FLIP ROBO

Flight Price Prediction

Submitted by:

Subhashchandra Pandey

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my SME Mr. Shwetank Mishra from Flip Robo Technologies for letting me work on this project and providing me with all necessary information and dataset for the project. I would also like to thank my mentor of Data Trained academy for providing me sufficient knowledge so that I can complete the project.

I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

- https://www.google.com/
- https://www.youtube.com/
- https://scikit-learn.org/stable/user_guide.html
- https://github.com/
- https://www.analyticsvidhya.com/

INTRODUCTION

- Business Problem Framing

The airline industry is considered as one of the most sophisticated industries in using complex pricing strategies. Nowadays, ticket prices can vary dynamically and significantly for the same flight, even for nearby seats. The ticket price of a specific flight can change up to 7 times a day. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit. However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company losing revenue. Airlines companies are generally equipped with advanced tools and capabilities that enable them to control the pricing process. However, customers are also becoming more strategic with the development of various online tools to compare prices across various airline companies. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone.

Business goal: The main aim of this project is to predict the price of flight tickets based on various features. The purpose of the paper is to study the factors which influence the fluctuations in the airfare prices and how they are related to the change in the prices. Then using this information, build a system that can help buyers whether to buy a ticket or not. So, we will deploy a Machine Learning model for flight ticket price prediction and analysis. This model will provide the approximate selling price for the flight tickets based on different features.

- Conceptual Background of the Domain Problem

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travellers saying that flight ticket prices are so unpredictable.
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 to reduce sales and hold back inventory for those expensive last-minute expensive purchases).

Here we are trying to help the buyers to understand the price of the flight tickets by deploying machine learning models. These models would help the sellers/buyers to understand the flight ticket prices in market and accordingly they would be able to book their tickets.

- Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict the flight ticket prices in the market. In this study, we discuss various applications and methods which inspired us to build our supervised ML techniques to predict the price of flight tickets in different locations. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of data information by doing web scraping from www.yatra.com website which is a web platform where buyers can book their flight tickets.

This project is more about data exploration, feature engineering and pre-processing that can be done on this data. Since we scrape huge amount of data that includes more flight related features, we can do better data exploration and derive some interesting features using the available columns. Different techniques like ensemble techniques, and decision trees have been used to make the predictions.

The goal of this project is to build an application which can predict the price of flight tickets with the help of other features. In the long term, this would allow people to better explain and reviewing their purchase in this increasing digital world.

- Motivation for the Problem Undertaken

Air travel is the fastest method of transport around and can cut hours or days off a trip. But we know how unexpectedly the prices vary. So, I was interested in Flight Fares Prediction listings to help individuals and find the right fares based on their needs. And, to get hands on experience and to know that how the data scientist approaches and work in an industry end to end.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

We need to develop an efficient and effective Machine Learning model which predicts the price of flight tickets. So, "Price" is our target variable which is continuous in nature. Clearly it is a Regression problem where we need to use regression algorithms to predict the results. This project is done on three phases:

• **Data Collection Phase:** I have done web scraping to collect the data of flights from the well-known website https://www.yatra.com where I found more features of flights compared to other websites and I fetch data for different locations. As per the requirement we need to build the model to predict the prices of flight tickets.

• **Data Analysis:** After cleaning the data I have done some analysis on the data by using different types of visualizations.

• **Model Building Phase:** After collecting the data, I built a machine learning model. Before model building, have done all data pre-processing steps. The complete life cycle of data science that I have used in this project are as follows:

- Data Cleaning
- Exploratory Data Analysis

- Data Pre-processing
- Model Building
- Model Evaluation
- Selecting the best model

- Data Sources and their formats

We have collected the dataset from the website https://www.yatra.com which is a web platform where the people can purchase/book their flight tickets. The data is scraped using Web scraping technique and the framework used is Selenium. We scrapped nearly 1918 rows of the data and fetched the data for different locations and collected the information of different features of the flights and saved the collected data in excel format. The dimension of the dataset is 1918 rows and 9 columns including target variable “Price”. The dataset contains both categorical and numerical data type. The data description is as follows:

- Airline : This shows the list of all the Airline Names for which the data got scraped
- Source : Gives us the name of the source place where the flight journey began
- Destination : Shows us the name of the destination place where the flight journey ended
- Dep_Time : In this column we have the timings of every flight departure
- Arrival_Time : Here in this column we have the timings of every flight arrival
- Duration : We can see the total duration of a flight that it took to fly from the source to the destination
- Total_Stops : Lists the number of stops the flight is going to take to complete the entire journey
- Additional_Info : Provides us with any additional information like type of meal that the passenger is eligible for

- Price : Finally we have our label column that has the ticket prices for the aircraft journey

• Data Preprocessing Done

Data pre-processing is the process of converting raw data into a well-readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

➢ Importing necessary libraries and loading collected dataset as a data frame.

➢ Checked some statistical information like shape, number of unique values present, info, unique (), data types, value count function etc.

➢ Checked null values and found some missing values on column “Meal_Availability” and filled the null values by using mode method.

➢ Taking care of Timestamp variables by converting data types of “Dep_Time” and “Arrival_Time” from object data type into datetime data types.

➢ Done feature engineering on some features as they had some irrelevant values like “,”, “:” and replaced them by empty space.

➢ The column Duration had values in terms of minutes and hours. Duration means the time taken by the plane to reach the destination and it is the difference between the arrival time and Departure time. So, I have extracted proper duration time in terms of float data type from arrival and departure time columns.

➢ Extracted Departure_Hour, Deparutre_Min and Arrival_Hour, Arrival_Min columns from Dep_time and Arrival_Time columns and dropped these columns after extraction.

➢ The target variable "price" should be continuous numeric data but due to some string values like “,” it was showing as object data type. So, I replaced this sign by empty space and converted into float data type.

➢ From the value count function of Total_Stops, I found categorical data so

replaced them with numeric data according to stops.

➢ Checked statistical description of the data and separated categorical and numeric features.

➢Visualized each feature using seaborn and matplotlib libraries by plotting several categorical and numerical plots.

➢ Identified outliers using box plots.

➢ Checked for skewness and removed skewness in numerical column "Duration" using square root transformation method.

➢ Encoded the columns having object data type using Label Encoder method. Used Pearson's correlation coefficient to check the correlation between label and features. With the help of heatmap and correlation bar graph was able to understand the Feature vs Label relativity.

➢ Separated feature and label data and feature scaling is performed using Standard Scaler method to avoid any kind of data biasness.

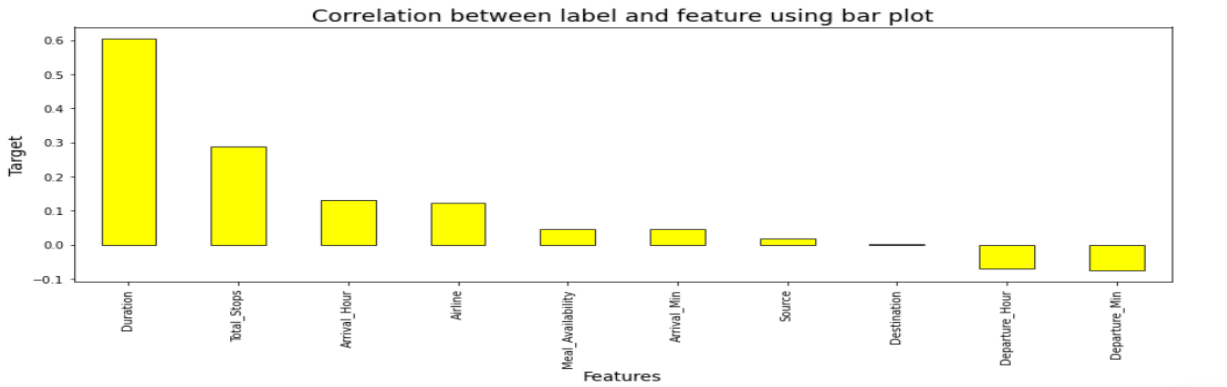
- Data Inputs- Logic- Output Relationships

The dataset consists of label and features. The features are independent, and label is dependent as the values of our independent variables changes as our label varies.

• Since we had both numerical and categorical columns, I checked the distribution of skewness using dist plots for numerical features and checked the counts using count plots & pie plots for categorical features as a part of univariate analysis.

• To analyse the relation between features and label I have used many plotting techniques where I found numerical continuous variables having some relation with label Price with the help of categorical and line plot.

• I have checked the correlation between the label and features using heat map and bar plot. Where I got both positive and negative correlation between the label and features. Below is the bar graph to know the correlation between features and label.


Correlation between label and feature using bar plot
Target
0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
Duration
Total_Stops
Arrival_Hour
Airline
Meal_Availability
Arrival_Min
Source
Destination
Departure_Hour
Departure_Min
Features

- Hardware and Software Requirements and Tools Used

Hardware Used:

1. Processor — core i5 and above
2. RAM — 8 GB or above
3. SSD — 250GB or above

Software Used:

i. Programming language: Python
ii. Distribution: Anaconda Navigator
iii. Browser based language shell: Jupyter Notebook
Libraries/Packages Used:
Pandas, NumPy, matplotlib, seaborn, scikit-learn and pandas_profiling

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

I. I have used imputation methods to treat the null values.

II. Used percentile method to remove outliers.

III. Removed skewness using power transformation (yeo-johnson) method.

IV. Encoded the object type data into numerical using Ordinal Encoder.

V. I have used Pearson's correlation coefficient method to check the correlation between the dependent and independent variables.

VI. I have scaled the data using Standard Scalar method to overcome with the data biasness.

VII. Used many Machine Learning models to predict the flight price.

- Testing of Identified Approaches (Algorithms)

Since “Price” is my target variable, which is continuous in nature, from this I can conclude that it is a regression type problem hence I have used following regression algorithms. After the pre-processing and data cleaning I left with 11 columns including target and with the help of feature importance bar graph I used these independent features for model building and prediction. The algorithms used on training the data are as follows:

1. Decision Tree Regressor
2. Random Forest Regressor
3. Extra Trees Regressor
4. Gradient Boosting Regressor
5. Extreme Gradient Boosting Regressor (XGB)
6. Bagging Regressor
7. KNeighbors Regressor

- Run and Evaluate selected models

I used a total of 7 Regression Models after choosing the random state amongst 1-200 number. Then I have defined a function for getting the regression model trained and evaluated.
The code for the models is listed below.

Random State:

```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
    mod = RandomForestRegressor()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.9906915893868445  on Random_state  16
```

Here we are getting accuracy score of 99.06% with Random state 16

Decision Tree Regressor

```
# Checking R2 score for Decision Tree Regressor
DTR=DecisionTreeRegressor()
DTR.fit(x_train,y_train)

# prediction
predDTR=DTR.predict(x_test)
R2_score = r2_score(y_test,predDTR)*100
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predDTR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predDTR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predDTR)))

# Cross Validation Score
cv_score = (cross_val_score(DTR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)
# Visualizing the predicted values
sns.regplot(y_test,predDTR,color="g")
plt.show()
```

```
R2_Score: 99.71021661520835
Mean Absolute Error: 8.29225352112676
Mean Squared Error: 7903.866197183099
Root Mean Squared Error: 88.90369057121926

Cross Validation Score: 97.54349674369239

R2 Score - Cross Validation Score is 2.1667198715159657
```

Random Forest Regressor

```
# Checking R2 score for Random Forest Regressor
RFR=RandomForestRegressor()
RFR.fit(x_train,y_train)

# prediction
predRFR=RFR.predict(x_test)
R2_score = r2_score(y_test,predRFR)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predRFR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predRFR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predRFR)))

# Cross Validation Score
cv_score = (cross_val_score(RFR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predRFR,color="g")
plt.show()
```

```
R2_Score: 98.93575611599407
Mean Absolute Error: 79.7098943661972
Mean Squared Error: 29027.341461971828
Root Mean Squared Error: 170.37412204314313

Cross Validation Score: 97.94166725451832

R2 Score - Cross Validation Score is 0.9940888614757455
```

Extra Trees Regressor

```
# Checking R2 score for Extra Trees Regressor
XT=ExtraTreesRegressor()
XT.fit(x_train,y_train)

# prediction
predXT=XT.predict(x_test)
R2_score = r2_score(y_test,predXT)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predXT))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predXT))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predXT)))

# Cross Validation Score
cv_score = (cross_val_score(XT, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predXT,color="g")
plt.show()
```

```
R2_Score: 99.57574216271094
Mean Absolute Error: 29.45258802816902
Mean Squared Error: 11571.668201232394
Root Mean Squared Error: 107.57168866031803

Cross Validation Score: 98.35182978915877

R2 Score - Cross Validation Score is 1.2239123735521673
```

GradientBoosting Regressor

```
# Checking R2 score for GradientBoosting Regressor
GB=GradientBoostingRegressor()
GB.fit(x_train,y_train)

# prediction
predGB=GB.predict(x_test)
R2_score = r2_score(y_test,predGB)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predGB))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predGB))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predGB)))

# Cross Validation Score
cv_score = (cross_val_score(GB, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)
# Visualizing the predicteed values
sns.regplot(y_test,predGB,color="g")
plt.show()
```

```
R2_Score: 91.16050443153483
Mean Absolute Error: 322.9841931630887
Mean Squared Error: 241097.98522083394
Root Mean Squared Error: 491.01729625425

Cross Validation Score: 88.42466447399362

R2 Score - Cross Validation Score is 2.7358399575412164
```

Extreme Gradient Boosting Regressor (XGB)

```
# Checking R2 score for XGB Regressor
from xgboost import XGBRegressor as xgb
XGB=xgb(verbosity=0)
XGB.fit(x_train,y_train)

# prediction
predXGB=XGB.predict(x_test)
R2_score = r2_score(y_test,predXGB)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predXGB))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predXGB))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predXGB)))

# Cross Validation Score
cv_score = (cross_val_score(XGB, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predXGB,color="g")
plt.show()
```

```
R2_Score: 99.63275731191581
Mean Absolute Error: 27.428464003012213
Mean Squared Error: 10016.575210473775
Root Mean Squared Error: 100.0828417386006

Cross Validation Score: 98.72992050426724

R2 Score - Cross Validation Score is 0.9028368076485691
```

Bagging Regressor

```
# Checking R2 score for BaggingRegressor
BR=BaggingRegressor()
BR.fit(x_train,y_train)

# prediction
predBR=BR.predict(x_test)
R2_score = r2_score(y_test,predBR)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predBR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predBR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predBR)))
# Cross Validation Score
cv_score = (cross_val_score(BR, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predBR,color="g")
plt.show()
```

```
R2_Score: 98.84408561732411
Mean Absolute Error: 80.33433098591551
Mean Squared Error: 31527.662024647896
Root Mean Squared Error: 177.56030531807468

Cross Validation Score: 96.6421452793384

R2 Score - Cross Validation Score is 2.201940337985718
```

KNeighbors Regressor

```
# Checking R2 score for KNeighborsRegressor
from sklearn.neighbors import KNeighborsRegressor as KNN
knn=KNN()
knn.fit(x_train,y_train)

# prediction
predknn=knn.predict(x_test)
R2_score = r2_score(y_test,predknn)*100      # R squared score
print('R2_Score:',R2_score)
# Evaluation Metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predknn))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predknn))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predknn)))

# Cross Validation Score
cv_score = (cross_val_score(knn, x, y, cv=5).mean())*100
print("\nCross Validation Score:", cv_score)

# Difference of R2 score minus cv scores
Difference = R2_score - cv_score
print("\nR2 Score - Cross Validation Score is", Difference)

# Visualizing the predicteed values
sns.regplot(y_test,predknn,color="g")
plt.show()
```

```
R2_Score: 65.12732879578095
Mean Absolute Error: 624.5369718309859
Mean Squared Error: 951155.0406338029
Root Mean Squared Error: 975.2717778310838

Cross Validation Score: 61.77160539647982

R2 Score - Cross Validation Score is 3.3557233993011337
```

Model Selection:

From the above created models, we can conclude that “XGB Regressor” as the best fitting model as it is giving high R2 score and low MAE, MSE and RMSE values. Then I tried to increase our model score by tuning the best model using different types of hyper parameters.

Hyper Parameter Tuning:

```
from sklearn.model_selection import GridSearchCV
```

```
#XGB Regressor
parameters = {'n_estimators' : [50,100,150,200],
              'learning_rate':np.arange(0.05,0.5,0.05),
              'gamma' : np.arange(0,0.5,0.1),
              'max_depth' : [4, 6, 8,10]}
```

```
GCV=GridSearchCV(xgb(),parameters,cv=5)
```

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

```
GridSearchCV(cv=5,
             estimator=XGBRegressor(base_score=None, booster=None,
                                    callbacks=None, colsample_bylevel=None,
                                    colsample_bynode=None,
                                    colsample_bytree=None,
                                    early_stopping_rounds=None,
                                    enable_categorical=False, eval_metric=None,
                                    gamma=None, gpu_id=None, grow_policy=None,
                                    importance_type=None,
                                    interaction_constraints=None,
                                    learning_rate=None, max_bin=None,
                                    max_cat...
                                    min_child_weight=None, missing=nan,
                                    monotone_constraints=None, n_estimators=100,
                                    n_jobs=None, num_parallel_tree=None,
                                    predictor=None, random_state=None,
                                    reg_alpha=None, reg_lambda=None, ...),
             param_grid={'gamma': array([0. , 0.1, 0.2, 0.3, 0.4]),
                         'learning_rate': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45]),
                         'max_depth': [4, 6, 8, 10],
                         'n_estimators': [50, 100, 150, 200]})
```

```
# Finding best parameters
GCV.best_params_
```

```
{'gamma': 0.1,
 'learning_rate': 0.15000000000000002,
 'max_depth': 10,
 'n_estimators': 200}
```

```
# Creating final model
Flight_price_model = xgb(gamma=0.1, learning_rate=0.15000000000000002, max_depth=10, n_estimators=200)

# Prediction
Flight_price_model.fit(x_train, y_train)
pred = Flight_price_model.predict(x_test)
print('R2_Score:',r2_score(y_test,pred)*100)

# Metric Evaluation
print('Mean absolute error:',metrics.mean_absolute_error(y_test, pred))
print('Mean squared error:',metrics.mean_squared_error(y_test, pred))
print('Root Mean Squared error:',np.sqrt(metrics.mean_squared_error(y_test, pred)))

# Visualizing the predicted values
sns.regplot(y_test,pred,color="crimson")
plt.show()
```

```
R2_Score: 99.76479925175724
Mean absolute error: 18.282492839114767
Mean squared error: 6415.120193743029
Root Mean Squared error: 80.09444546123676
```

Saving the Final model

```
# Saving the model using joblib library
import joblib
joblib.dump(Flight_price_model,"FlightPricePrediction.pkl")
```

```
['FlightPricePrediction.pkl']
```

Loading the saved model and predicting Flight price

```
# Loading the saved model
Model=joblib.load("FlightPricePrediction.pkl")

#Prediction
prediction = Model.predict(x_test)
prediction
```

```
array([ 7413.2227,  8948.91  ,  4687.039 ,  7948.634 ,  6716.9287,
        5259.9365,  8992.24  ,  7634.802 ,  7422.9927,  6324.0684,
        7425.035 ,  5942.671 ,  7489.882 ,  5983.6226,  5952.1357,
        7423.9985,  2870.1104,  7948.867 ,  2521.991 ,  8515.992 ,
        4687.152 ,  6506.719 ,  8992.05  ,  6506.929 ,  3422.9993,
        5373.7925,  7564.928 ,  9831.951 ,  4277.8354,  6506.5566,
        9528.819 ,  6372.468 ,  7412.949 ,  8991.898 ,  5952.1816,
        7964.189 ,  4668.0107,  4686.0493,  7684.002 ,  5114.6685,
        6716.8877,  8885.671 ,  5371.893 ,  4686.9653,  5935.6216,
        3814.2585,  7537.4463,  6577.97  ,  5373.9697,  4274.8687,
        6243.487 ,  5932.186 ,  5944.0347,  2433.121 ,  7423.7046,
```

```
Predicted_Flight_Ticket_Price = pd.DataFrame([Model.predict(x_test)[:],y_test[:]],index=["Predicted","Actual"]).T
Predicted_Flight_Ticket_Price
```

	Predicted	Actual
0	7413.222656	7413.0
1	8948.910156	8949.0
2	4687.039062	4687.0
3	7948.633789	7949.0
4	6716.928711	6717.0
...	...	...
563	6370.316406	6364.0
564	7570.982422	7513.0
565	5954.242188	5954.0
566	6926.932617	6927.0
567	5941.911133	5942.0

568 rows × 2 columns

- Key Metrics for success in solving problem under Consideration

The key metrics used here were r2_score, Cross Validation Score, MAE, MSE and RMSE. We tried to find out the best parameters and to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:
Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the

2. R2 Score:
It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE):
MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):
MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are

different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.
We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically.

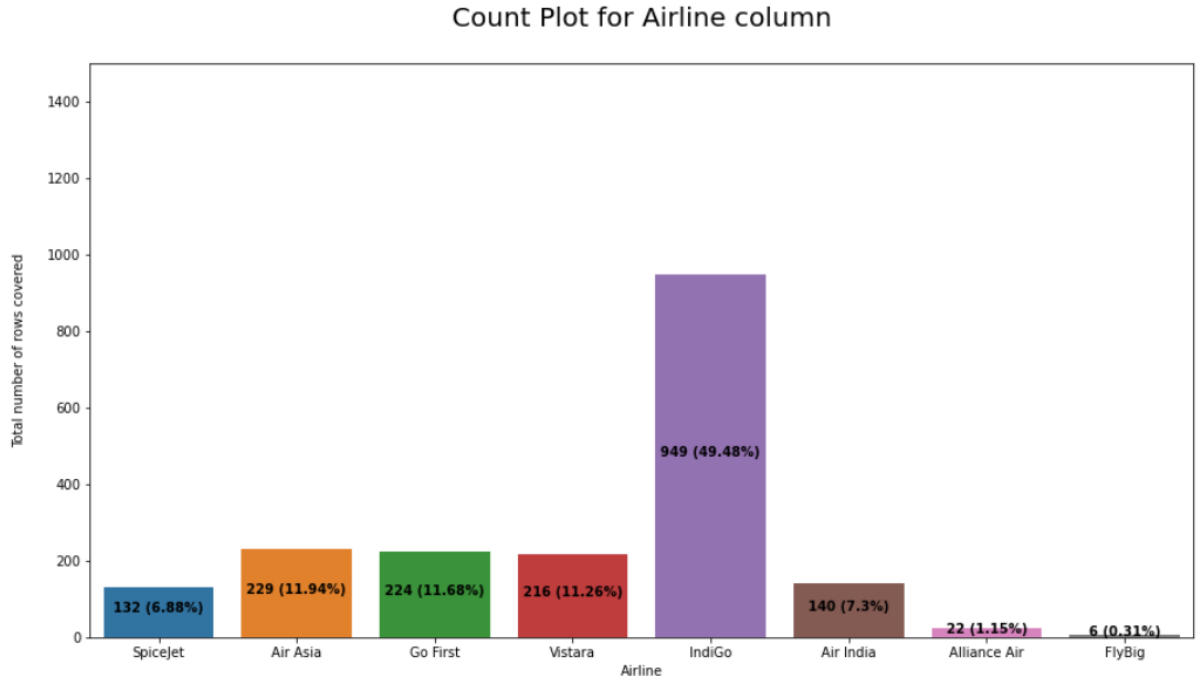
6.GridSearchCV:

GridSearchCV is a function that comes in Scikit-learn (or SK-learn) model selection package. An important point here to note is that we need to have Scikit-learn library installed on the computer. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.
It is possible that there are times when the default parameters perform better than the parameters list obtained from the tuning and it only indicates that there are more permutations and combinations that one needs to go through for obtaining better results.

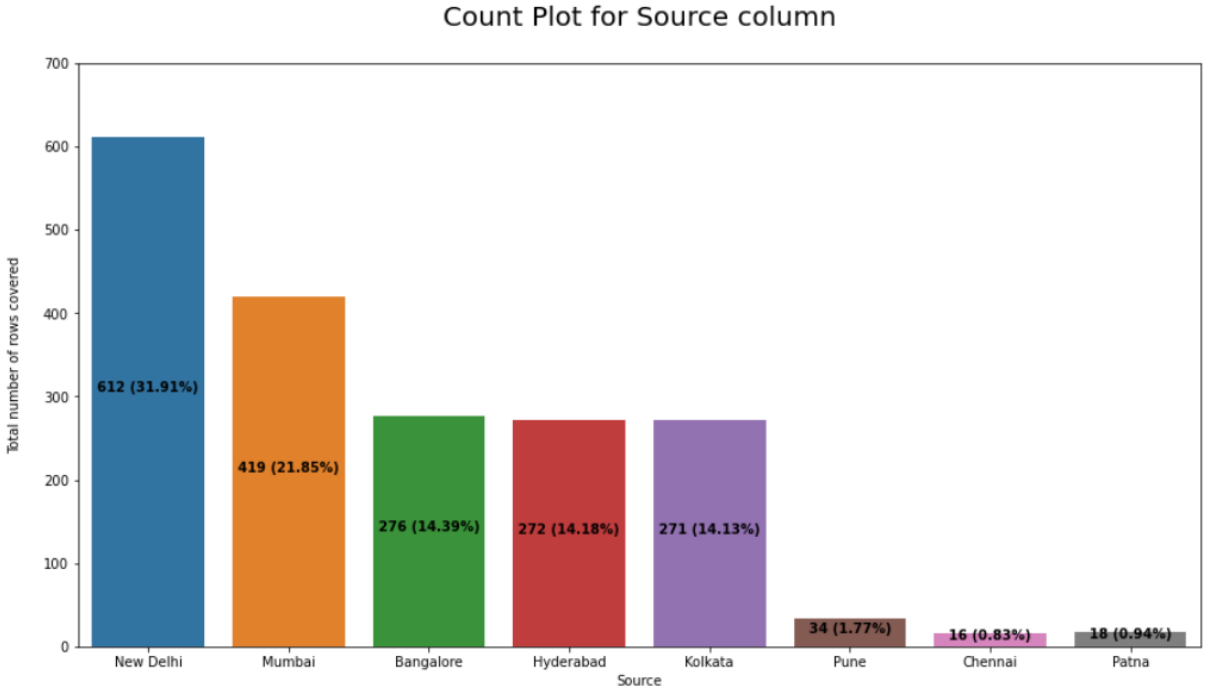
- Visualizations

I have used the pandas profiling feature to generate an initial detailed report on my dataframe values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

I generated count plots, bar plots, pair plots, heatmap and others to visualise the datapoint present in our column records.



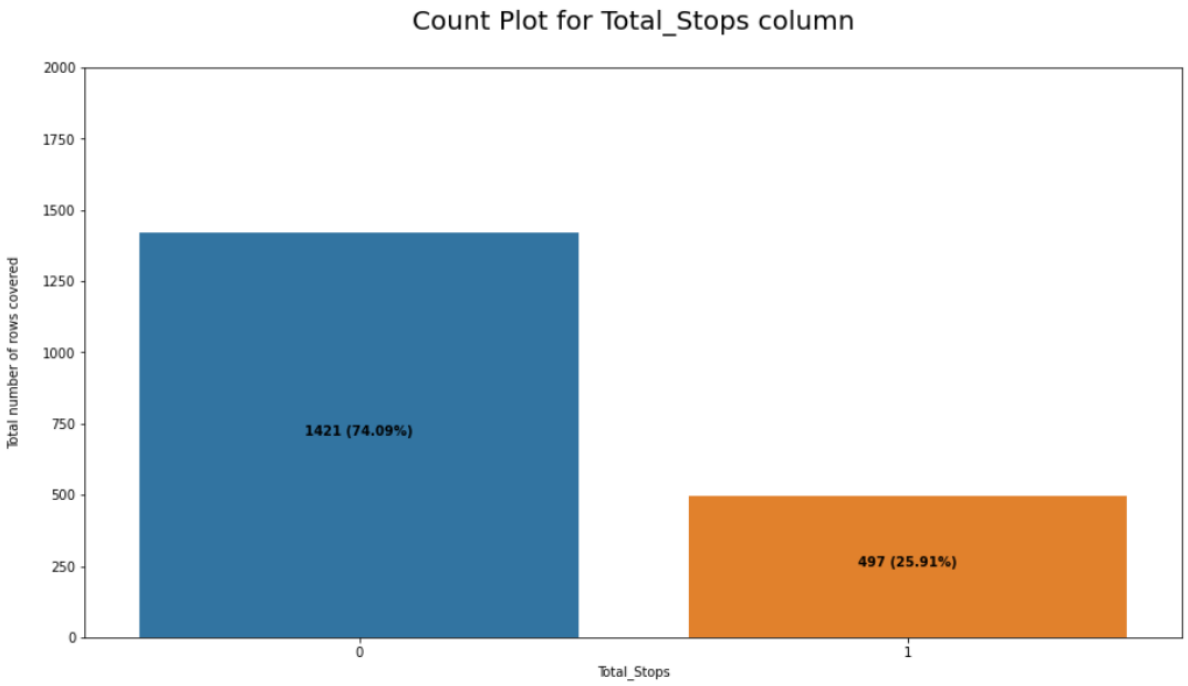
Highest number of airline preferred by people are Indigo covering 49.48% of the total record. Air Asia, Go First and Vistara and similar in range. FlyBig has the lowest numbers.



- The departure area or source place highly used or people majorly

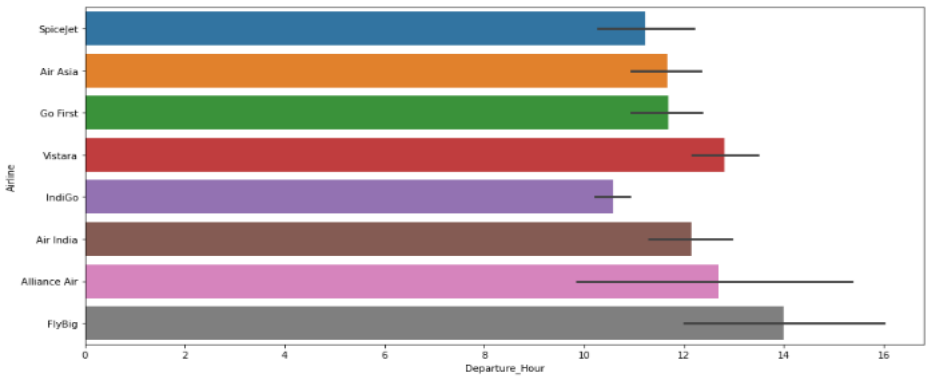
flying from the city is "New Delhi" covering 31.91% record in the column

- We see that "Mumbai" is a close second wherein it covers 21.85% records in the column
- Other two famous locations where people chose to fly from are "Bangalore", "Hyderabad" and "Kolkata"
- The least travel from location is "Chennai"

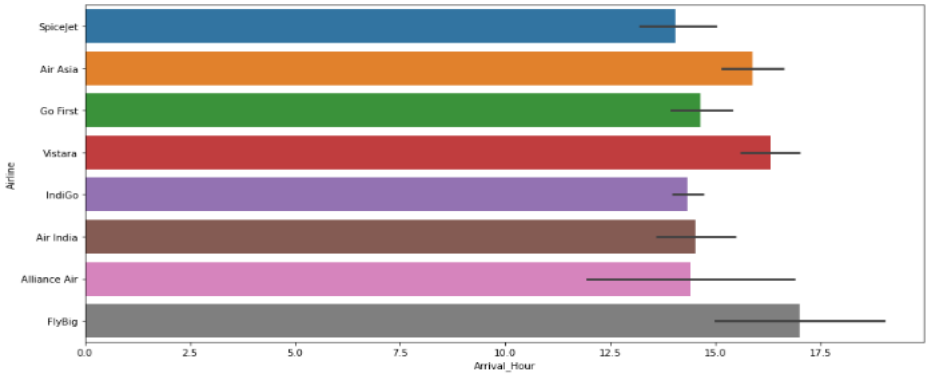


- Majority of the cases, everyone flies with direct flight and followed by 1 stop flight. No one prefers flight with more than 1 stop.

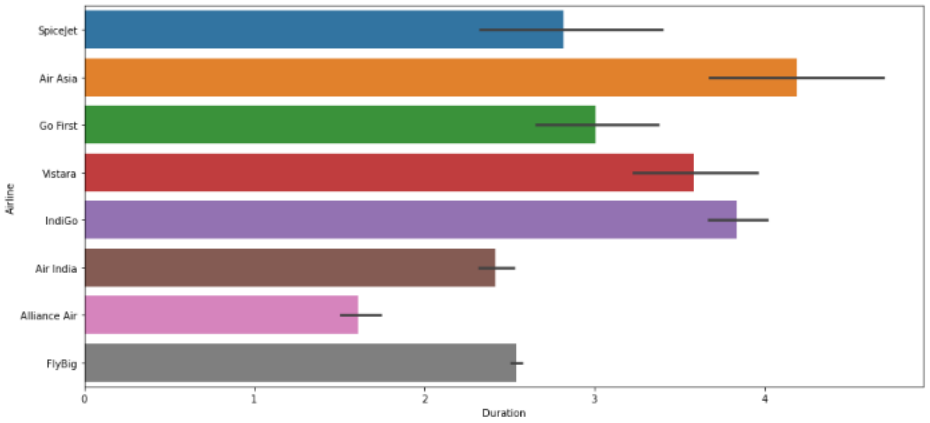
Barplot for Departure_Hour column vs Airline column



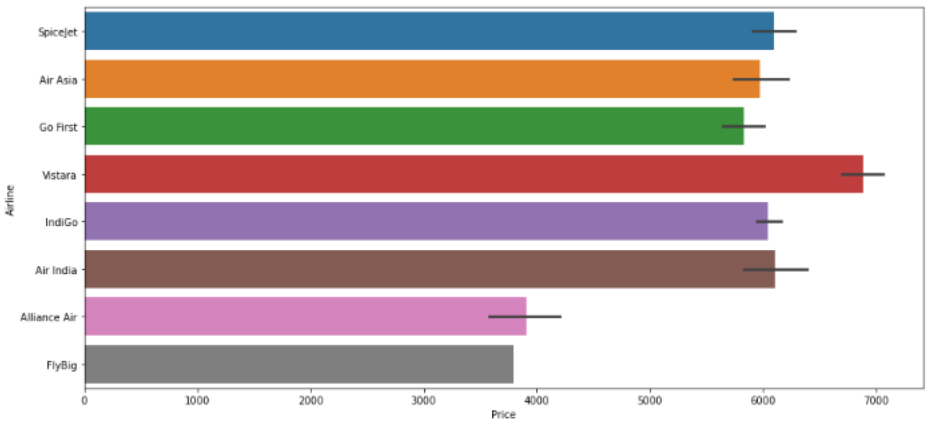
Barplot for Arrival_Hour column vs Airline column



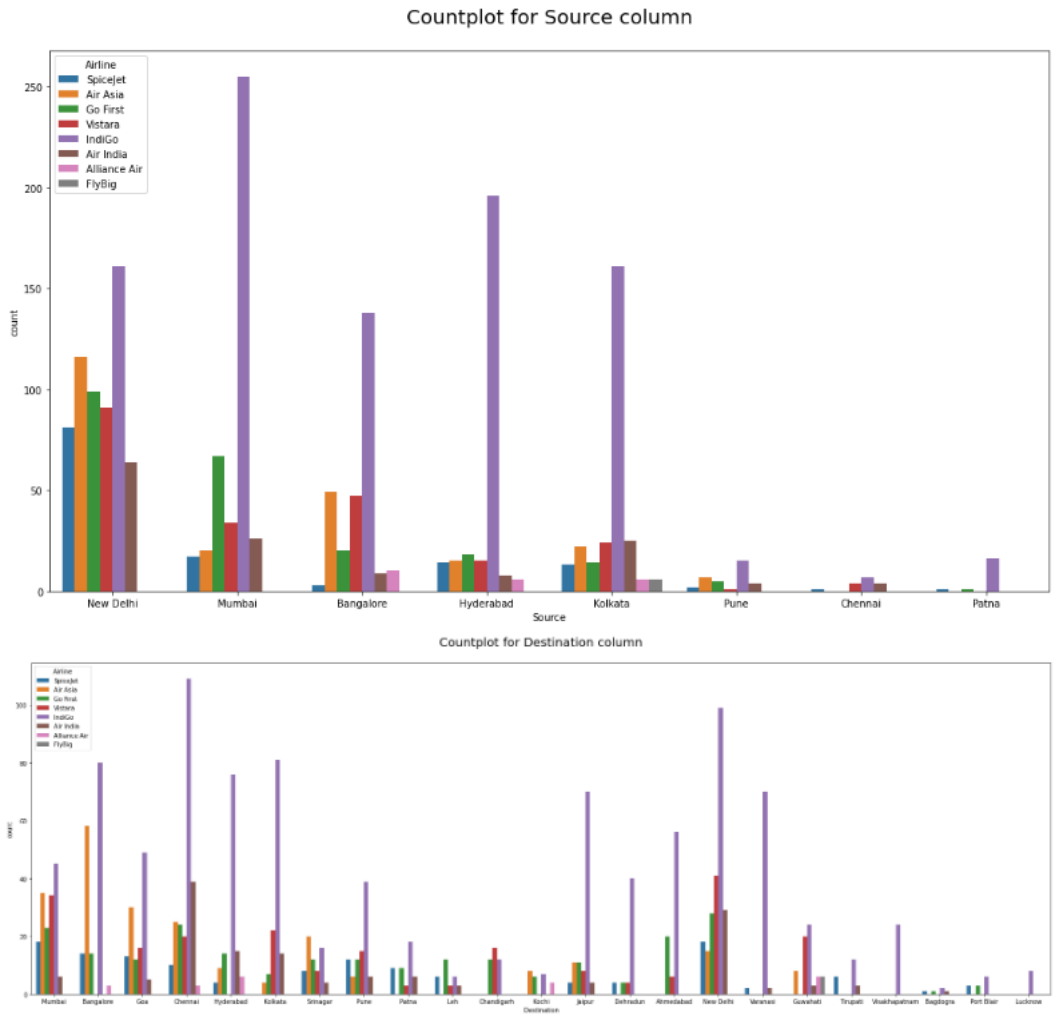
Barplot for Duration column vs Airline column



Barplot for Price column vs Airline column



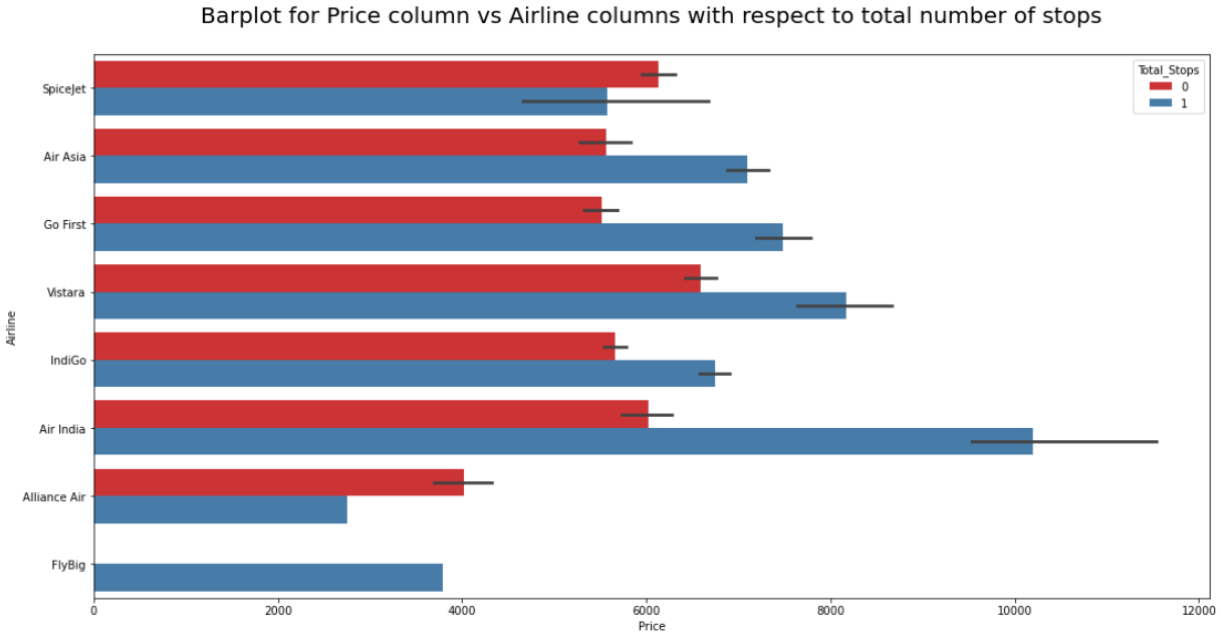
- When we observe the barplot for Departure hour vs Airline we can see that FlyBig has the highest departure time while IndiGo has the lowest departure time
- Considering the barplot for Arrival time vs Airline we can see that FlyBig has the highest arrival time while Vistara have the lowest arrival time
- Looking at the barplot for Flight duration vs Airline we observe that Ai Asia has the highest flight duration while Alliance Air has the lowest flight duration collectively
- Comparing the barplots for Flight prices vs Airline we can clearly see that Vistara have very high flight prices while the FlyBig has the lowest fare.



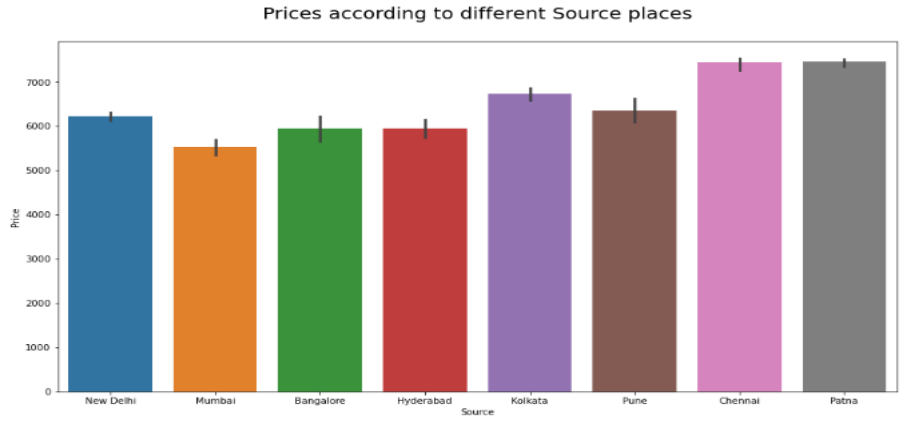
- Checking out the Source place details for each and every airline we

can see that Mumbai city has the highest number of departure flights for Indigo airlines

- Looking at the Destination place details for each airline we can see that Chennai city has the highest number of arrival flights for Indigo airlines
- Overall, I can notice that Indigo flights do quite well and can be used for arrival and departure to and from any location in India



- Spicejet has the maximum non stop flight
- Air India has the maximum no of 1 stop flights



- Airfares in Vistara and Air India are high when compared to other airlines.

- Flight prices when departing from cities like Chennai and Patna have higher price range but the others are around the similar range a bit lesser in pricing but not providing a huge difference as such
- Similarly, prices when arriving in cities Portblair and Dheradun have high price range
- When we consider the layovers for pricing situation then obviously direct flights are cheaper when compared to flights that have 1 or more stops.

- Interpretation of the Results

From the above EDA we can easily understand the relationship between features, and we can even see which things are affecting the price of flights. 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 have tried 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.

CONCLUSION

- Key Findings and Conclusions of the Study

In this project we have scraped the flight data from airline webpages. Then the comma separated value file is loaded into a data frame. Luckily, we don't have any missing values in our data set. Looking at the data set we understand that there are some features needs to be processed like converting the data types and get the actual value from the string entries from the time related columns. After the data has been processed, I have done some EDA to understand the relation among features and the target variable. Features like flight duration, number of stops during the journey and the availability of meals are playing major role in predicting the prices of

the flights. As we have seen, the prediction is showing a similar relationship with the actual price from the scrapped data set. This means the model predicted correctly and it could help airlines by predicting what prices they can maintain. It could also help customers to predict future flight prices and plan the journey accordingly, because it is difficult for airlines to maintain prices since it changes dynamically due to different conditions. Hence by using Machine Learning techniques we can solve this problem. The above research will help our client to study the latest flight price market and with the help of the model built he can easily predict the price ranges of the flight and will helps him to understand Based on what factors the fight price is decided.

- Learning Outcomes of the Study in respect of Data Science

While working on this project I learned many things about the features of flights and about the flight ticket selling web platforms and got the idea that how the machine learning models have helped to predict the price of flight tickets. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe price of tickets. Data cleaning was one of the important and crucial things in this project where I dealt with features having string values, features extraction and selection. Finally got XGB Regressor as best model.

The challenges I faced while working on this project was when I wab scrapping the real time data from yatra website, it took so much time to gather data. Finally, our aim was achieved by predicting the flight ticket price and built flight price evaluation model that could help the buyers to understand the future flight ticket prices.

- Limitations of this work and Scope for Future Work

Limitations: The main limitation of this study is the low number of records that have been used. In the dataset our data is not properly distributed in some of the columns many of the values in the columns are having string values which I had taken care. Due to some reasons our models may not make the right patterns and the performance of the model also reduces. So that issues need to be taken care.

Future work: The greatest shortcoming of this work is the shortage of data. Anyone wishing to expand upon it should seek alternative sources of historical data manually over a period. Additionally, a more varied set of flights should be explored, since it is entirely plausible that airlines vary their pricing strategy according to the characteristics of the flight (for example, fares for regional flights out of small airports may behave differently than the major, well flown routes we considered here). Finally, it would be interesting to compare our system's accuracy against that of the commercial systems available today (preferably over a period).