FLIGHT PRICE PREDICTION

Submitted by:

Dipesh Ramesh Limaje

Problem Statement

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 that 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 on flight fares with other features and work to make a model to predict the fares of flights.

Conceptual Background of the Domain Problem

The domain of flight price prediction involves predicting the prices of airline tickets for flights in the future. This problem is important because it can help travelers make informed decisions about when to purchase tickets and how to save money on their travel costs.

There are several factors that can affect the price of a flight, including demand for the route, availability of seats, competition from other airlines, and the time of year. In order to make accurate price predictions, it is necessary to take these factors into account.

One approach to predicting flight prices is to use historical data on ticket prices and other relevant variables (such as demand and competition) to build a statistical model. This model can then be used to make predictions about future prices based on current and anticipated conditions. Another approach is to use machine learning algorithms, which can learn patterns in the data and make more accurate predictions.

Regardless of the approach used, it is important to have a large and diverse dataset in order to accurately predict flight prices. This dataset should include information on ticket prices, as well as other relevant variables such as demand, competition, and time of year. It should also include data from a range of different routes and airlines in order to capture the full range of factors that can affect flight prices.

Review of Literature

There has been a significant amount of research on the topic of flight price prediction in recent years. This research has typically focused on the development and evaluation of statistical models and machine learning algorithms for predicting flight prices.

One common approach to flight price prediction is to use time series analysis. This involves analysing historical data on ticket prices and other relevant variables (such as demand and competition) in order to identify patterns and trends over time. These patterns and trends can then be used to make predictions about future prices.

Another approach is to use machine learning algorithms, such as decision trees, random forests, and neural networks. These algorithms can learn patterns in the data and make more accurate predictions than traditional statistical models. Some studies have found that machine learning algorithms can outperform time series models in predicting flight prices, particularly when the data is highly nonlinear or when there are many variables to consider.

Overall, the literature suggests that both time series analysis and machine learning can be effective approaches to flight price prediction. The choice of approach will depend on the specific characteristics of the data and the goals of the prediction.

Motivation for the Problem Undertaken

There are several reasons why predicting flight prices is an important problem to undertake.

First, the cost of air travel is a significant expense for many people, and the ability to accurately predict flight prices can help travellers save money on their travel costs. By knowing when to purchase tickets and which routes offer the best prices, travellers can make more informed decisions about their travel plans and potentially save hundreds of dollars.

Second, predicting flight prices can also be beneficial for airlines and other travel companies. By being able to anticipate changes in demand and competition, these companies can adjust their pricing and marketing strategies to maximize revenue. Accurate price predictions can also help them better understand the factors that influence demand and competition, allowing them to make more informed decisions about their operations.

Finally, predicting flight prices can also have broader economic benefits. By helping to make air travel more affordable and predictable, it can encourage more people to travel, which can stimulate economic activity and contribute to the growth of the tourism industry.

Mathematical/ Analytical Modelling of the Problem

There are a number of mathematical and analytical modelling approaches that can be used to tackle the problem of flight price prediction. Some common approaches include:

Time series analysis: This involves analysing historical data on ticket prices and other relevant variables (such as demand and competition) in order to identify patterns and trends over time. These patterns and trends can then be used to make predictions about future prices. Time series models can be simple, such as moving average models or exponential smoothing, or more complex, such as autoregressive integrated moving average (ARIMA) models or seasonal decomposition models.

Regression analysis: This involves building a statistical model to predict a continuous outcome (such as flight prices) based on one or more predictor variables (such as demand or competition). Common types of regression models include linear regression, polynomial regression, and multiple regression.

Machine learning algorithms: These algorithms can learn patterns in the data and make predictions based on those patterns. Some common machine learning algorithms used for flight price prediction include decision trees, random forests, and neural networks.

Optimization techniques: These techniques can be used to identify the optimal combination of variables (such as ticket prices and demand) that maximize a specific objective (such as revenue). Examples of optimization techniques include linear programming and integer programming.

Ultimately, the choice of modeling approach will depend on the specific characteristics of the data and the goals of the prediction. A combination of approaches may also be used in order to achieve the best results.

Data Sources and their formats

Web scraping can be a useful way to obtain data for flight price prediction, particularly if the data is not readily available from other sources. When web-scraping airline websites for flight price data, some considerations to keep in mind include:

Data sources: The data you are looking to obtain will determine which airline websites you need to scrape. For example, if you are interested in international flights, you may need to scrape multiple airline websites that operate in different countries.

Data formats: The data on airline websites is likely to be presented in a variety of formats, including tables, lists, and individual flight details pages. You will need to determine the best way to extract the data you are interested in from these formats.

Data quality: The accuracy and completeness of the data you obtain through web scraping will depend on the quality of the source website. You will need to carefully check the data for errors or missing values and take steps to address any issues you find.

Web scraping tools: There are a number of tools and libraries available for web scraping, including Python libraries such as Beautiful Soup and Selenium. These tools can make it easier to extract the data you need from airline websites, but you will still need to have some programming skills in order to use them effectively.

Overall, web scraping can be a useful way to obtain data for flight price prediction, but it requires careful planning and attention to detail in order to ensure that the data is of high quality.

Data Pre-processing Done

Pre-processing

```
In [88]: df.shape
```

```
Out[88]: (1643, 9)
```

```
In [89]: #checking the total columns present
         df.columns
```

```
Out[89]: Index(['Flight Name', 'Depart Location', 'Arrival Location', 'Depart Time',
                'Arrival Time', 'Total Stop', 'Total Time', 'Price', 'Date of Journey'],
               dtype='object')
```

```
In [90]: # checking the information and datatypes of each columns
         df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1643 entries, 0 to 1642
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Flight Name       1643 non-null   object
 1   Depart Location   1643 non-null   object
 2   Arrival Location  1643 non-null   object
 3   Depart Time       1643 non-null   object
 4   Arrival Time      1643 non-null   object
 5   Total Stop        1643 non-null   object
 6   Total Time        1643 non-null   object
 7   Price             1643 non-null   object
 8   Date of Journey   1643 non-null   object
dtypes: object(9)
memory usage: 115.6+ KB
```

```
In [91]: df.isnull().sum()
```

```
Out[91]: Flight Name         0
         Depart Location     0
         Arrival Location    0
         Depart Time         0
         Arrival Time        0
```

Pre-processing I have done

1] Check for the Null values

2] Check the Shape of the dataset

3] Check all the column’s Names and Compared them with the Data Description given to us.

4] Then I have check for the duplicates and remove the duplicate by drop duplicate method

In [92]:
```
#  check  the duplicate
duplicate = df[df.duplicated()]
print("Duplicate Rows :")

#  Print the resultant Dataframe
duplicate
```

Duplicate Rows :

Out[92]:

	Flight Name	Depart Location	Arrival Location	Depart Time	Arrival Time	Total Stop	Total Time	Price	Date of Journey
362	Vistara	Goa	New Delhi	00:40	16:45	2 Stop(s)	16h 05m	25,037	04-01-2023
365	Vistara	Goa	New Delhi	00:40	22:00	2 Stop(s)	21h 20m	25,491	04-01-2023
449	Vistara	Chennai	Mumbai	07:05	14:05	2 Stop(s)	31h 00m	16,624	04-01-2023
552	Air India	Chennai	Pune	11:10	21:10	2 Stop(s)	10h 00m	22,373	04-01-2023
663	Air India	Chennai	Pune	06:10	18:10	2 Stop(s)	12h 00m	14,044	04-02-2023
926	Air India	Goa	Mumbai	01:15	14:55	2 Stop(s)	13h 40m	16,744	04-02-2023
929	Air India	Goa	Mumbai	14:05	22:05	2 Stop(s)	32h 00m	18,942	04-02-2023
1282	Vistara	Goa	Mumbai	14:40	14:35	2 Stop(s)	23h 55m	10,524	19-01-2023
1284	Vistara	Goa	Mumbai	14:40	16:55	2 Stop(s)	26h 15m	10,525	19-01-2023

In [93]:
```
df.drop_duplicates(inplace=True)
```

In [94]:
```
#  check  the duplicate
duplicate = df[df.duplicated()]
print("Duplicate Rows :")

#  Print the resultant Dataframe
duplicate
```

Duplicate Rows :

5] Then I have checked the Datatypes and I found that the price column has the wrong datatype.

In [111]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1634 entries, 0 to 1642
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Flight Name       1634 non-null   object
 1   Depart Location   1634 non-null   object
 2   Arrival Location  1634 non-null   object
 3   Total Stop        1634 non-null   object
 4   Price             1634 non-null   object
 5   Day               1634 non-null   int64
 6   Month             1634 non-null   int64
 7   year              1634 non-null   int64
 8   Dep_hour          1634 non-null   int64
 9   Dep_min           1634 non-null   int64
 10  Arrival_hour      1634 non-null   int64
 11  Arrival_min       1634 non-null   int64
 12  Duration_hours    1634 non-null   int64
 13  Duration_mins     1634 non-null   int64
dtypes: int64(9), object(5)
memory usage: 191.5+ KB
```

6] After treating all duplicate values and dropping all unwanted columns finally adding some columns and treating all datatypes our dataset's final shape is 1634 Rows and 14 Columns.

```
In [138]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1634 entries, 0 to 1642
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Flight Name       1634 non-null   int32  
 1   Depart Location   1634 non-null   int32  
 2   Arrival Location  1634 non-null   int32  
 3   Total Stop        1634 non-null   int32  
 4   Price             1634 non-null   float64
 5   Day               1634 non-null   int64  
 6   Month             1634 non-null   int64  
 7   year              1634 non-null   int64  
 8   Dep_hour          1634 non-null   int64  
 9   Dep_min           1634 non-null   int64  
 10  Arrival_hour      1634 non-null   int64  
 11  Arrival_min       1634 non-null   int64  
 12  Duration_hours    1634 non-null   int64  
 13  Duration_mins     1634 non-null   int64  
dtypes: float64(1), int32(4), int64(9)
memory usage: 230.5 KB
```

7] After that I have to Describe the dataset to observe the numerical values and write the Observations.

```
In [139]: df.describe()
```

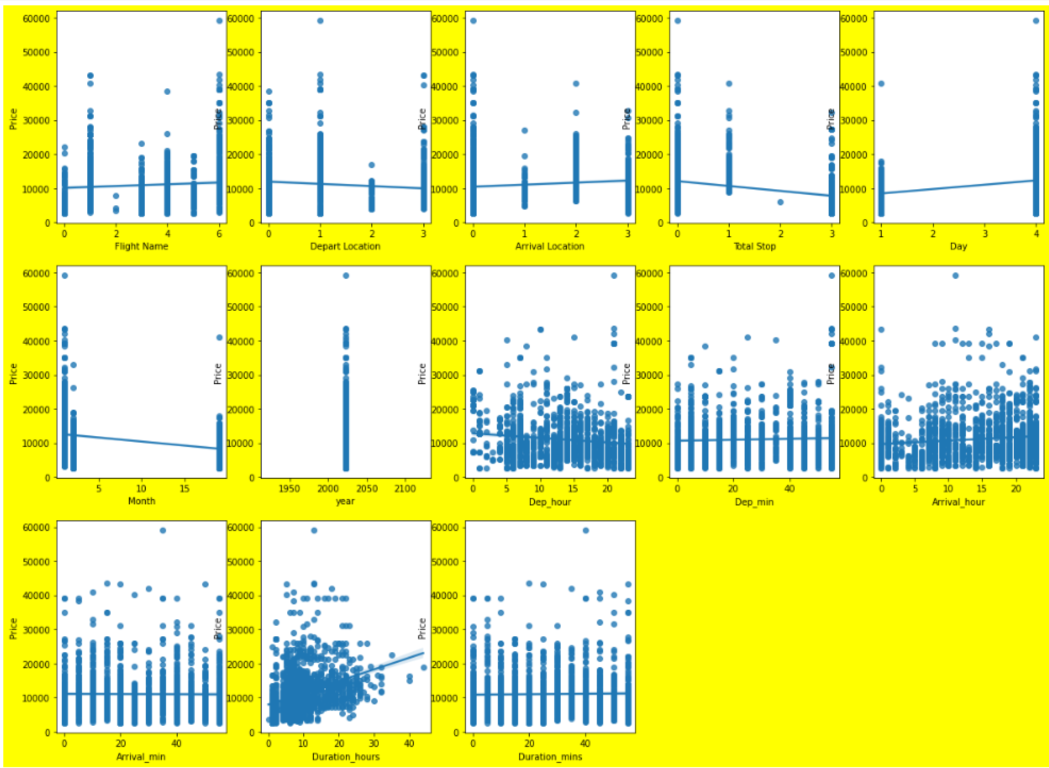
Out[139]:

ht Name	Depart Location	Arrival Location	Total Stop	Price	Day	Month	year	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_h
.000000	1634.000000	1634.000000	1634.000000	1634.000000	1634.000000	1634.000000	1634.0	1634.000000	1634.000000	1634.000000	1634.000000	1634.0C
.447368	1.403305	0.951040	0.755202	11056.040392	2.990208	7.376377	2023.0	13.439412	26.704406	13.436353	24.850061	8.9C
.024040	1.217655	1.174354	1.235840	6344.599763	1.418071	8.292108	0.0	6.046285	18.306739	7.166376	17.573017	6.72
.000000	0.000000	0.000000	0.000000	2600.000000	1.000000	1.000000	2023.0	0.000000	0.000000	0.000000	0.000000	0.0C
.000000	0.000000	0.000000	0.000000	6828.750000	1.000000	1.000000	2023.0	8.000000	10.000000	8.000000	10.000000	4.0C
.000000	1.000000	0.000000	0.000000	9627.500000	4.000000	2.000000	2023.0	14.000000	25.000000	14.000000	20.000000	7.0C
.000000	3.000000	2.000000	1.000000	13708.000000	4.000000	19.000000	2023.0	18.000000	45.000000	20.000000	40.000000	12.0C
.000000	3.000000	3.000000	3.000000	59133.000000	4.000000	19.000000	2023.0	23.000000	55.000000	23.000000	55.000000	44.0C

observation

```
1] The dataset contains 1643 rows and 9 columns
2] There are no null value present in the dataset
3] There were duplicates in the dataset we have remove the duplicates
4] We have change the datatype of the price column  from object to float
5] The year columns has one specific number so we will drop it
```

Data Inputs- Logic- Output Relationships



To observe the relationship between Feature and label so I created this Regression plot to observe which features are positively co-related and which features are negatively co-related.

State the set of assumptions (if any) related to the problem under consideration

1] For this particular problem I have dropped the duplicate value which I have found.

2] For this particular problem I have assumed that the Maximum VIF should be 5, if any of the features has a VIF which is greater than 5 we should drop that feature.

Hardware and Software Requirements and Tools Used

Hardware Requirements: -Computer with minimum 8 GB RAM -High-speed internet connection -High-end graphics card -External storage device

Software Requirements: -Python programming language -TensorFlow -Keras -Scikit-Learn -Pandas -Matplotlib -Seaborn

Tools Used: -Jupyter Notebook -Google Collab -Tableau -Power BI

Predicting the price of a flight ticket: -Linear regression -Random Forest -GBoost -ADA Boost

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

There are a number of problem-solving approaches that can be used for flight price prediction, including:

Time series analysis: This involves analyzing historical data on ticket prices and other relevant variables (such as demand and competition) in order to identify patterns and trends over time. These patterns and trends can then be used to make predictions about future prices. Time series models can be simple, such as moving average models or exponential smoothing, or more complex, such as autoregressive integrated moving average (ARIMA) models or seasonal decomposition models.

Regression analysis: This involves building a statistical model to predict a continuous outcome (such as flight prices) based on one or more predictor variables (such as demand or competition). Common types of regression models include linear regression, polynomial regression, and multiple regression.

Machine learning algorithms: These algorithms can learn patterns in the data and make predictions based on those patterns. Some common machine learning algorithms used for flight price prediction include decision trees, random forests, and neural networks.

Simulation: This involves creating a model of the system being studied (in this case, the airline industry) and using it to generate predictions about future outcomes. Simulation can be useful for understanding the complex interactions between different variables and for testing different scenarios.

Ultimately, the choice of problem-solving approach will depend on the specific characteristics of the data and the goals of the prediction. A combination of approaches may also be used in order to achieve the best results.

Testing of Identified Approaches (Algorithms)

- LR (Linear Regression Model)
- GBDT (Gradient Boosting Regressor Model)
- RF (Random Forest Regressor Model)
- ADA (AdaBoost Regressor Model)

Run and Evaluate selected models

1st Model I have Created is the Logistic Regression Model

LinearRegression Model

In [162]:
```
#lets import necessary library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

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

Finding the best random state

In [163]:
```
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
    regression=LinearRegression()
    regression.fit(X_train,y_train)

    pred=regression.predict(X_train)
    training=regression.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=regression.predict(X_test)
    testing=regression.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Training Score 40.769840932165394 RandomState 11
Testing Score 32.10157004892004 RandomState 11
```

```
In [164]: print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , MaxRS)
```

```
MAXINING TESTING SCORE 49.27789679835741 ON RANDOM STATE OF 152
```

Training the model

```
In [165]: #splliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=152)
```

```
In [166]: #Training the data on Linear Regression Model
regression=LinearRegression()
regression.fit(X_train,y_train)
```

```
Out[166]: ▾ LinearRegression
LinearRegression()
```

```
In [167]: #training score
regression.score(X_train,y_train)
```

```
Out[167]: 0.36570394252237004
```

```
In [168]: #testing score
regression.score(X_test,y_test)
```

```
Out[168]: 0.4927789679835741
```

Model Score

```
Training Score = 36.570394252237004 %
Testing Score = 49.27789679835741 %
```

LR (Linear Regression Model) Score are

Training Score = 36.570394252237004 %

Testing Score = 49.27789679835741 %

LASSO MODEL

```
In [173]: #import library
          from sklearn.linear_model import Ridge,Lasso,RidgeCV,LassoCV

In [174]: ##### LASSO MODEL######

          lasscv = LassoCV(alphas = None , max_iter = 100)

          lasscv.fit(X_train , y_train)

Out[174]: LassoCV
          LassoCV(max_iter=100)

In [175]: # best aplha parameter
          alpha = lasscv.alpha_
          alpha

Out[175]: 2.117376988615322

In [176]: # now we have best parametr noe train according to it
          lasso_reg = Lasso(alpha)
          lasso_reg.fit(X_train,y_train)

Out[176]: Lasso
          Lasso(alpha=2.117376988615322)

In [177]: # now check r2 score
          lasso_reg.score(X_test,y_test)

Out[177]: 0.4914524199805541
```

RIDGE MODEL

```
In [178]: ############ RIDGE MODEL#########

          ridgecv = RidgeCV(alphas = np.arange(0.001,0.1,0.01))
          ridgecv.fit(X_train , y_train)

Out[178]: RidgeCV
          RidgeCV(alphas=array([0.001, 0.011, 0.021, 0.031, 0.041, 0.051, 0.061, 0.071, 0.081,
                 0.091]))

In [179]: # best aplha parameter
          alpha = ridgecv.alpha_
          alpha

Out[179]: 0.001

In [180]: # now we have best parametr noe train according to it
          ridge_reg = Ridge(alpha)
          ridge_reg.fit (X_train,y_train)

Out[180]: Ridge
          Ridge(alpha=0.001)

In [181]: # now check r2 score
          ridge_reg.score(X_test,y_test)

Out[181]: 0.49279939310127785

In [ ]:
```

SCORES

LASSO SCORES = 49.14524199805541 %
RIDGE SCORES = 49.279939310127785 %

LASSO SCORES = 49.14524199805541 %

RIDGE SCORES = 49.279939310127785 %

2nd Model I have Created is Ada Boost Regressor Model

AdaBoostRegressor Model

```
In [185]: # IMPORT LIBRARY
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import AdaBoostRegressor
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline

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

Finding the best random state

```
In [186]: #Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
    ada=AdaBoostRegressor()
    ada.fit(X_train,y_train)

    pred=ada.predict(X_train)
    training=ada.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=ada.predict(X_test)
    testing=ada.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

Training the model

```
In [188]: #spliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=125)

In [189]: # adaboost inilize
from sklearn.ensemble import AdaBoostRegressor
ada=AdaBoostRegressor()
ada.fit(X_train,y_train)

Out[189]: ▾ AdaBoostRegressor
AdaBoostRegressor()

In [190]:  # model prediction on training dataset
y_pred = ada.predict(X_train)

In [191]: accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.6129775142389957

In [192]: # model prediction on testing datadet
pred = ada.predict(X_test)

In [193]: accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.6027296465562604
```

Model Scores

```
Training Score = 61.29775142389957 %
testing Score = 60.27296465562604 %
```

Ada Boost Regressor Model Scores

Training Score = 61.29775142389957 %
testing Score = 60.27296465562604 %

Hyperparameter Tuning for Ada Boost

In [194]:
```
### HYPERPARAMETER TUNING ###
from sklearn.model_selection import RandomizedSearchCV
```

In [195]:
```
params = {'n_estimators': [45,47,53,55,60,70] ,
          'learning_rate':[0.25,0.30,0.40]}
```

In [196]:
```
rnd_srch = RandomizedSearchCV(AdaBoostRegressor() , cv=5 , param_distributions=params , n_jobs=-1)
```

In [197]:
```
rnd_srch.fit(X_train,y_train)
```

Out[197]:
```
▸ RandomizedSearchCV
▸ estimator: AdaBoostRegressor
  ▸ AdaBoostRegressor
```

In [198]:
```
rnd_srch.best_params_
```

Out[198]:
```
{'n_estimators': 70, 'learning_rate': 0.4}
```

In [199]:
```
rnd_srch.best_estimator_
```

Out[199]:
```
▾ AdaBoostRegressor
AdaBoostRegressor(learning_rate=0.4, n_estimators=70)
```

In [206]:
```
ada = AdaBoostRegressor(learning_rate=0.41, n_estimators=62)
ada.fit(X_train,y_train)

pred=ada.predict(X_train)
print('====Training Score====')
print(metrics.r2_score(y_train,pred))
y_pred = ada.predict(X_test)

print ('=== Testing Score ===')
print (metrics.r2_score(y_test,y_pred))
```

```
====Training Score====
0.6141657549055579
=== Testing Score ===
0.6199375310688287
```

Model Score after Hyperparameter Tuning

Training Score = 61.41657549055579 %
Testing Score = 61.99375310688287 %

<u>3rd Model I have Created is Random Forest Regressor</u>

RandomForestRegressor Model

In [209]:
```
#import necessary library

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split,GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt
import seaborn as sns

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

Finding the best random state

In [210]:
```
#Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
    rf=RandomForestRegressor()
    rf.fit(X_train,y_train)

    pred=rf.predict(X_train)
    training=rf.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=rf.predict(X_test)
    testing=rf.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
MAXINING TESTING SCORE 80.1283444598521 ON RANDOM STATE OF 0
Training Score 95.81582990311529 RandomState 1
Testing Score 77.89392176169969 RandomState 1
```

Training the model

In [212]:
```
#splliting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=12)
```

In [213]:
```
rf=RandomForestRegressor()
rf.fit(X_train,y_train)
```

Out[213]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [214]:
```
 # model prediction on training dataset
y_pred = rf.predict(X_train)
```

In [215]:
```
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.9551342597494257
```

In [216]:
```
# model prediction on testing datadet
pred = rf.predict(X_test)
```

In [217]:
```
accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.8176377878169061
```

Model Score

```
Training Score = 95.51342597494257 %
Testing Score = 81.76377878169061 %
```

In [ ]:

<u>Random Forest Regressor Model Score</u>

Training Score = 95.51342597494257 %
Testing Score = 81.76377878169061 %

Hyperparameter tuning for Random Forest

In [219]:
```
# define parameters
parameters={'criterion':['mse','mae','poisson'],
           'max_features':['auto','sqrt','log2'],
           'min_samples_split':[1,11],
           'max_depth':[1,15],
           'min_samples_leaf':[1,7]}
```

In [220]:
```
rf=RandomForestRegressor()
clf=GridSearchCV(rf,parameters)
clf.fit(X_train,y_train)
```

Out[220]:
```
GridSearchCV
estimator: RandomForestRegressor
RandomForestRegressor
```

In [221]:
```
#print best parameters
print(clf.best_params_)
```

```
{'criterion': 'poisson', 'max_depth': 15, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 1}
```

In [222]:
```
#reassign best parameters
rf=RandomForestRegressor(criterion= 'poisson', max_depth= 15, max_features= 'auto', min_samples_leaf= 1, min_samples_split= 1)
rf.fit(X_train,y_train)
```

Out[222]:
```
RandomForestRegressor
RandomForestRegressor(criterion='poisson', max_depth=15, max_features='auto',
                      min_samples_split=1)
```

In [223]:
```
from sklearn.metrics import r2_score
print ('Training R2 Score: ' ,rf.score(X_train,y_train)*100)
```

```
Training R2 Score:  95.51452904666877
```

In [224]:
```
pred_decision=rf.predict(X_test)
rfs = r2_score(y_test,pred_decision)
```

In [225]:
```
print('Testing R2 Score:' , rfs*100)
```

```
Testing R2 Score: 82.53559591247217
```

Model Score after Hyperparameter Tuning

Training Score = 95.51452904666877 %
Testing Score = 82.53559591247217 %

4th Model I have Created is Gradient Boosting Regressor Model

GradientBoostingRegressor Model

```
In [228]: # import Library

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectPercentile , chi2
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingRegressor
```

Finding the best random state

```
In [229]: #Best Random State
MaxAccu=0
MaxRS=0

for i in range (0,200):
    X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=i)
    gbdt=GradientBoostingRegressor()
    gbdt.fit(X_train,y_train)

    pred=gbdt.predict(X_train)
    training=gbdt.score(X_train,y_train)
    print ('Training Score' , training*100 , 'RandomState' ,i)

    y_pred=gbdt.predict(X_test)
    testing=gbdt.score(X_test,y_test)
    print ('Testing Score' , testing*100 , 'RandomState' ,i)
    print('\n')


    if testing>MaxAccu:
        MaxAccu=testing
        MaxRS=i
        print('MAXINING TESTING SCORE' , MaxAccu*100 , 'ON RANDOM STATE OF' , i)
```

```
Training Score 80.43614490258028 RandomState 0
Testing Score 77.77345700312569 RandomState 0


MAXINING TESTING SCORE 77.77345700312569 ON RANDOM STATE OF 0
Training Score 81.68933977356201 RandomState 1
Testing Score 69.58611899914416 RandomState 1
```

Training the model

```
In [231]: #splitting our data into train test split and randomstate 8
X_train,X_test,y_train,y_test=train_test_split(X_scaled,y,test_size=0.25,random_state=12)

In [232]: # initiate GradientBoostingClassifier
gbdt= GradientBoostingRegressor()
gbdt.fit(X_train , y_train)

Out[232]: ▸ GradientBoostingRegressor
GradientBoostingRegressor()

In [233]:  # model prediction on training dataset
y_pred = gbdt.predict(X_train)

In [234]: from sklearn.metrics import r2_score
import sklearn.metrics as metrics
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.794920444374785

In [235]: # model prediction on testing datadet
pred = gbdt.predict(X_test)

In [236]: accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)

R Squared Score :  0.8014707890064717

In [ ]:
```

Model Score

```
Training Score = 79.4920444374785 %
Testing Score = 80.14707890064717 %
```

Gradient Boosting Regressor Model Model Score

Training Score = 79.4920444374785 %
Testing Score = 80.14707890064717 %

Hyperparameter tuning for GradientBoostingRegressor

In [238]:
```
# internally it will use decision tree as name suggest GBDT and here we are going to add one new parameter i.e learning rate

grid_params = {'max_depth' : range(1,8),
               'min_samples_split': range(2,12,1),
               'learning_rate': np.arange(0.1 , 0.9),
               'n_estimators': [90,95,100,105,110]}
```

In [239]:
```
grid = GridSearchCV(GradientBoostingRegressor() , param_grid = grid_params , n_jobs = -1)
```

In [240]:
```
grid.fit(X_train,y_train)
```

Out[240]:
```
GridSearchCV
estimator: GradientBoostingRegressor
GradientBoostingRegressor
```

In [241]:
```
grid.best_params_
```

Out[241]:
```
{'learning_rate': 0.1,
 'max_depth': 6,
 'min_samples_split': 7,
 'n_estimators': 100}
```

In [272]:
```
gbdt_clf = GradientBoostingRegressor(learning_rate= 0.13,
 max_depth= 8,
 min_samples_split= 8,
 n_estimators= 102)
```

In [273]:
```
gbdt_clf.fit(X_train,y_train)
```

Out[273]:
```
GradientBoostingRegressor
GradientBoostingRegressor(learning_rate=0.13, max_depth=8, min_samples_split=8,
                          n_estimators=102)
```

In [275]:
```
accuracy = metrics.r2_score (y_train , y_pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.998991722928025
```

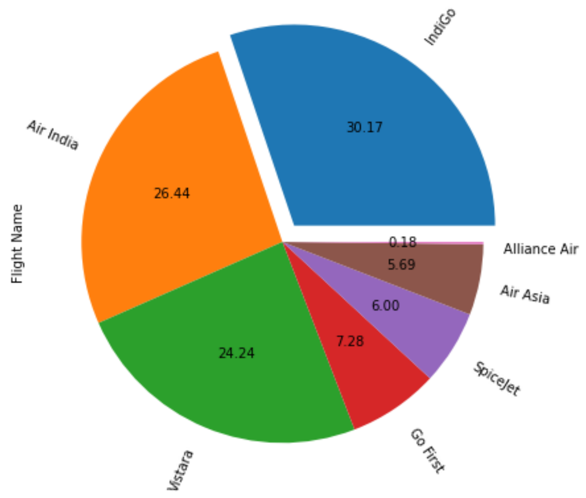
In [277]:
```
accuracy = metrics.r2_score(y_test,pred)
print ('R Squared Score : ' , accuracy)
```

```
R Squared Score :  0.8252709404118146
```

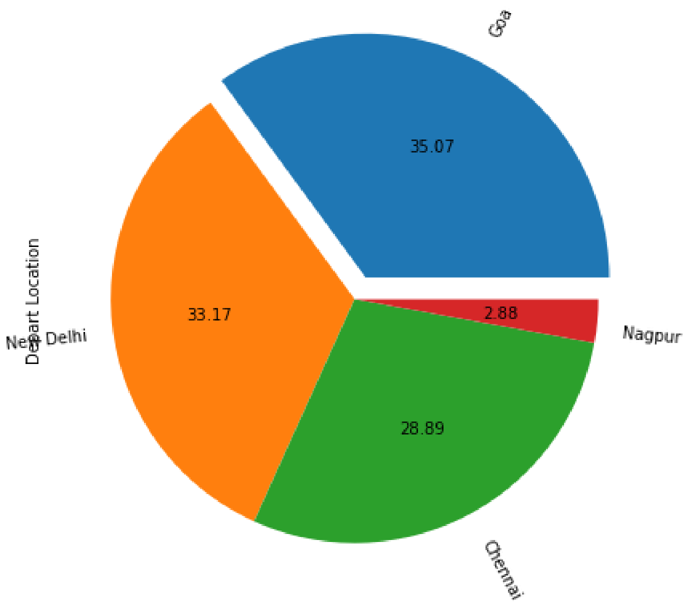
Model Score after Hyperparameter Tuning

Training Score = 99.8991722928025 %
Testing Score = 82.52709404118146 %

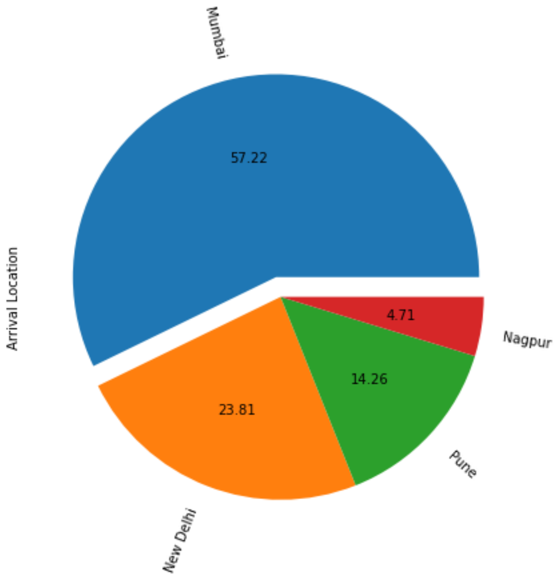
<u>Visualizations and EDA</u>



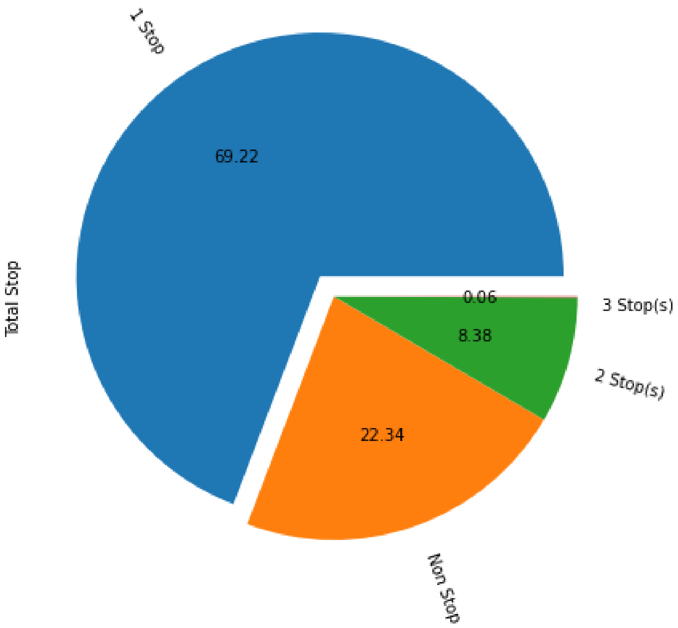
From the above pie chart, we observe that we have 7 different airlines and the indigo count is almost 30.17% followed by air India at 26.44% and the least is Alliance Air which is 0.18%.



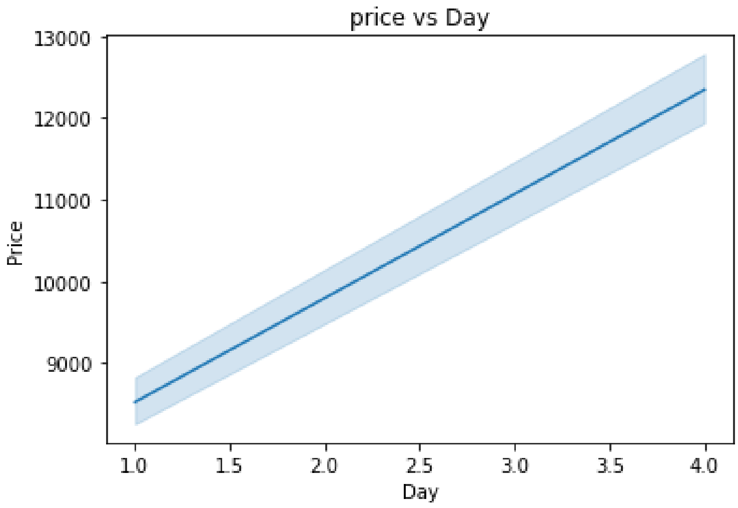
From the above pie chart, we have 4 different Departure locations and the count of people departing from goa is the most which are 35.07% followed by New Delhi is 33.17% followed by Chennai is 28.89% and the least is Nagpur which is 2.88%.



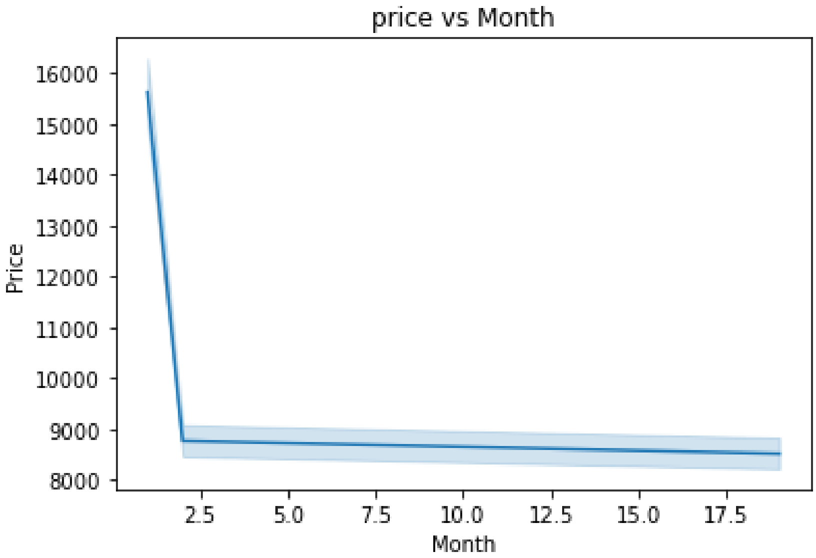
From the above pie chart, we have 4 different Arrival locations and the count of people who arrived from Mumbai is the most which are 57.22% followed by NewDelhi which is 23.81% followed by Pune is 14.26% and the least is Nagpur which is 4.71%.



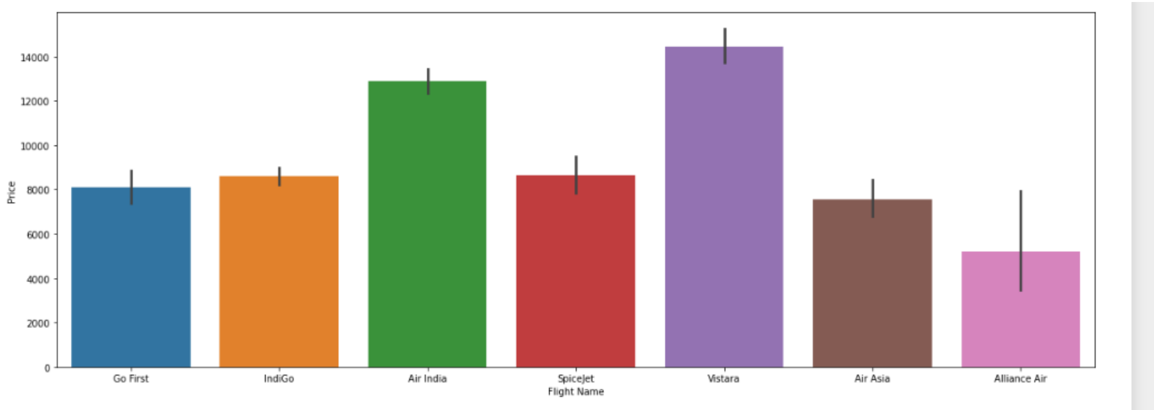
From the above pie chart, we observe the count of the flight with one stop is more which is 69.22% followed by non-stop flights with 22.34% followed by 2 stops and 3 stops which is 8.36% and 0.06%.



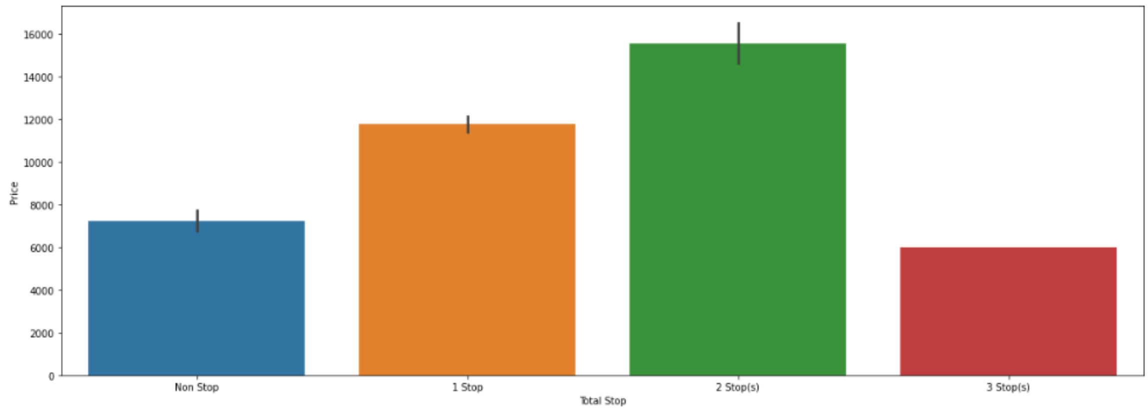
From the above line chart we observe as days go by the price of the flight ticket goes on increasing



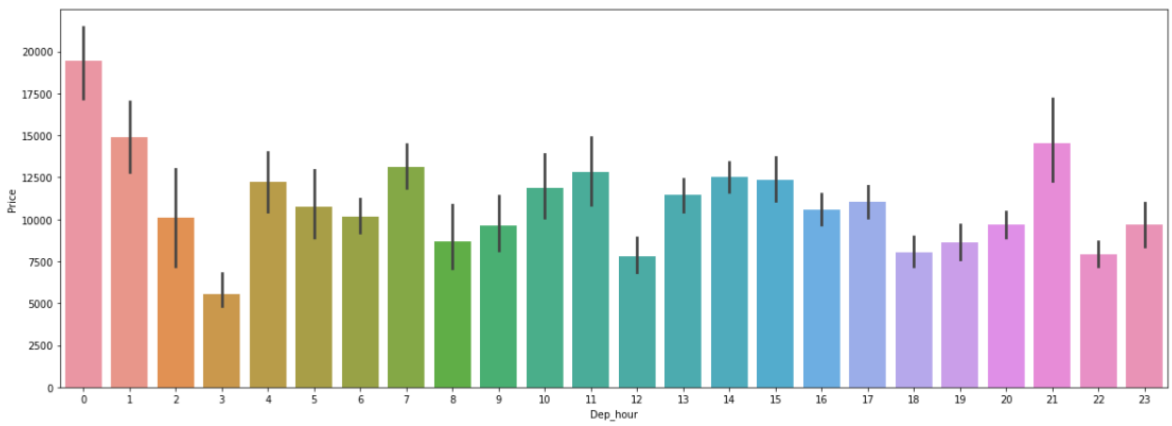
From the above line chart we observe as months go the price of flight tickets decreases (Advance Booking).



From the above bar chart, we observe the flight ticket price is high for Vistara and Air India airlines.



From the above bar chart, we observe the flight which has 2 stops has had a higher price.



From the above graph, we observe the time vs price of the night flight has more price as compared to day-time flights

Data Analysis insights

Q1] Do airfares change frequently?

ANS Yes, flight prices can change frequently for a variety of reasons. Some of the factors that can affect flight prices include:
1] Demand: When there is high demand for a particular flight or route, airlines may increase their prices to take advantage of the demand. Conversely, when demand is low, airlines may lower their prices in order to attract more passengers.
2] Competition: If there are multiple airlines offering flights on a particular route, they may compete with each other by adjusting their prices in order to attract passengers.
3] Economic conditions: Economic conditions such as recession or inflation can also affect flight prices. For example, during a recession, airlines may need to lower their prices in order to remain competitive.
Overall, flight prices can be affected by a range of factors, and as a result, they can change frequently. This can make it challenging for travelers to predict how much they will need to pay for their tickets and can make budgeting for travel difficult.

Q2] Do they move in small increments or in large jumps?

ANS When comparing flight prices over short periods of time, such as a few days, the changes in price may seem more significant. This is because there may be more factors at play that can cause prices to fluctuate over a short period of time. For example, a sudden increase in demand for a particular flight or route could lead to a significant jump in price within a few days.
However, when comparing prices over a longer period of time, such as a month, the changes in price may appear smaller. This is because the impact of any one factor is likely to be less pronounced over a longer period of time. For example, if demand for a particular flight or route increases over the course of a month, the price may gradually increase rather than jumping significantly all at once.
Overall, it is important to consider the time frame over which prices are being compared when evaluating changes in flight prices. A large jump over a short period of time may not be as significant when viewed in the context of a longer period of time.

Q3] Do they tend to go up or down over time?

ANS the price of flights tends to decrease, or trend downward, over a certain period of time. It does not necessarily mean that the price of every individual flight will decrease during this time, but rather that the average price of flights will be lower compared to the previous period.
It is important to note that this trend may not hold true for all flights or for all periods of time. The price of flights can be influenced by a variety of factors such as demand, competition, fuel costs, and other economic factors, and these factors can change over time. As a result, the trend of flight prices can vary and may not always be downward.

Q4] What is the best time to buy so that the consumer can save the most by taking the least risk?

ANS Generally speaking, it is often a good idea to book your flight tickets at least one month in advance. This is because by booking in advance, you are more likely to find the flight that you want with the seat availability that you need, and the fare is also typically lower than it would be closer to the travel date. Additionally, by booking your tickets well in advance, you have more time to make any necessary changes or cancellations if unexpected issues arise. However, it is important to note that this may not always be the case, as flight prices can be influenced by a variety of factors such as demand, competition, fuel costs, and other economic factors, and these factors can change over time."

Q5] Does price increase as we get near to departure date?

ANS yes, price increase as we get near to the departure date" would be: "As the departure date for a flight gets closer, the price of the flight tends to increase. This is because airlines often release their flights and offer their lowest fares well in advance, and as the travel date gets closer, the demand for seats tends to increase, leading to higher prices. Additionally, as the departure date approaches, there are fewer seats available on the flight, and the remaining seats may be in higher demand, resulting in higher prices.

Q6] Is Indigo cheaper than Jet Airways?

ANS In the dataset that I am using for my analysis, there are no flights from the airline called Air Jetways. The dataset only includes domestic flights. Based on the data that I have, I can say that the airline called Indigo tends to have lower prices compared to Air India and Vistara. However, when comparing Indigo to Spicejet, the prices seem to be approximately equal, according to the graph that I have created. It is important to note that these conclusions are based on the data that I have in my dataset and may not necessarily hold true for all flights.

Q7] Are morning flights expensive?

ANS According to my analysis, if we compare the price of morning flights, which are defined as flights that depart between 4 am and 10 am, to the price of flights that depart during other times of the day such as the evening, the prices seem to be similar and there is not a significant difference. However, when we compare the price of night-time flights, which are defined as flights that depart between 10 pm and 2 am, to the price of morning and evening flights, the night-time flights tend to be more expensive. This suggests that the time of day that a flight departs can affect its price, with night-time flights being more expensive on average compared to morning and evening flights.

CONCLUSION

Key Findings and Conclusions of the Study

So from above all 4 model scores, we observe Random Forest Regressor Model is best Suited model for this particular model as the training score is 95.51452904666877 % and the testing score is 82.53559591247217 % thus saving this model.

Learning Outcomes of the Study in respect of Data Science

1) First we look for null values and there was not any null present

2)Then I identified duplicates and I have dropped duplicates

3) Then we Performed EDA and wrote all observations for each graph

4)Then I dropped unnecessary columns

5)Then applied a label encoder to the categorical columns

6)Then also plotted the Distribution plot and regression plot

7)Then plotted boxplot to remove outliers

8)Then treated outliers with the Z-score method

9)Then scaled data and Also check for VIF

10)Then find the co-relation between feature and label by the CORR method

11)Then we selected all the features except Day and Month because VIF was greater than 5

12)Then I created 4 models that are Gradient Boosting Regressor, Random Forest Regressor, linear Regressor model, and Ada Boosting regressor model with hyperparameter tuning for all 4 models.

13)At last I selected the best model according to their Hyperparameter score and Training(R2) and testing score(R2)

Limitations of this work and Scope for Future Work

Limitations of this work:

- The model is only able to predict flight prices for a limited set of airports and airlines. It may not be able to accurately predict prices for flights from other airports or on other airlines.
- The model is only able to predict prices for a limited time period in the future. As time goes on, the accuracy of the predictions may decrease.
- The model may not be able to accurately predict prices in the event of unexpected events such as natural disasters, strikes, or changes in the political climate.
- The model is only able to predict prices based on the data that it has been trained on. If the patterns in the data change, the model may not be able to accurately predict future prices.

Scope for Future Work:

- Expanding the model to include data from a wider range of airports and airlines would allow it to make more accurate predictions for a larger number of flights.
- Incorporating data on unexpected events such as natural disasters, strikes, and changes in the political climate would allow the model to better handle such situations and make more accurate predictions.
- Training the model on a larger and more diverse dataset would allow it to better capture changes in patterns and make more accurate predictions.
- Developing methods to continuously update the model with new data as it becomes available would allow it to remain accurate over longer periods of time.
- Investigating the use of additional features such as the time of year, holidays, and local events could potentially improve the accuracy of the model.