Flight Price Prediction Project Report

Submitted by:

G. Premsagar

ACKNOWLEDGMENT

I'd like to express my heartfelt gratitude to my SME (Subject Matter Expert) Khushboo Garg, as well as Flip Robo Technologies, for allowing me to work on this project on flight price prediction and for assisting me in conducting extensive research, which allowed me to learn a lot of new things, particularly in terms of data collection.

In addition, I used a few other resources to assist me finish the job. I made sure to learn from the samples and adjust things to fit my project's needs. The following are all of the external resources that were utilised to create this project:

1) https://www.google.com/

2) https://www.youtube.com/

3) https://scikit-learn.org/stable/user_guide.html

4) https://github.com/

5) https://www.kaggle.com/

6) https://medium.com/

7) https://towardsdatascience.com/

8) https://www.analyticsvidhya.com/

INTRODUCTION

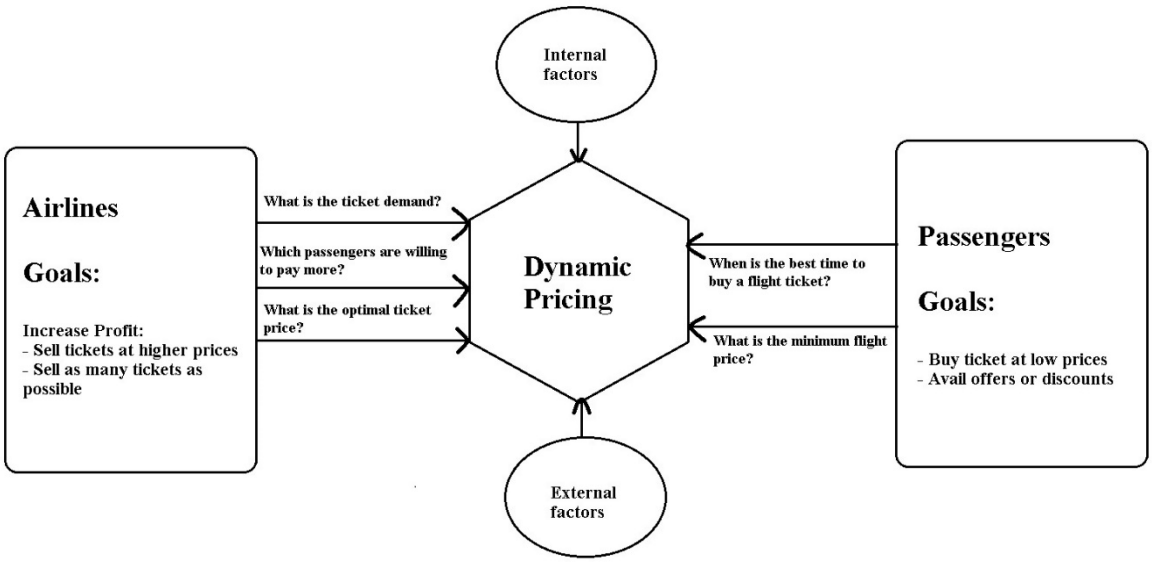
- Business Problem Framing

The airline sector is often regarded as one of the most advanced in terms of pricing methods. Nowadays, ticket rates for the same aircraft, even for seats close together, can vary dramatically. A flight's ticket price might fluctuate up to seven times each day. Customers are looking for the best deal on a ticket, while airlines are aiming to optimise their profit by keeping their overall income as high as possible. Mismatches between available seats and passenger demand, on the other hand, frequently result in either the consumer paying more or the airline losing money. Airlines are often outfitted with sophisticated tools and capabilities that allow them to exert control over the pricing process.

Mismatches between available seats and passenger demand, on the other hand, frequently result in either the consumer paying more or the airline losing money. Airlines are often outfitted with sophisticated tools and capabilities that allow them to exert control over the pricing process. Customers are becoming more strategic as a result of the introduction of numerous web tools that allow them to compare rates across different airline firms. Furthermore, the process of identifying ideal price is difficult for everyone due to airline competition.

Anyone who has purchased a plane ticket understands how pricing may fluctuate dramatically. Over time, the cheapest possible ticket on a specific flight becomes more and more costly. This frequently occurs as a result of a desire to increase revenue.

1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)



- Conceptual Background of the Domain Problem

  Airlines utilise complicated algorithms to determine flight fares based on the many factors that exist at the moment. To anticipate flight fares, these strategies include financial, marketing, and numerous societal elements. The number of individuals who fly has dramatically grown in recent years. Pricing alter dynamically owing to many variables, making it difficult for airlines to maintain prices. As a result, we will attempt to tackle this problem using machine learning. This can assist airlines in determining what rates they can keep. Customers may also use it to forecast future airline rates and plan their trip appropriately.

- Review of Literature

  As per the client's request, I scraped data from web sites and conducted analysis based on that data, such as determining which features of my data are altering prices. He looked at the price of a flight in relation to all of the features, such as which flight he should take.

- Motivation for the Problem Undertaken

  I worked on this based on the client's specifications and followed all of the stages till the model was deployed.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  In our scrapped dataset, our target variable "Flight_Prices " is a continuous variable. Therefore, we will be handling this modelling problem as regression.

  This project is done in three parts:

- Data Collection
- Data Analysis
- Model Building

  1. Data Collection

  You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data better the model. In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

  2. Data Analysis

  After cleaning the data, you have to do some analysis on the data. Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the consumer can save the most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

  3. Model Building

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

- Data Sources and their formats

The dataset is in the form of CSV (Comma Separated Value) format and consists of 8 columns (7 features and 1 label) with 1829 number of records as explained below:

- Airline_Names : This shows the list of all the Airline Names for which the data got scraped
- Departure_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
- Flight_Duration : We can see the total duration of a flight that it took to fly from the source to the destination
- Source_Place : Gives us the name of the source place where the flight journey began
- Destination_Place : Shows us the name of the destination place where the flight journey ended

- Number_Of_Stops : Lists the number of stops the flight is going to take to complete the entire journey
- Flight_Prices : Finally we have our label column that has the ticket prices for the aircraft journey.

```
#open dataset
df.head()
```

	Airline_Names	Departure_time	Arrival_time	Flight_Duration	Source_Place	Destination_Place	Number_Of_Stops	Flight_Prices
0	Go First	02:00	04:15	02 h 15 m	New Delhi	Mumbai	Non stop	5,954
1	Go First	08:00	10:10	02 h 10 m	New Delhi	Mumbai	Non stop	5,954
2	Go First	10:30	12:50	02 h 20 m	New Delhi	Mumbai	Non stop	5,954
3	Go First	12:45	15:00	02 h 15 m	New Delhi	Mumbai	Non stop	5,954
4	Go First	14:20	16:35	02 h 15 m	New Delhi	Mumbai	Non stop	5,954

- Data Preprocessing Done

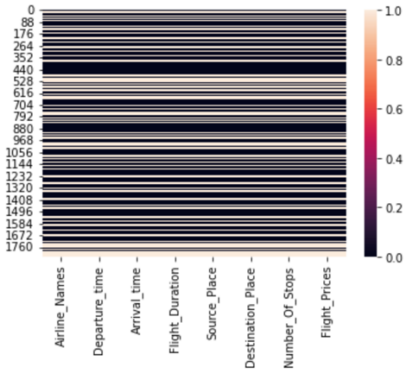
For the data pre-processing step, I checked through the dataframe for missing values and renamed values that needed a better meaningful name.

```
#checking for null values
df.isna().sum()
```

```
Airline_Names         742
Departure_time        742
Arrival_time          742
Flight_Duration       742
Source_Place          742
Destination_Place     742
Number_Of_Stops       742
Flight_Prices         742
dtype: int64
```

```
sns.heatmap(df.isna())
```

```
<AxesSubplot:>
```

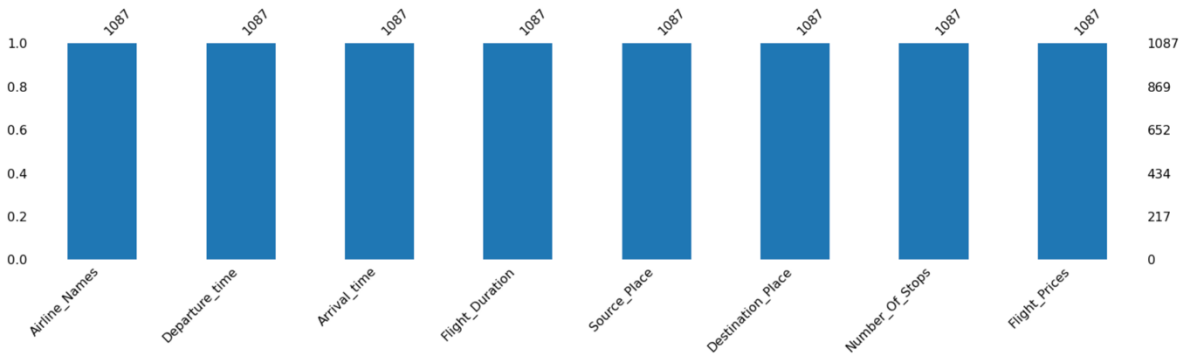


Removed null values.

```
#droping null values
df= df.dropna()
```

```
missingno.bar(df, figsize = (25,5), color="tab:blue")
```

```
<AxesSubplot:>
```



Checked the datatype details for each column to understand the numeric ones and its further conversion process.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1829 entries, 0 to 1828
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Airline_Names      1087 non-null   object
 1   Departure_time     1087 non-null   object
 2   Arrival_time       1087 non-null   object
 3   Flight_Duration    1087 non-null   object
 4   Source_Place       1087 non-null   object
 5   Destination_Place  1087 non-null   object
 6   Number_Of_Stops    1087 non-null   object
 7   Flight_Prices      1087 non-null   object
dtypes: object(8)
memory usage: 128.6+ KB
```

The various data processing performed on our data set are shown below with the code.

```
# Departure_Time

df["Dep_Hour"] = pd.to_datetime(df.Departure_time, format="%H:%M").dt.hour
df["Dep_Min"] = pd.to_datetime(df.Departure_time, format="%H:%M").dt.minute
df["Departure_time"] = df['Dep_Hour'] + df['Dep_Min'] / 60
#df.drop(columns = ['Dep_Hour','Dep_Min'], inplace=True)
df.head()
```

```
# Arrival_Time

df["Arr_Hour"] = pd.to_datetime(df.Arrival_time, format="%H:%M").dt.hour
df["Arr_Min"] = pd.to_datetime(df.Arrival_time, format="%H:%M").dt.minute
df["Arrival_time"] = df['Arr_Hour'] + df['Arr_Min'] / 60
#df.drop(columns = ['Arr_Hour','Arr_Min'], inplace=True)
df.head()
```

```
# Assigning and converting Duration column into hours column seperate

Duration = list(df["Flight_Duration"])


Dur_hour = []

for i in range(len(Duration)):

    if len(Duration[i].split()) != 2:    # Check if duration contains only hour or mins

        if "h" in Duration[i]:

            Duration[i] = Duration[i].replace(' h ',' ')   # Adds 0 minute
            Duration[i] = Duration[i].replace(' m','')   # Adds 0 minute
            Duration[i] = Duration[i].split(' ',1)[0]
            Dur_hour.append(float(Duration[i]))

        else:
            Duration[i] = "02" + Duration[i]           # Adds 0 hour
            Dur_hour.append(float(Duration[i]))
```

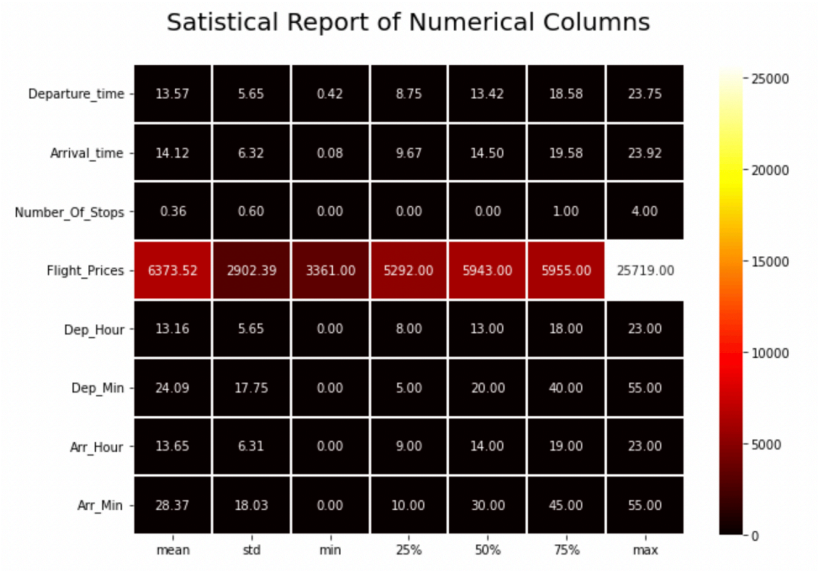
```
# Flight_Prices

df['Flight_Prices'] = df['Flight_Prices'].str.replace(',','')
df['Flight_Prices'] = df['Flight_Prices'].astype('float')
```

I then used the “describe” method to check the count, mean, standard deviation, minimum, maximum, 25%, 50% and 75% quartile data.

```
df.describe(include="all").T
```

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Airline_Names	1087	8	IndiGo	405	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Departure_time	1087.0	NaN	NaN	NaN	13.565241	5.648488	0.416667	8.75	13.416667	18.583333	23.75
Arrival_time	1087.0	NaN	NaN	NaN	14.121435	6.32478	0.083333	9.666667	14.5	19.583333	23.916667
Source_Place	1087	8	New Delhi	270	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Destination_Place	1087	7	New Delhi	276	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Number_Of_Stops	1087.0	NaN	NaN	NaN	0.355106	0.604563	0.0	0.0	0.0	1.0	4.0
Flight_Prices	1087.0	NaN	NaN	NaN	6373.524379	2902.386154	3361.0	5292.0	5943.0	5955.0	25719.0
Dep_Hour	1087.0	NaN	NaN	NaN	13.163753	5.645028	0.0	8.0	13.0	18.0	23.0
Dep_Min	1087.0	NaN	NaN	NaN	24.089236	17.748456	0.0	5.0	20.0	40.0	55.0
Arr_Hour	1087.0	NaN	NaN	NaN	13.648574	6.305144	0.0	9.0	14.0	19.0	23.0
Arr_Min	1087.0	NaN	NaN	NaN	28.371665	18.029489	0.0	10.0	30.0	45.0	55.0

Took a visual on just the numeric part as well and saw just the maximum value for Flight_Prices column at a higher scale.



- Data Inputs- Logic- Output Relationships

  Because the input data was all object datatype, we had to clean it up by deleting unnecessary information like "h" and "m" from the Flight Duration column and ensuring that the numeric data was transformed correctly. I then converted all of the category feature columns to numeric representation using the Ordinal Encoding technique.

  Code:

```
# Ordinal Encoder

oe = OrdinalEncoder()
def ordinal_encode(df, column):
    df[column] = oe.fit_transform(df[column])
    return df

column=["Airline_Names", "Source_Place", "Destination_Place"]
df=ordinal_encode(df, column)
df
```

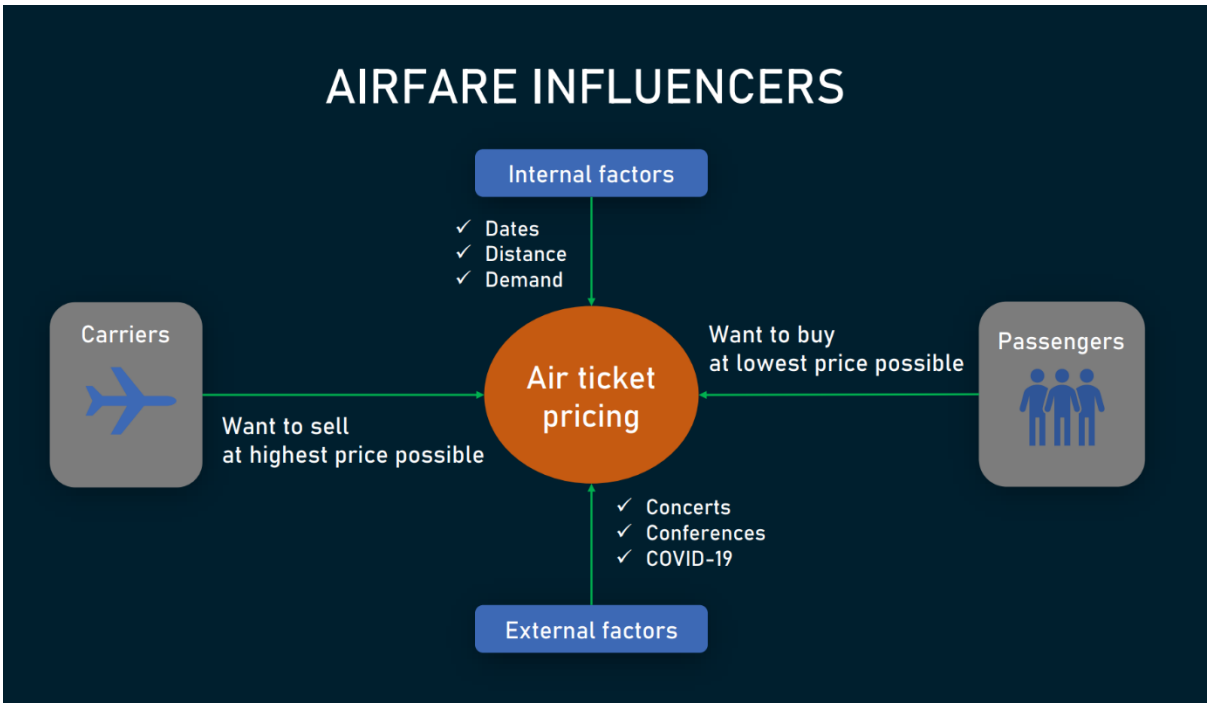
  Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

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

  The airline industry's pricing is frequently characterised to a mental game between carriers and passengers, in which both parties

compete for the best rates. Carrier executives like selling tickets at the highest feasible price while avoiding losing customers to competition. Passengers are obsessed with finding the cheapest airfares while also ensuring that they do not miss their trip. As a result of all of this, flight costs are volatile and difficult to forecast. But for those with intelligence and algorithms, nothing is impossible.

In the travel business, there are two basic use cases for flight price prediction. This function is included by OTAs and other travel sites to attract more people seeking for the greatest deals. Airlines use technology to predict competition rates and modify their pricing tactics as a result.



An OTA's passenger-side predictor advises the optimal time to purchase a ticket so that travellers can make educated selections. Carriers, on the other hand, are attempting to determine the best pricing to establish in order to maximise income while remaining competitive.

The endeavour is difficult in both circumstances since a variety of internal and environmental variables impact airfares.

Internal factors include

- purchase and departure dates,
- seasonality,
- holidays,
- the number of available airlines and flights,
- fare class,
- the current market demand, and
- flight distance.

External factors embrace events going on in the arrival or departure cities — like

- concerts,
- sports competitions,
- festivals,
- terrorist attacks,
- natural disasters,
- political gatherings,
- epidemic outbursts, and
- economic activities.

- Hardware and Software Requirements and Tools Used

- **Hardware technology being used:**
  CPU : MacBook Pro, M1 Chipset
  RAM : 8 GB
  GPU : 8GB
- **Software technology being used:**
  Programming language : Python
  Distribution : Anaconda Navigator
  Browser based language shell : Jupyter Notebook
- **Libraries/Packages specifically being used:**
  Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling, missingno

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  1. Clean the dataset from unwanted scraped details.

  2. Rename values with meaningful information.

  3. Encoding the categorical data to get numerical input data.

  4. Compare different models and identify the suitable model.

  5. R2 score is used as the primary evaluation metric.

  6. MSE and RMSE are used as secondary metrics.

  7. Cross Validation Score was used to ensure there are no overfitting our underfitting models.

- Testing of Identified Approaches (Algorithms)

  Libraries and Machine Learning Regression models that were used in this project are shown below.

```
# Importing required libraries

import pandas as pd
import numpy as np

import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

import missingno
import pandas_profiling
from sklearn import metrics
from scipy.stats import zscore
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor

from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV

import joblib

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

All the regression machine learning algorithms used are:

- Linear Regression Model
- Ridge Regularization Model
- Lasso Regularization Model
- Support Vector Regression Model
- Decision Tree Regression Model
- Random Forest Regression Model
- K Neighbours Regression Model
- Gradient Boosting Regression Model
- Ada Boost Regression Model
- Extra Trees Regression Model

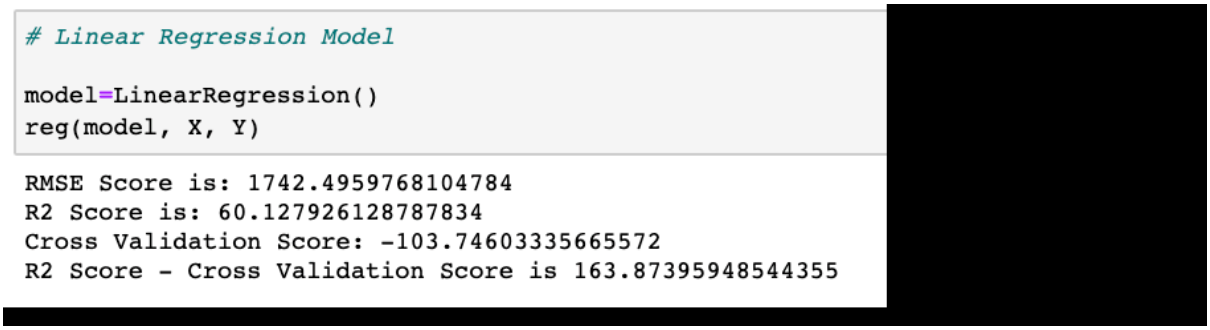
- Run and Evaluate selected models

I created a Regression Machine Leaning Model function incorporating the evaluation metrics so that we can get the required data for all the above models.

Code:

```
# Regression Model Function

def reg(model, X, Y):
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25, random_state=638)

    # Training the model
    model.fit(X_train, Y_train)

    # Predicting Y_test
    pred = model.predict(X_test)

    # RMSE - a lower RMSE score is better than a higher one
    rmse = mean_squared_error(Y_test, pred, squared=False)
    print("RMSE Score is:", rmse)

    # R2 score
    r2 = r2_score(Y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, X, Y, cv=5).mean())*100
    print("Cross Validation Score:", cv_score)

    # Result of r2 score minus cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

Output:



```
# Linear Regression Model

model=LinearRegression()
reg(model, X, Y)
```

```
RMSE Score is: 1742.4959768104784
R2 Score is: 60.127926128787834
Cross Validation Score: -103.74603335665572
R2 Score - Cross Validation Score is 163.87395948544355
```

- Key Metrics for success in solving problem under consideration

RMSE Score:
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

R2 Score:
The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

Cross Validation Score:
Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are

common tactics that you can use to select the value of k for your dataset (I have used 5-fold validation in this project). There are commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.

Hyper Parameter Tuning:
In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

```
# Choosing Extra Trees Regressor

fmod_param = {'n_estimators' : [100, 200, 300],
              'criterion' : ['squared_error', 'mse', 'absolute_error', 'mae'],
              'n_jobs' : [-2, -1, 1],
              'random_state' : [42, 251, 340]
             }

GSCV = GridSearchCV(ExtraTreesRegressor(), fmod_param, cv=5)
GSCV.fit(X_train,Y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'n_estimators': [100, 200, 300], 'n_jobs': [-2, -1, 1],
                         'random_state': [42, 251, 340]})
```

Final model score after plugging in the best parameter values:

```
Final_Model = ExtraTreesRegressor(criterion='mae', n_estimators=300, n_jobs=-2, random_state=251)
Model_Training = Final_Model.fit(X_train, Y_train)
fmod_pred = Final_Model.predict(X_test)
fmod_r2 = r2_score(Y_test, fmod_pred, multioutput='variance_weighted')*100
print("R2 score for the Best Model is:", fmod_r2)
```

```
R2 score for the Best Model is: 93.34894805151457
```

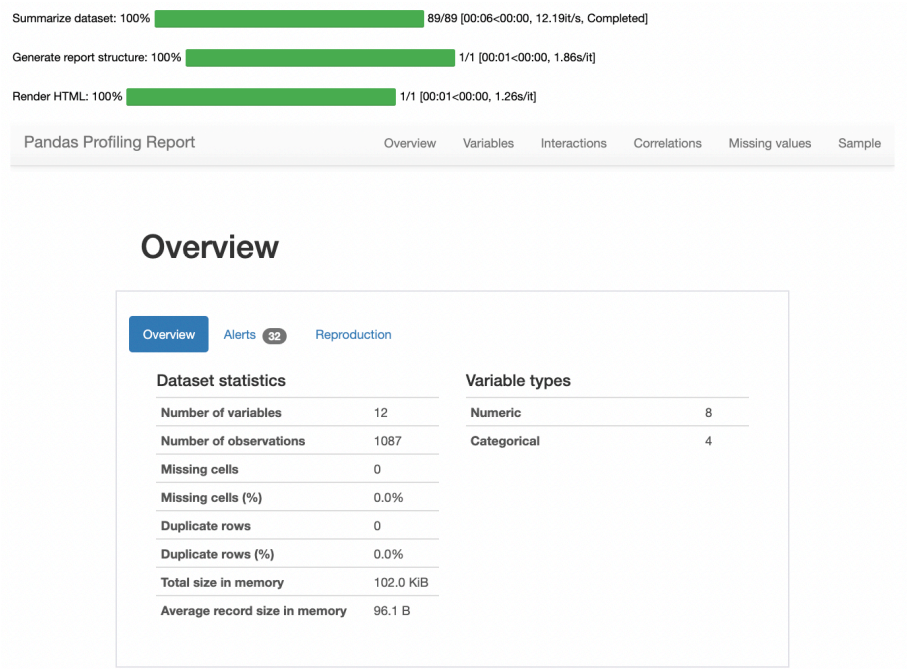
- Visualizations

I generated an initial thorough report on my dataframe values using the pandas profiling tool. It provides us with information like as correlations, missing values, duplicate rows, variable types, memory capacity, and so on for the produced dataset. This aids us in a more thorough visualisation by isolating each portion one by one and comparing and researching the effects of all the accessible feature columns on the prediction of our target label.

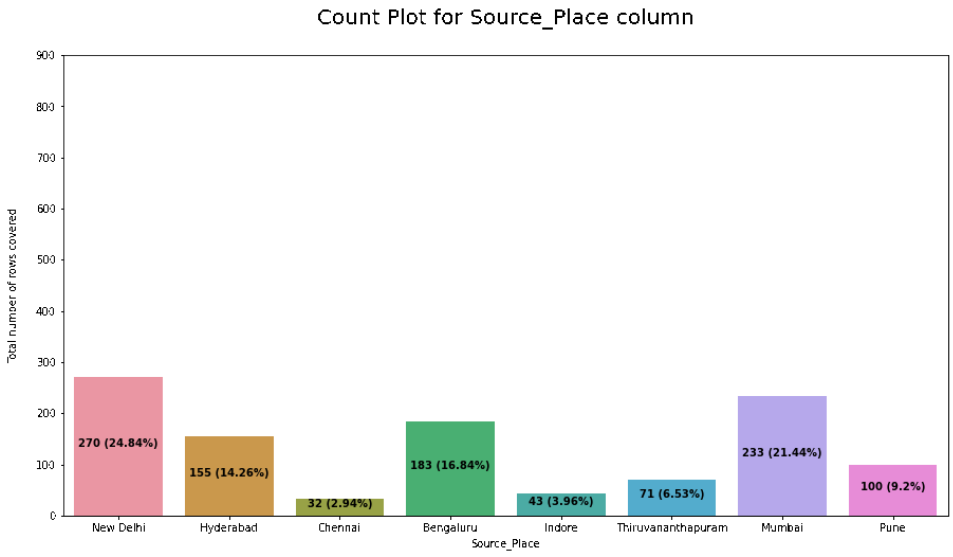
Code:

```
pandas_profiling.ProfileReport(df)
```

Output:



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



Observation Figure 1:

- Highest number of airline preferred by people are Indigo covering 37.26% of the total record.

- We can see that Go First is quite close to the first one and a close competitor standing at the second position holding 20.42% of the total record.
- At third place we have Vistara airlines that covers 17.66% of total record in our airline data.
- Airlines Air India, Air Asia and SpiceJet are the least used by people covering 14.63%, 2.58%, 0.18% and 0.37% respectively.

Observation Figure 2:

- Just as in case of departure area even in terms of arrival area or destination place people prefer to fly towards the city "New Delhi" covering 25.39% of record.
- Again in a similar fashion "Mumbai" city is a close second destination that people like to fly towards covering 21.53% record in the column.
- Surprisingly we have "Pune" city taking the third place for destination that people pefer landing covering 15.09% of record.
- Finally similar to the departure location the least travel to area is "Indore" and it covers 3.31% record in the column.

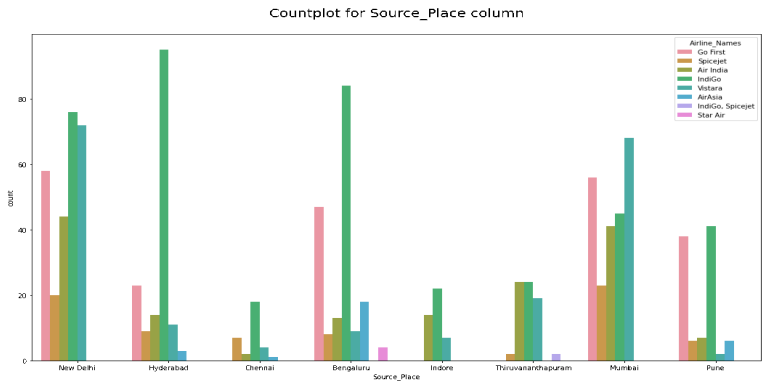
Observation Figure 3:

- People generally buy flight tickets that have 0 stop layover covering 70.1% rows in the column.
- Next in line are 1 stop layovers which cover 25.3% rows.
- Here we can see that the 2 stop flights availability is around 3.77% of the total record.
- In domestic flight we rarely see 3 or 4 stop layovers and therefore they cover 0.64% and 0.18% of total rows in the column respectively.

Code:

```
x = "Source_Place"
plt.figure(figsize=(18,10))
sns.countplot(x = x, hue = "Airline_Names", data = df)
plt.title(f"Countplot for {x} column\n", fontsize = 20)
plt.show()

x = "Destination_Place"
plt.figure(figsize=(18,10))
sns.countplot(x = x, hue = "Airline_Names", data = df)
plt.title(f"Countplot for {x} column\n", fontsize = 20)
plt.show()
```

Output:



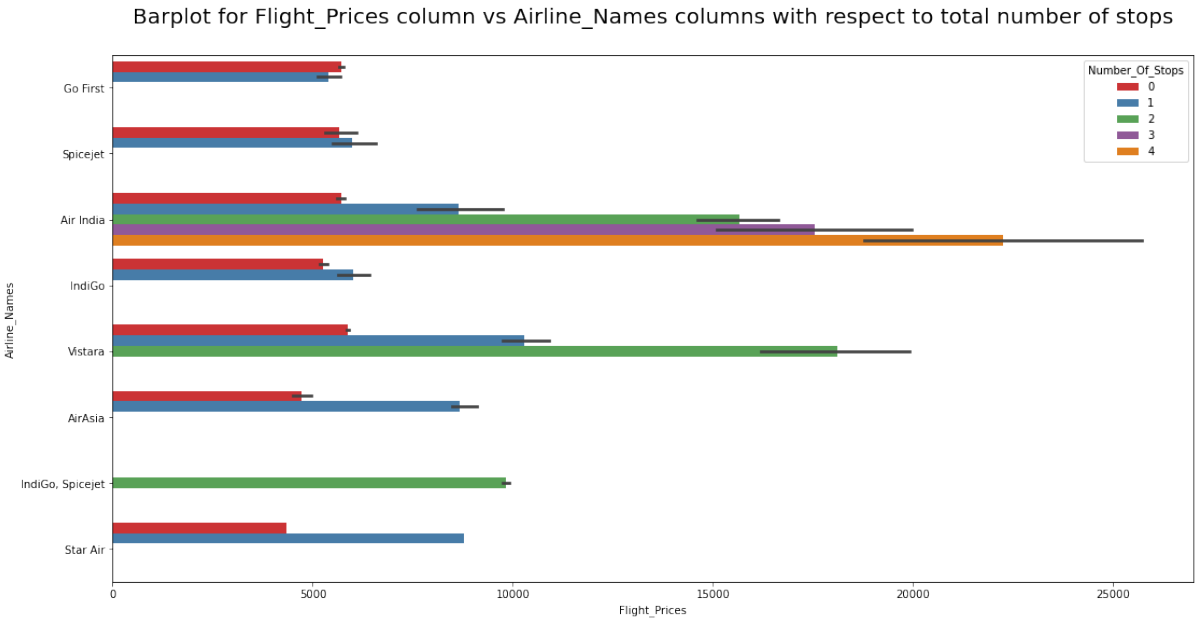
Observation:

- Checking out the Source place details for each and every airline we can see that Hyderabad city has the highest number of departure flights for Indigo airlines.
- Indigo, Vistara and Air India are the airlines that are used in almost all the cities to depart while the other airlines do not cover some or the other city.
- Looking at the Destination place details for each and every airline we can see that Pune city has the highest number of arrival flights for Indigo airlines.
- Once again we can observe that Go First, Spicejet, Indigo, Vistara and Air India are the leading airlines that are used in almost all cities to arrive while the other airlines miss out on some or the other regions.
- Overall I can notice that Air India, Vistara and Indigo flights do quite well and can be used for arrival and departure to and from any location in India.

Code:

```
x = "Flight_Prices"
y = "Airline_Names"


plt.figure(figsize=(18,9))
sns.barplot(x=df[x], y=df[y], hue=df['Number_Of_Stops'], palette="Set1")
plt.title(f"Barplot for {x} column vs {y} columns with respect to total number of stops\n", fontsize = 20)
plt.show()
```

Output:



Observation:

- If we see a trend the flights with 0 stops or rather direct flights for every airline is cheaper when compared to 1 or more layovers.
- Also flights with 2 and 3 stops have a considerably high price and number of flights available in those records are high too.
- Only Air India provides flights with 4 stop layover even in a domestic environment.
- SpiceJet, Go first, indigo, airindia, star air on the other hand only has flights available with 0 and 1 stop layovers.

Code:

```
y = "Flight_Prices"

x = "Airline_Names"
plt.figure(figsize = (15,8))
sns.barplot(data = df, y = y, x = x)
plt.title("Prices according to different Airlines\n", fontsize = 20)
plt.show()

x = "Source_Place"
plt.figure(figsize = (15,8))
sns.barplot(data = df, y = y, x = x)
plt.title("Prices according to different Source places\n", fontsize = 20)
plt.show()

x = "Destination_Place"
plt.figure(figsize = (15,8))
sns.barplot(data = df, y = y, x = x)
plt.title("Prices according to different Destination places\n", fontsize = 20)
plt.show()

x = "Number_Of_Stops"
plt.figure(figsize = (8,8))
sns.barplot(data = df, y = y, x = x)
plt.title("Prices according to different Number of layover stops\n", fontsize = 20)
plt.show()
```

Output:

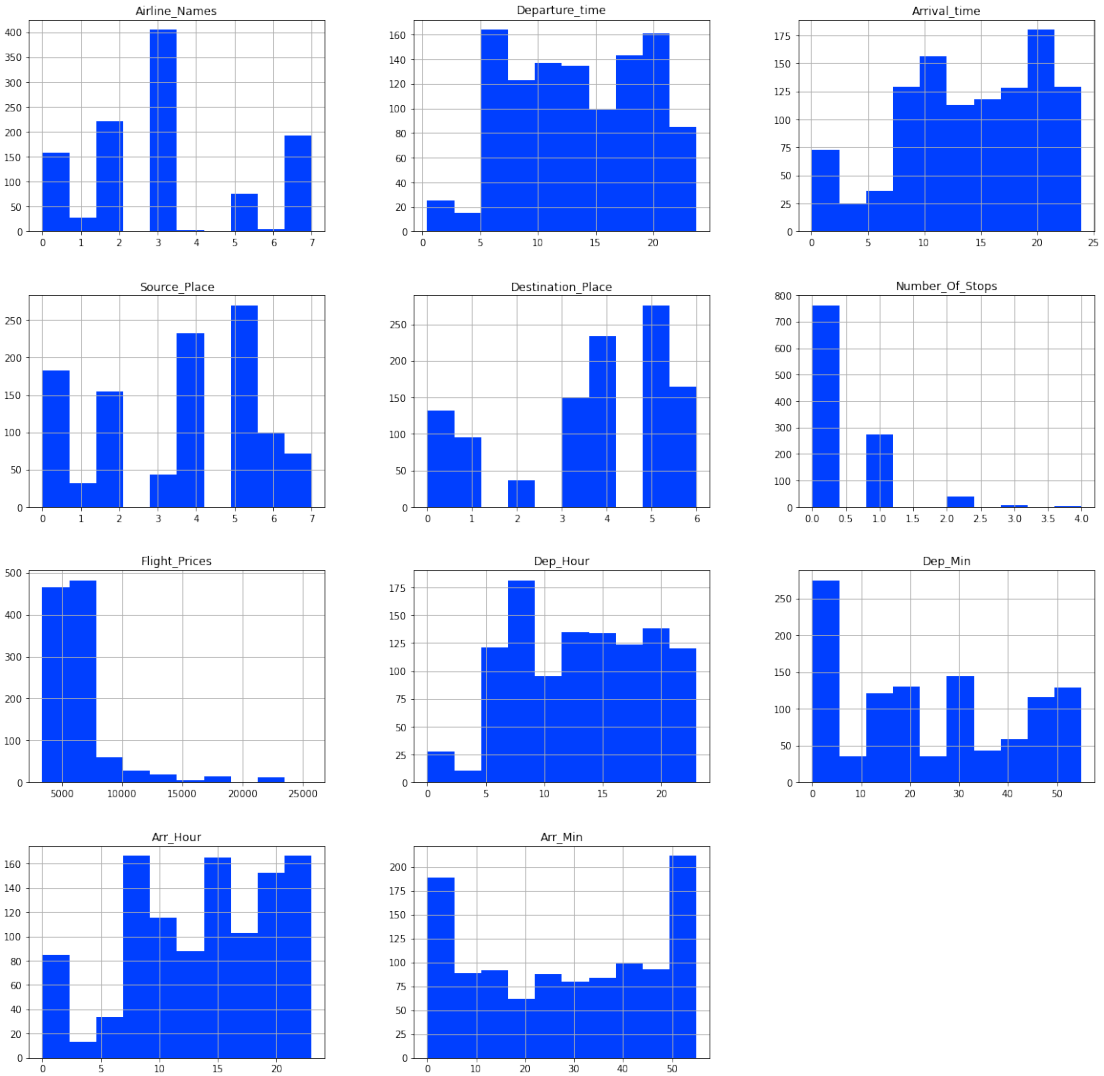
Observation:

- Airfares in Indigo, spicejet & Air India are pretty high when compared to other airlines.
- Flight prices when departing from cities like Thiruvanathpuram and Indore 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 from cities Mumbai and New delhi have high price range however the highest recorded is for Kolkota city and the rest fall in similar price bracket again not making any huge difference in savings.
- When we consider the layovers for pricing situtation then obviously direct flights are cheaper when compared to flights that have 1 or more stops.

Code:

```
plt.style.use('seaborn-bright')

df.hist(figsize=(20,20))
plt.show()
```

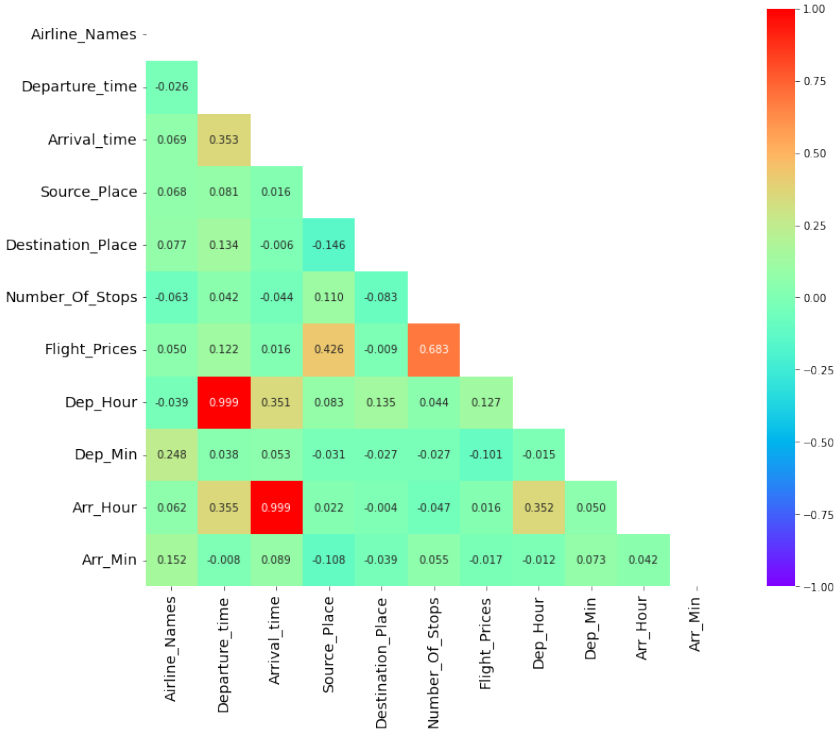
Output:



Code:

```
upper_triangle = np.triu(df.corr())
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(), vmin=-1, vmax=1, annot=True, square=True, fmt='0.3f',
            annot_kws={'size':10}, cmap="rainbow", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

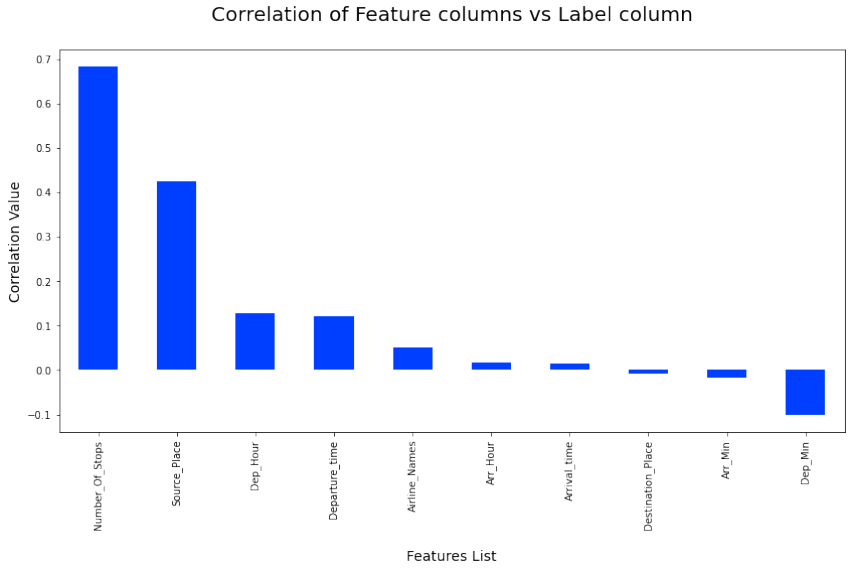
Output:



Code:

```
df_corr = df.corr()
plt.figure(figsize=(14,7))
df_corr['Flight_Prices'].sort_values(ascending=False).drop('Flight_Prices').plot.bar()
plt.title("Correlation of Feature columns vs Label column\n", fontsize=20)
plt.xlabel("\nFeatures List", fontsize=14)
plt.ylabel("Correlation Value", fontsize=14)
plt.show()
```

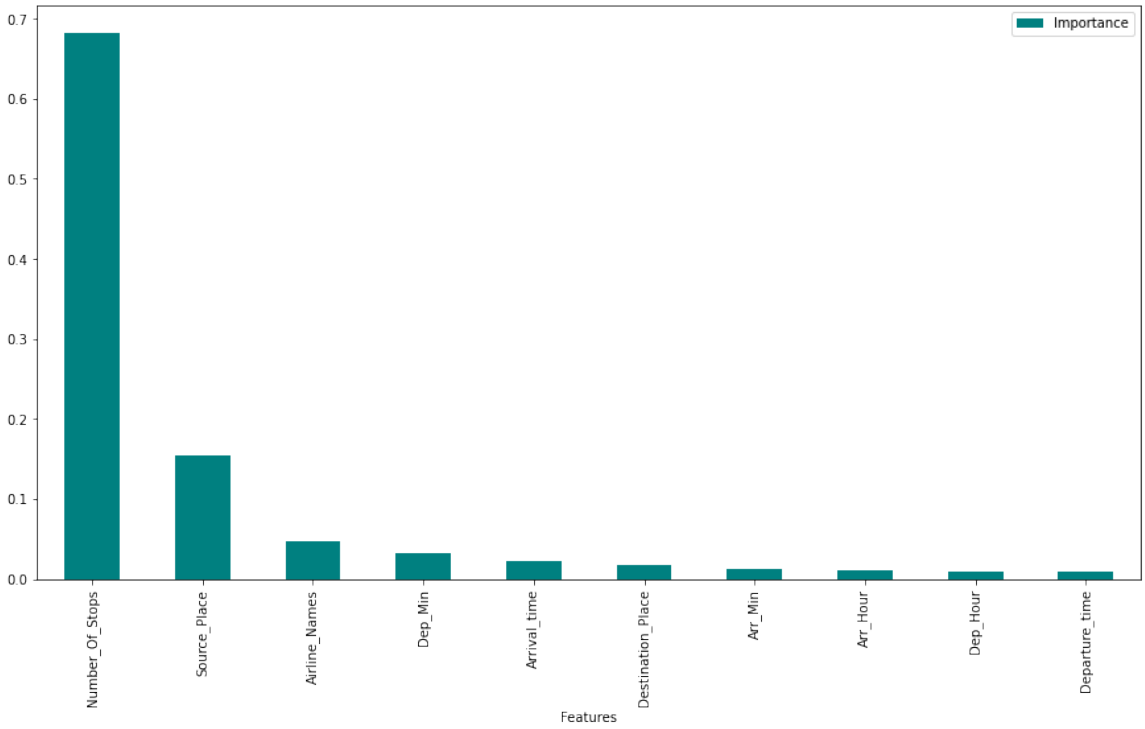
Output:



Code:

```
rf=RandomForestRegressor()
rf.fit(X_train, Y_train)
importances = pd.DataFrame({'Features':X.columns, 'Importance':np.round(rf.feature_importances_,3)})
importances = importances.sort_values('Importance', ascending=False).set_index('Features')
plt.rcParams["figure.figsize"] = (15,8)
importances.plot.bar(color='teal')
importances
```

Output:



- ## Interpretation of the Results

  We can readily comprehend the relationship between characteristics using the EDA above, and we can even identify which factors influence flight prices. To anticipate flight fares, these strategies include financial, marketing, and numerous societal elements. The number of individuals who fly has dramatically grown in recent years. Pricing alter dynamically owing to many variables, making it difficult for airlines to maintain prices. That is why we have attempted to fix this problem using machine learning. This can assist airlines in determining what rates they can keep. Customers may also use it to forecast future airline rates and plan their trip appropriately.

CONCLUSION

- Key Findings and Conclusions of the Study

  We grabbed flight data from airline websites for our research. The data frame is then loaded with the comma separated value file. Fortunately, our data collection does not include any missing values. Looking at the data collection, we see that various aspects must be processed, such as converting data types and extracting the actual value from string entries in time-related columns. After the data was analysed, I used EDA to determine the relationship between characteristics and the target variable. Travel duration, number of stops along the way, and food availability all have a part in determining flight pricing. As we've already seen, The forecast and the actual price from the trashed data set have a similar connection. This indicates that the model anticipated properly, and it may be able to assist airlines in determining what pricing they can keep. It may also assist clients in predicting future flight rates and planning their travel appropriately, as it is difficult for airlines to maintain pricing as they fluctuate owing to many factors. As a result, we can tackle this problem by employing Machine Learning approaches. The aforementioned research will assist our customer in studying the most recent flight price market, and with the model established, he will be able to easily forecast the price ranges of the flight, as well as understand how the flight price is determined.

- Learning Outcomes of the Study in respect of Data Science

  The data visualisation section assisted me in understanding the data by providing a graphical depiction of large amounts of data. It helped me comprehend the value of characteristics, discover outliers/skewness, and compare independent and dependent features. Because data cleansing is the most crucial element of model construction, I made sure the data was clean before starting.

I created several regression machine learning models in order to find the best one, and Extra Trees Regressor Model was the best based on the metrics I utilised.

- Limitations of this work and Scope for Future Work

Looking at the features, we discovered that there was a very small amount of features, resulting in lower R2 scores. Some algorithms have issues with overfitting, which might be due to the reduced amount of features in our dataset. We can improve our R2 score by getting more features through web scraping, which will also help us lessen the overfitting problem in our models. Another disadvantage of the study is that the data was collected in a volatile and shifting market. To be more specific, we used data from the period of the epidemic as well as recent data, so after the outbreak is over, the market may have a delayed correction. So, based on that, the determining variables may alter, and we've selected and gathered data from the most major cities in India. If the consumer is from a foreign nation, our algorithm may be unable to anticipate the flight's accurate price.