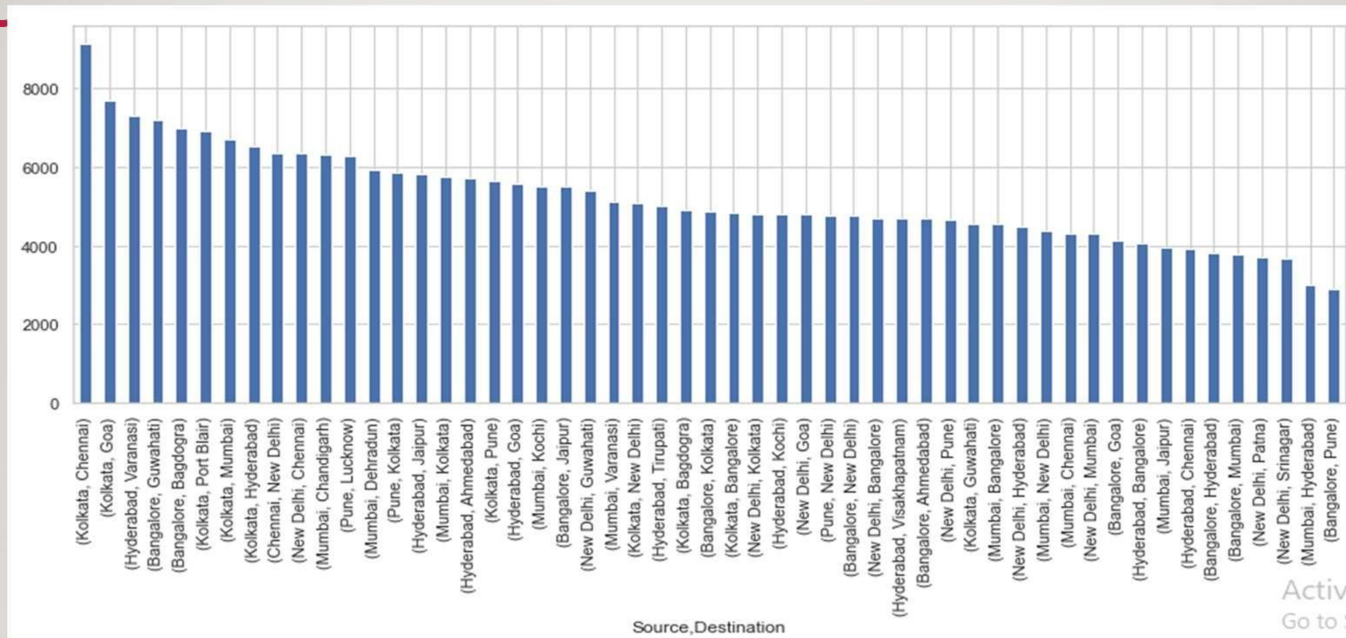


From graph, we can see the relation between Airline and Price.
Indigo has the highest Price.



This graph shows the relation between Source and Price. Flights starting from Kolkata are having highest price and flights starting from Chennai are having the lowest price.

KOLKATA TO CHENNAI AVERAGE PRICE IS RS9500 APPROX., BANGALORE TO PUNE AVERAGE PRICE IS LOWEST WHICH IS AROUND RS 3000 APPROX.



Graphs show the mean price of different source and destinations

LABEL ENCODING

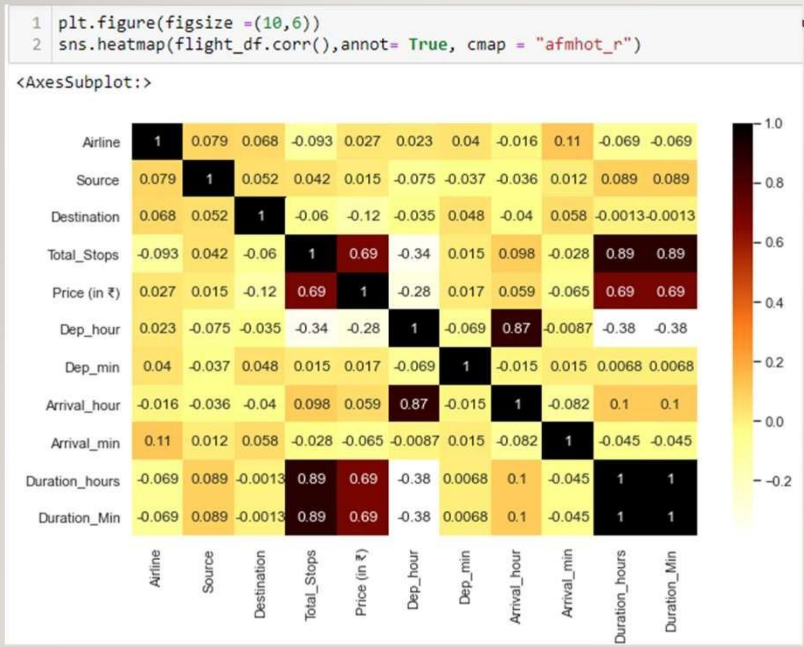
```python
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

flight_df["Airline"] = le.fit_transform(flight_df["Airline"])
flight_df["Source"] = le.fit_transform(flight_df["Source"])
flight_df["Destination"] = le.fit_transform(flight_df["Destination"])
```

```python
flight_df.head()
```

	Airline	Source	Destination	Total_Stops	Price (in ₹)	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_Min
0	1	5	13	0	4065	7	0	9	5	2	2
1	1	5	13	0	4065	8	0	10	10	2	2
2	1	5	13	0	4065	9	0	11	15	2	2
3	1	5	13	0	4065	14	0	16	15	2	2
4	1	5	13	0	4065	21	15	23	35	2	2

Converted all the categorical columns to numerical columns using a Label encoder

CHECKING THE MULTICOLLINEARITY



Observation:

- Total_stops ,Duration Minutes and Duration_hours have positive correlation with Target column.
- Total_stops and Duration hours are also correlation but we will keep the same in the dataset because there are only two which reflect maximum variance.

CHECK FOR SKEWNESS

- Skewness is removed using Log transform.

```
# Cheking Skewness
x.skew().sort_values(ascending=False)
```

```
Duration_hours     1.625229
Duration_Min       1.625229
Total_Stops        1.362949
Destination        0.302640
Dep_hour           0.170535
Arrival_min        0.031550
Dep_min           -0.006688
Arrival_hour      -0.081766
Source            -0.321417
Airline           -0.781667
dtype: float64
```

```
# Again Cheking Skewness if it has been removed
x.skew().sort_values(ascending=False)
```

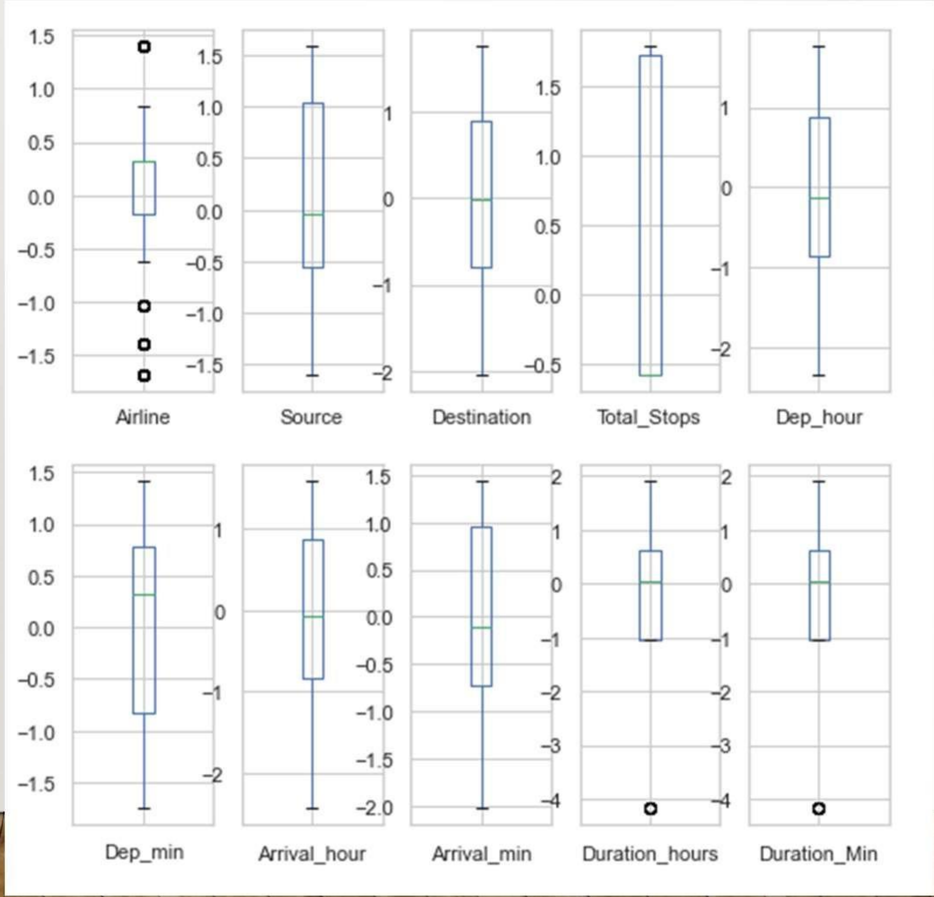
```
Total_Stops        1.146249
Duration_hours     0.126515
Duration_Min       0.126515
Dep_hour          -0.104519
Arrival_hour      -0.122206
Destination       -0.136892
Arrival_min       -0.258910
Source            -0.296870
Dep_min           -0.389752
Airline           -0.514544
dtype: float64
```

```
from sklearn.preprocessing import power_transform
x_new=power_transform(x)
```

We can see the skewness is removed.

OUTLIERS CHECK USING BOX PLOT

There is not much outliers in numerical columns . Very Few are there in categorical columns which don't need to be removed.



Feature Scaling

```
# Performing Standard scaler
sc = StandardScaler()
X = sc.fit_transform(x)
```

By using a standard scaler, I have scaled the data in one range.

```
X
```

```
array([[-1.39481233,  1.04114663,  0.76455477, ..., -1.48966862,
         0.04904176,  0.04904176],
       [-1.39481233,  1.04114663,  0.76455477, ..., -1.08414886,
         0.04904176,  0.04904176],
       [-1.39481233,  1.04114663,  0.76455477, ..., -0.73034527,
         0.04904176,  0.04904176],
       ...,
       [ 0.31866501, -0.03312358,  1.29054104, ...,  0.71177723,
         0.04904176,  0.04904176],
       [ 0.31866501, -0.03312358,  1.29054104, ..., -1.08414886,
         0.61147925,  0.61147925],
       [-0.17162813, -0.03312358,  1.29054104, ..., -1.48966862,
         1.66897404,  1.66897404]])
```

Model Building

FINDING THE BEST RANDOM STATE

```
maxScore = 0
maxRS = 0

for i in range(1,300):
    x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=i)
    lr = LinearRegression()
    lr.fit(x_train,y_train)
    pred_train = lr.predict(x_train)
    pred_test = lr.predict(x_test)
    acc=r2_score(y_test,pred_test)
    if acc>maxScore:
        maxScore=acc
        maxRS=i
print('Best score is',maxScore,'on Random State',maxRS)
```

```
Best score is 0.6494313332521702 on Random State 124
```

Applying on 5 different algorithms

- ☐ LinearRegression(),
- ☐ Lasso()
- ☐ Ridge()
- ☐ DecisionTreeRegressor()
- ☐ KNeighborsRegressor()]

TRAIN AND TEST SCORES OF 5 DIFFERENT ALGORITHMS

```python
model = [LinearRegression(),Lasso(alpha=1.0),Ridge(alpha=1.0),DecisionTreeRegressor(criterion='squared_error'),
         KNeighborsRegressor()]
for i in model:
    X_train1,X_test1,y_train1,y_test1 = train_test_split(X,y, test_size = 0.2, random_state =maxRS)
    i.fit(X_train1,y_train1)
    pred = i.predict(X_test1)
    print('Train Score of', i , 'is:' , i.score(X_train1,y_train1))
    print("r2_score", r2_score(y_test1, pred))
    print("mean_squred_error", mean_squared_error(y_test1, pred))
    print("RMSE", np.sqrt(mean_squared_error(y_test1, pred)),"\n")
```

```
Train Score of LinearRegression() is: 0.5648093647501061
r2_score 0.6494313332521702
mean_squred_error 617187.0652469436
RMSE 785.612541426716

Train Score of Lasso() is: 0.5647849750028762
r2_score 0.6493976214073971
mean_squred_error 617246.4160004853
RMSE 785.6503140713974

Train Score of Ridge() is: 0.5648092779789935
r2_score 0.6494509278223357
mean_squred_error 617152.5683953927
RMSE 785.5905857349569

Train Score of DecisionTreeRegressor() is: 0.9972975535231607
r2_score 0.9923616711534731
mean_squred_error 13447.516025641025
RMSE 115.96342537904366

Train Score of KNeighborsRegressor() is: 0.9314382366095284
r2_score 0.7978593093899049
mean_squred_error 355874.9866666667
RMSE 596.5525849970535
```

- Have checked Multiple Model and their score also.
- I have found that Decision tree regressor model is overfitting. Other models are working well.
- But DecisionTreeRegressor() is having less train and test score difference with least mean square error and least RMSE.
- Now i will check with ensemble method to boost up score.

ENSEMBLE TECHNIQUE TO BOOST UP SCORE

- ☐ **Random Forest Regressor:**

```
Train Score of RandomForestRegressor(random_state=124) is: 0.9957430019522039
r2_score 0.9866703484841391
mean_squred_error 23467.266986973325
RMSE 153.19029664757923
```

- ☐ **AdaBoostRegressor:**

```
Train Score of AdaBoostRegressor(base_estimator=DecisionTreeRegressor(), random_state=124) is: 0.9961570396869603
r2_score 0.9944539392101273
mean_squred_error 9764.012894640458
RMSE 98.81301986398583
```

- ☐ **GradientBoostingRegressor:**

```
Train Score of GradientBoostingRegressor() is: 0.9135786715106335
r2_score 0.9003430013998592
mean_squred_error 175449.25240447314
RMSE 418.8666284206384
```

Conclusion:

- Here we can see the least difference between train score and test score is coming in AdaBoostRegressor.So the model is working well with the both train model and the test model.
- For RandomForestRegressor, GradientBoostingRegressor., the difference is little more as compared to AdaBoostRegressor.
- So the model is overfitting.
- **So selecting AdaBoostRegressor as final model**

HYPERPARAMETER TUNING TO FIND BEST PARAMETERS OF ADABOOSTREGRESSOR

```
Ada_Boost = AdaBoostRegressor()
Para ={'n_estimators' : [50, 100, 150, 200],
        'learning_rate' : [0.001, 0.01, 0.1, 1],
        'loss' : ["linear", "square", "exponential"],
        'random_state' : [21, 42, 104, 111]
        }
Ada_search = RandomizedSearchCV(Ada_Boost,Para,cv = 5,scoring = "r2",n_jobs =-1,verbose = 2)
Ada_search.fit(X_train1,y_train1)
print(Ada_search.best_params_)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
{'random_state': 111, 'n_estimators': 50, 'loss': 'linear', 'learning_rate': 1}
```

```
prediction = Ada_search.predict(X_test1)
```

```
FlightPrice = AdaBoostRegressor(n_estimators= 50, loss= 'linear', learning_rate =1, random_state=111)
FlightPrice.fit(x_train, y_train)
pred = FlightPrice.predict(x_test)
print('R2_Score:',r2_score(y_test,pred)*100)
print("RMSE value:",np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2_Score: 75.08080097429422
RMSE value: 667.8138380332518
```

Selecting Cv score as 5

CROSS VALIDATION

```python
best_Ada_Boost = AdaBoostRegressor(n_estimators= 50, loss= 'linear', learning_rate =1, random_state=111)

for i in range(2,11):
    cross_score = cross_val_score(best_Ada_Boost,X,y,cv = i,n_jobs = -1)
    print(i,"mean",cross_score.mean() ,"and STD" , cross_score.std())
```

```
2 mean 0.7079885311697873 and STD 0.03750158080833105
3 mean 0.6896722276958461 and STD 0.08054815896823123
4 mean 0.6978318289655345 and STD 0.08159298797693146
5 mean 0.6807099102493558 and STD 0.09614381233553472
6 mean 0.6702078159005903 and STD 0.09881004588221423
7 mean 0.6332033699475218 and STD 0.15776782167766093
8 mean 0.6339620102070194 and STD 0.17464143798158782
9 mean 0.6189491793496691 and STD 0.18593627306154722
10 mean 0.6326366873127885 and STD 0.19414716310795876
```

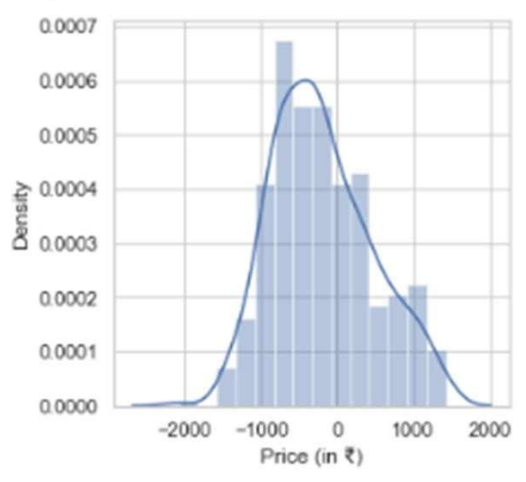
Selecting Cv score as 5

```python
# Cross validate of AdaBoostRegressor using cv=5
from sklearn.model_selection import cross_val_score
score=cross_val_score(best_Ada_Boost,X,y,cv=5,scoring='r2')
print('Score:', score)
print('Mean Score:', score.mean())
print('Standard Deviation:', score.std())
```

```
Score: [0.51750106 0.6563048  0.69478879 0.72429181 0.81066309]
Mean Score: 0.6807099102493558
Standard Deviation: 0.09614381233553472
```

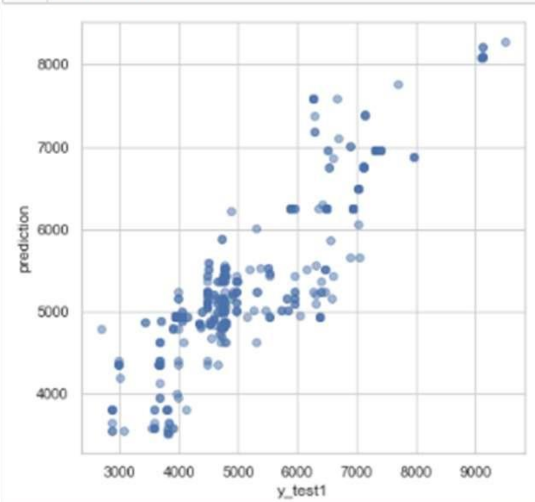
PLOTTING THE RESIDUALS.

```
plt.figure(figsize = (4,4))
sns.distplot(y_test1-prediction)
plt.show()
```



Plotting y_test vs predictions.

```
plt.figure(figsize = (6,6))
plt.scatter(y_test1, prediction, alpha = 0.5,)
plt.xlabel("y_test1")
plt.ylabel("prediction")
plt.show()
```



CONCLUSION:

- After Scraping Flight Ticket prices for different source and destination cities like Delhi, Mumbai, Hyderabad, Bangalore, Chennai, and Kolkata from different websites like Yatra.com, I have prepared an excel sheet and loaded the dataset for further EDA process.
- So, as we saw that we have done a complete EDA process, getting data insights, feature engineering, and data visualization as well so after all these steps one can go for the prediction using machine learning model-making steps.
- We have all the features of categorical data types in the datasets and the dependent variable i.e. Price is also an object data type. **I am changing the target column to an integer type and** I applied the regression method for prediction.
- Once data has been cleaned and missing value is replaced, Label encoding is applied to them to convert them into Numerical ones. I trained the model on five different algorithms but for most of the models, train and test data was having a variance, and the model was overfitting.
- Only Ada BoostRegressor worked well out of all the models, as there was less difference between train score and test score and RMSE was also low hence I used it as the final model and have done further processing.
- After applying hyperparameter tuning I got an accuracy(r2_score) of 975% from the AdaBoostRegressor model after hyper parameter tuning which is a good score.

Then I saved the model.

LIMITATIONS AND SCOPE

- This study used only Yatra.com for web scraping. More websites can give more ideas and accurate reading. However, there was a relatively small dataset for making a strong inference because number of observations was only 1948. Gathering more data can yield more robust predictions.
- Secondly, there could be more features that can be good predictors. For example, here are some variables that might improve the model: Date of Journey, meal Details
- Another point that has room to improve is that the data cleaning process can be done more rigorously with the help of more technical information. For example, I had to drop meal info column because of lack of data..
- As a suggestion for further studies, while pre-processing data, instead of using a label encoder, one hot encoder method can be used. Thus, all non-numeric features can be converted to nominal data instead of ordinal data.

Thank You