
# *Flight Price Prediction Project Report*

*Submitted by: PRATIK KUMAR*

# *INTRODUCTION*

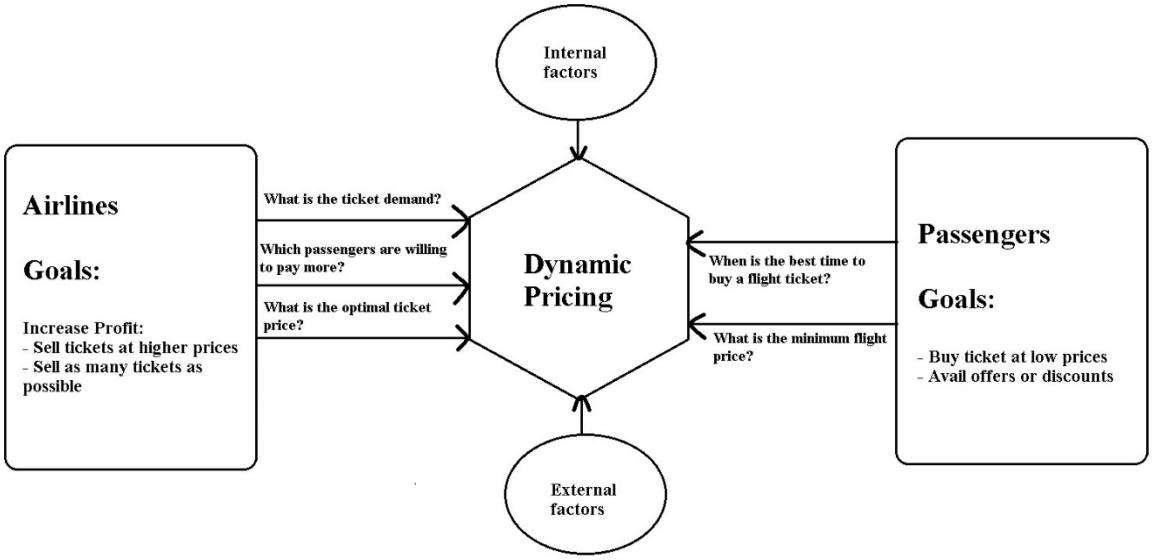
- *Business Problem Framing*

  *The airline industry is considered as one of the most sophisticated industry 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.*

  *Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on*

1. *Time of purchase patterns (making sure last-minute purchases are expensive)*

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




- *Conceptual Background of the Domain Problem*

  *Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.*

- *Review of Literature*

  *As per the requirement of client, I have scrapped the data from online sites and based on that data I have did analysis like for based on which feature of my data prices are changing. and checked the relationship of flight price with all the feature like what flight he*

  *should choose.*

- *Motivation for the Problem Undertaken*
  *I have worked on this on the bases of client requirements and followed all the steps till model deployment.*

## *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 9 columns (8 features and 1 label) with 5805 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*
  - *Meal_Availability : Provides us with the information on type of meal that the passenger is eligible for*
  - *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*

*We can see our dataset includes a target label "Used Car Price" column and the remaining feature columns can be used to determine or help in predicting the price of the used cars. Since price is a continuous value it makes this to be a Regression problem!*

```
df = pd.read_csv("Flight_Price_Data.csv")
```

I am importing the collected dataset comma separated values file and storing it into our dataframe for further usage.

```
df # checking the first 5 and last 5 rows
```

| | Airline_Names | Departure_Time | Arrival_Time | Flight_Duration | Source_Place | Destination_Place | Meal_Availability | Number_Of_Stops | Flight_Prices |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Air Asia | 12:40 | 20:15 | 7h 35m | New Delhi | Mumbai | No Meal Fare | 1 Stop | 5,953 |
| **1** | Air Asia | 11:55 | 20:15 | 8h 20m | New Delhi | Mumbai | No Meal Fare | 1 Stop | 5,953 |
| **2** | Air Asia | 16:15 | 06:20 | 14h 05m | New Delhi | Mumbai | No Meal Fare | 1 Stop | 5,953 |
| **3** | Go First | 18:50 | 20:45 | 1h 55m | New Delhi | Mumbai | No Meal Fare | Non Stop | 5,954 |
| **4** | Go First | 09:05 | 11:05 | 2h 00m | New Delhi | Mumbai | No Meal Fare | Non Stop | 5,954 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **5800** | Air India | 08:55 | 08:20 | 23h 25m | Lucknow | Jaipur | No Meal Fare | 1 Stop | 9,302 |
| **5801** | Air India | 08:55 | 09:20 | 24h 25m | Lucknow | Jaipur | No Meal Fare | 2 Stop(s) | 16,287 |
| **5802** | Air India | 14:45 | 09:20 | 18h 35m | Lucknow | Jaipur | No Meal Fare | 2 Stop(s) | 16,865 |
| **5803** | Air India | 08:55 | 09:20 | 24h 25m | Lucknow | Jaipur | No Meal Fare | 2 Stop(s) | 16,865 |
| **5804** | Air India | 15:30 | 09:20 | 17h 50m | Lucknow | Jaipur | Free Meal | 3 Stop(s) | 19,749 |

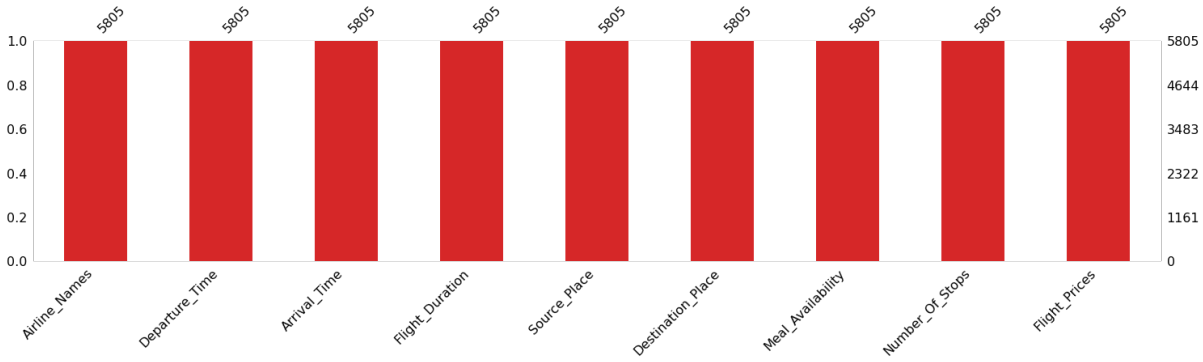
5805 rows × 9 columns

- *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.*

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

```
Airline_Names          0
Departure_Time         0
Arrival_Time           0
Flight_Duration        0
Source_Place           0
Destination_Place      0
Meal_Availability      0
Number_Of_Stops        0
Flight_Prices          0
dtype: int64
```




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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5805 entries, 0 to 5804
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Airline_Names      5805 non-null   object
 1   Departure_Time     5805 non-null   object
 2   Arrival_Time       5805 non-null   object
 3   Flight_Duration    5805 non-null   object
 4   Source_Place       5805 non-null   object
 5   Destination_Place  5805 non-null   object
 6   Meal_Availability  5805 non-null   object
 7   Number_Of_Stops    5805 non-null   object
 8   Flight_Prices      5805 non-null   object
dtypes: object(9)
memory usage: 408.3+ KB
```

*I also took a look at all the unique value present in each of the columns and then decided to view the details for those column values that had less than 10 categories.*

```
df.nunique().sort_values().to_frame("Unique Values")
```

| | Unique Values |
|---|---|
| Meal_Availability | 3 |
| Number_Of_Stops | 5 |
| Airline_Names | 6 |
| Source_Place | 9 |
| Destination_Place | 9 |
| Departure_Time | 223 |
| Arrival_Time | 232 |
| Flight_Duration | 409 |
| Flight_Prices | 1571 |

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

```
# Meal_Availability

df.Meal_Availability.replace({"No Meal Fare": "No Meals", "Free Meal": "Free Meals", "eCash 250": "eCash Meals"},
                             inplace = True)
df["Meal_Availability"].value_counts()
```

```
# Number_Of_Stops

df.Number_Of_Stops.replace({"Non Stop": 0, "1 Stop": 1, "2 Stop(s)": 2, "3 Stop(s)": 3, "4 Stop(s)": 4},
                           inplace = True)
df["Number_Of_Stops"].value_counts()
```

```
# 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()
```

```
# Flight_Duration

df["FD_Hour"] = df.Flight_Duration.str.split('h').str.get(0)
df["FD_Min"] = df.Flight_Duration.str.split('h').str.get(1)
df["FD_Min"] = df["FD_Min"].str.split('m').str.get(0)
df["FD_Hour"] = df['FD_Hour'].astype('float')
df["FD_Min"] = df['FD_Min'].astype('float')
df["Flight_Duration"] = df["FD_Hour"] + df["FD_Min"] / 60
#df.drop(columns = ["FD_Hour","FD_Min"], inplace=True)
df.head()
```

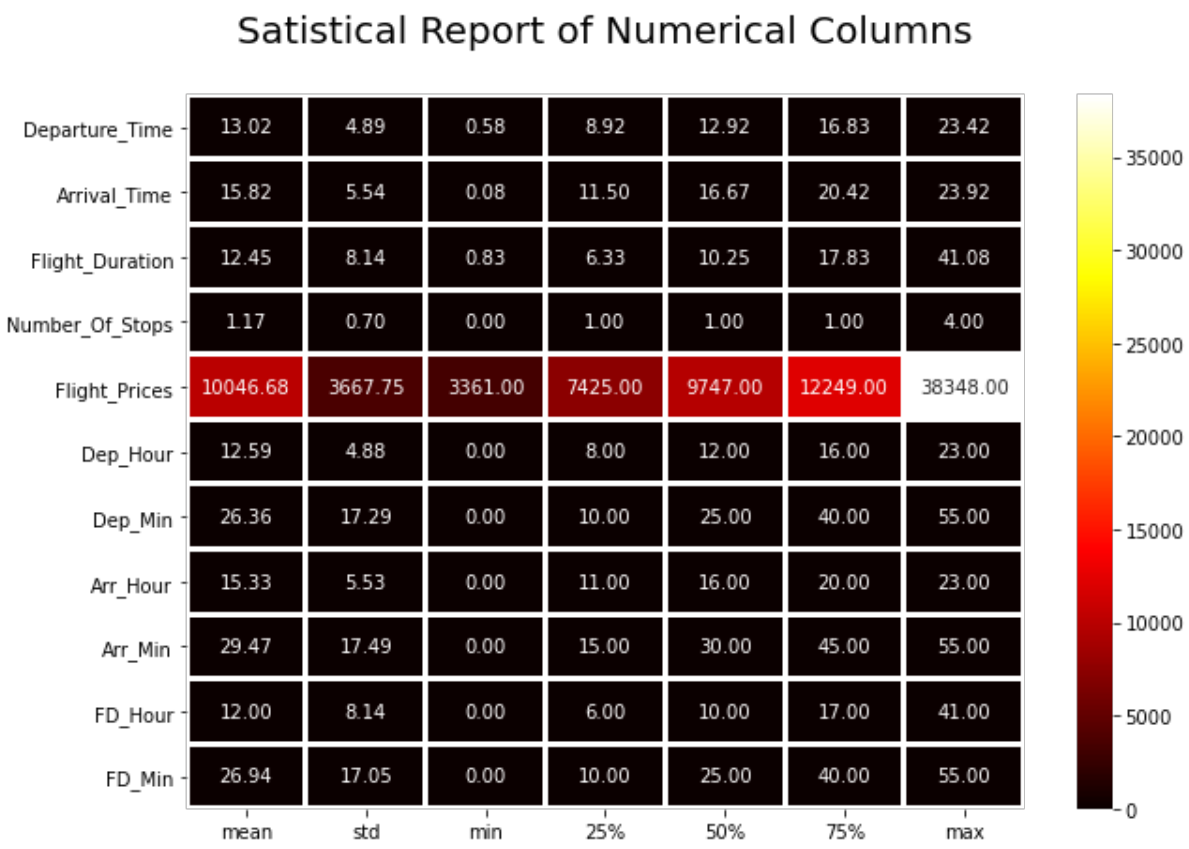
```
# Flight_Prices

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

*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** | 5805 | 6 | IndiGo | 1782 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Departure_Time** | 5805.0 | NaN | NaN | NaN | 13.024591 | 4.887243 | 0.583333 | 8.916667 | 12.916667 | 16.833333 | 23.416667 |
| **Arrival_Time** | 5805.0 | NaN | NaN | NaN | 15.823572 | 5.543619 | 0.083333 | 11.5 | 16.666667 | 20.416667 | 23.916667 |
| **Flight_Duration** | 5805.0 | NaN | NaN | NaN | 12.452728 | 8.137841 | 0.833333 | 6.333333 | 10.25 | 17.833333 | 41.083333 |
| **Source_Place** | 5805 | 9 | Mumbai | 806 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Destination_Place** | 5805 | 9 | Mumbai | 805 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Meal_Availability** | 5805 | 3 | No Meals | 4252 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **Number_Of_Stops** | 5805.0 | NaN | NaN | NaN | 1.172782 | 0.696018 | 0.0 | 1.0 | 1.0 | 1.0 | 4.0 |
| **Flight_Prices** | 5805.0 | NaN | NaN | NaN | 10046.68062 | 3667.752804 | 3361.0 | 7425.0 | 9747.0 | 12249.0 | 38348.0 |
| **Dep_Hour** | 5805.0 | NaN | NaN | NaN | 12.585185 | 4.881136 | 0.0 | 8.0 | 12.0 | 16.0 | 23.0 |
| **Dep_Min** | 5805.0 | NaN | NaN | NaN | 26.364341 | 17.288619 | 0.0 | 10.0 | 25.0 | 40.0 | 55.0 |
| **Arr_Hour** | 5805.0 | NaN | NaN | NaN | 15.332472 | 5.526615 | 0.0 | 11.0 | 16.0 | 20.0 | 23.0 |
| **Arr_Min** | 5805.0 | NaN | NaN | NaN | 29.465978 | 17.493203 | 0.0 | 15.0 | 30.0 | 45.0 | 55.0 |
| **FD_Hour** | 5805.0 | NaN | NaN | NaN | 12.00379 | 8.137239 | 0.0 | 6.0 | 10.0 | 17.0 | 41.0 |
| **FD_Min** | 5805.0 | NaN | NaN | NaN | 26.936262 | 17.047469 | 0.0 | 10.0 | 25.0 | 40.0 | 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*

*The input data were initially all object datatype so had to clean the data by removing unwanted information like "h" and "m" from Flight_Duration column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.*

*Code:*

```
# Ordinal Encoder

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

column=["Meal_Availability", "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.*

## *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.*

```
import warnings
warnings.simplefilter("ignore")
warnings.filterwarnings("ignore")
import joblib

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
```

*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: 2741.9142718035005
R2 Score is: 41.30962845018751
Cross Validation Score: 33.96122010005579
R2 Score - Cross Validation Score is 7.34840835013172
```

- *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.*
*Code:*

```
# 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: 72.4377190016525
```

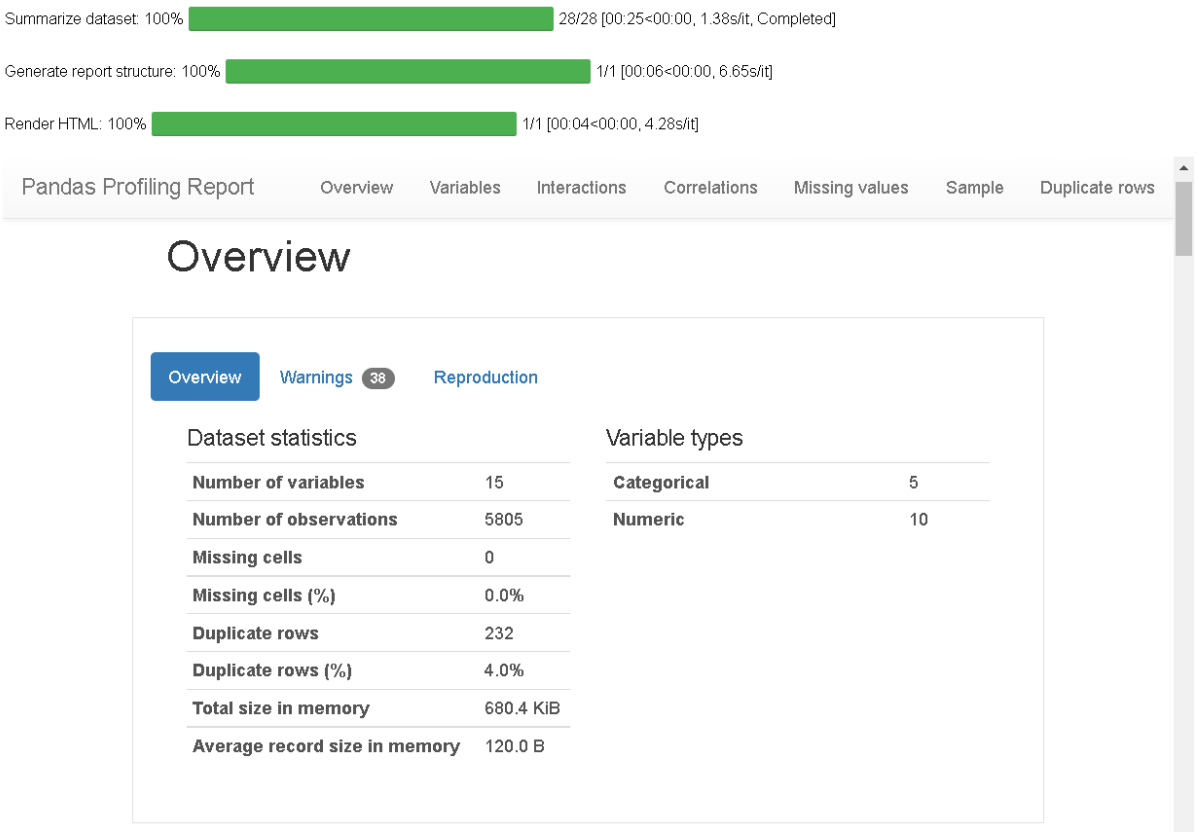
- *Visualizations*

*I 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.*

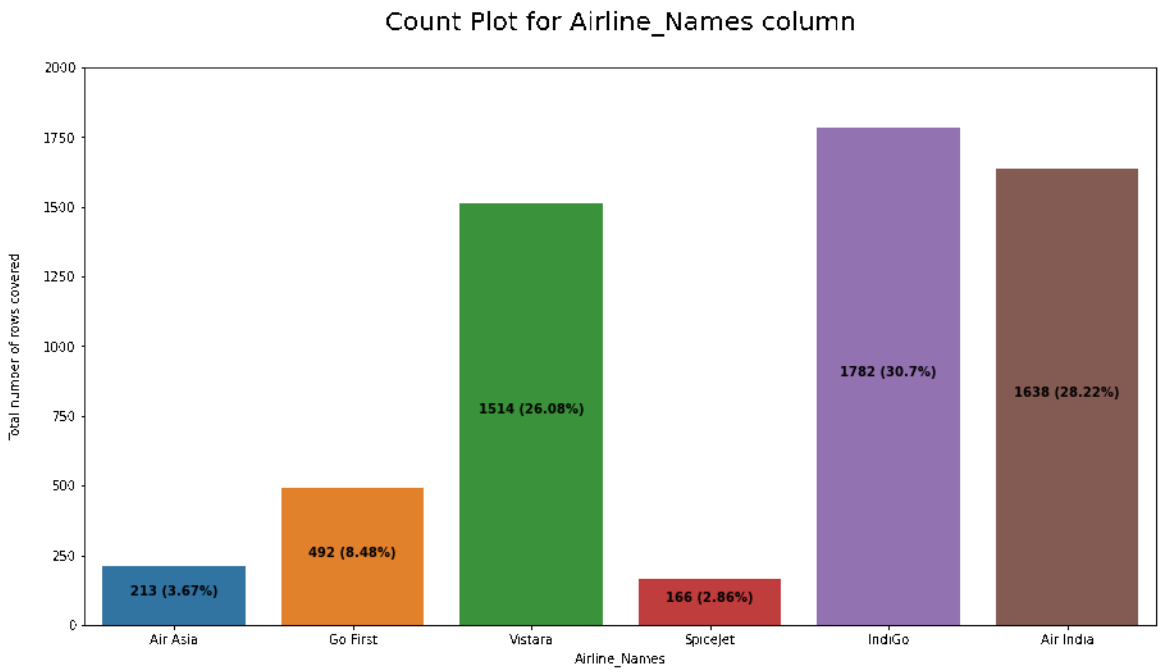
*Code:*

```
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.*




*<u>Observation on Fig 1:</u>*

*There are 73.25% flights that mostly serve no meals in their domestic journey since they are of short distances and duration*

*We can see that 17.47% flights serve free meals which are probably for tickets that include those prices and meal services*

*Finally, there are 9.29% flights which offer the eCash meals option that can be redeemed to purchase food during long journey flights mostly with multiple stops*

*<u>Observation on Fig 2:</u>*

*Highest number of airlines preferred by people are Indigo covering 30.7% of the total record*

*We can see that Air India is quite close to the first one and a close competitor standing at the second position holding 28.22% of the total record*

*At third place we have Vistara airlines that covers 26.08% of total record in our airline data*

*Airlines Go First, Air Asia and SpiceJet are the least used by people covering 8.48%, 3.67% and 2.86% respectively*

*Observation on Fig 3:*

*The departure area or source place highly used or people majorly flying from the city is "Mumbai" covering 13.88% record in the column*

*We see that "Bangalore" is a close second wherein it covers 13.01% records in the column*

*Other two famous locations where people chose to fly from are "New Delhi" and "Kolkata" covering 12.89% and 12.82% respectively*

*The least travel from location is "Jaipur" where I believe not many people chose to travel from unless they are returning from a vacation there and hence covers 7.36% of record in the column*

*Observation on Fig 4:*

*Just as in case of departure area even in terms of arrival area or destination place people prefer to fly towards the city "Mumbai" covering 13.87% of record*

*Again, in a similar fashion "Bangalore" city is a close second destination that people like to fly towards covering 13.54% record in the column*

*Surprisingly we have "Hyderabad" city taking the third place for destination that people prefer landing covering 13.07% of record*

*Finally, similar to the departure location the least travel to area is "Jaipur" and it covers 5.24% record in the column*

*Observation on Fig 5:*

*People generally buy flight tickets that have 1 stop layover covering 63.67% rows in the column*

*Next in line are 2 stop layovers which cover 19.88% rows*

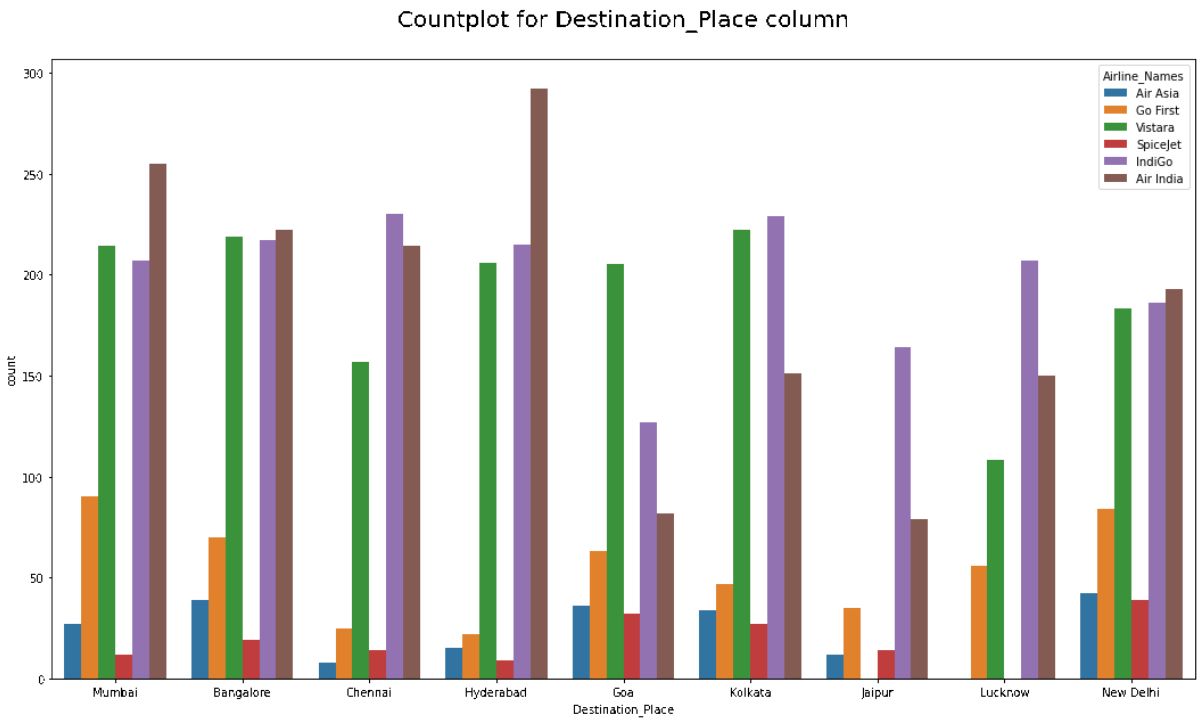
*Here we can see that the 0 stop flights availability is around 11.94% of the total record*

*In domestic flight we rarely see 3 or 4 stop layovers and therefore they cover 4.2% and 0.31% 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 Mumbai city has the highest number of departure flights for Air India airlines*

*Go First, Indigo 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 Hyderabad city has the highest number of arrival flights for Air India airlines*

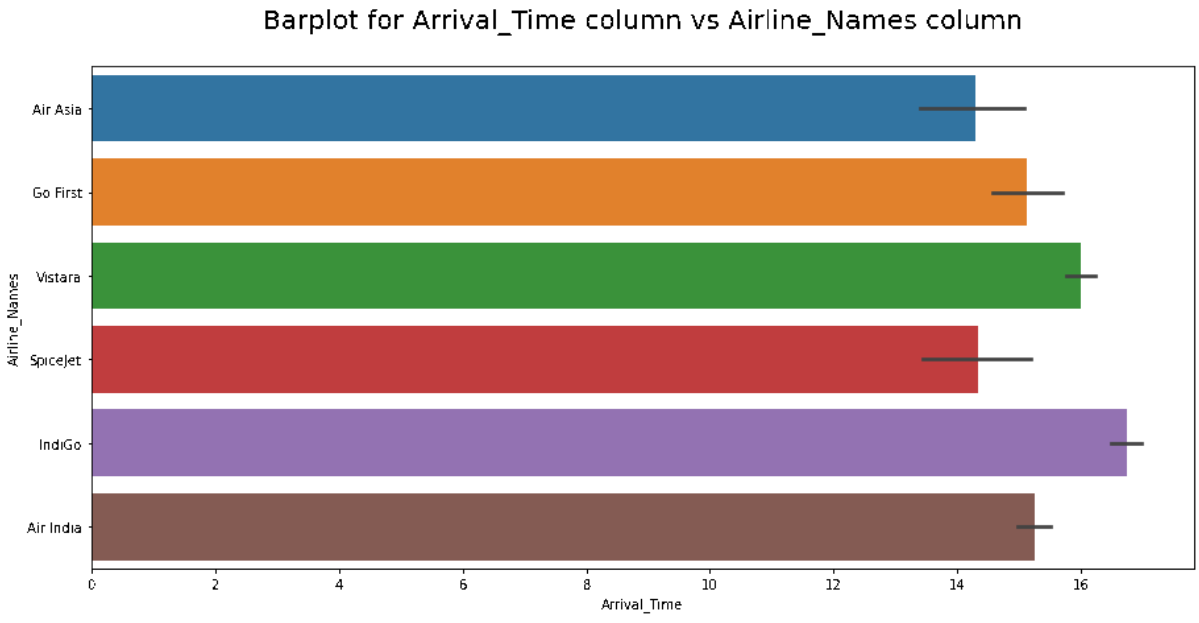
*Once again, we can observe that Go First, Indigo and Air India are the leading airlines that are used in almost all cities to arrive while the other airlines miss out on some or the other regions*

*Overall, I can notice that Air India and Indigo flights do quite well and can be used for arrival and departure to and from any location in India*

*Code:*

```
y = 'Airline_Names'

x = 'Departure_Time'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()

x = 'Arrival_Time'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()

x = 'Flight_Duration'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()

x = 'Flight_Prices'
plt.figure(figsize=[15,7])
sns.barplot(x,y,data=df,orient='h')
plt.title(f"Barplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()
```

*Output:*



*Observation:*

*When we observe the barplot for Departure time vs Airline we can see that Air Asia has the highest departure time while IndiGo has the lowest departure time*

*Considering the barplot for Arrival time vs Airline we can see that IndiGo has the highest arrival time while Air Asia and SpiceJet have the lowest arrival time*

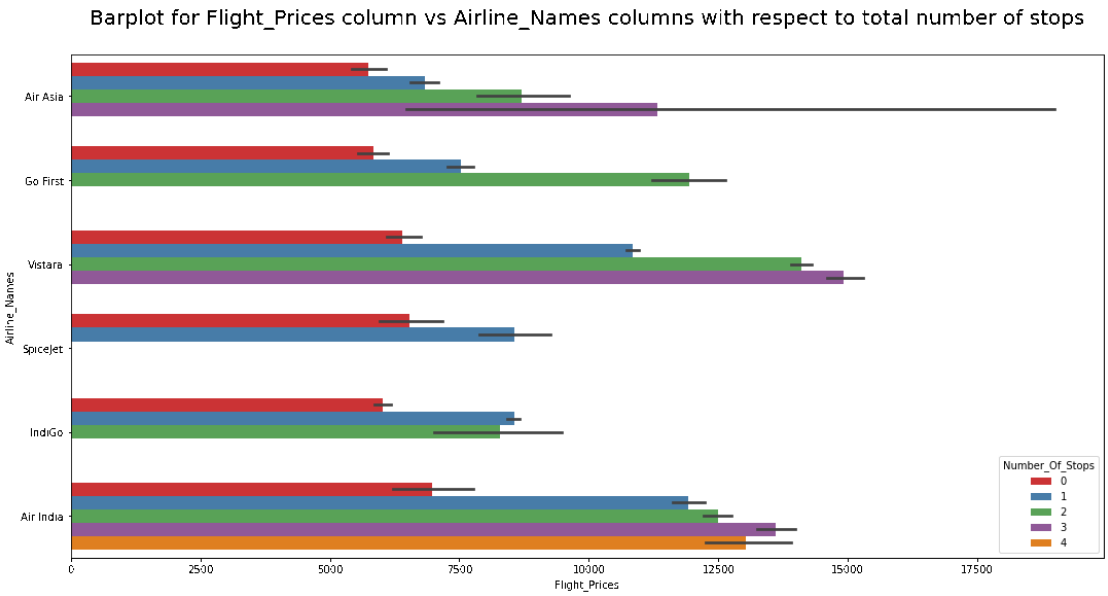
*Taking a look at the barplot for Flight duration vs Airline we observe that Ai India has the highest flight duration while Indigo has the lowest flight duration collectively*

*Comparing the bar plots for Flight prices vs Airline we can clearly see that Vistara and Air India have very high flight prices while the other airlines IndiGo, SpiceJet, Go First and Air Asia lie in a similar price bracket where Air Asia has the lowest fare*

*Code:*

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

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

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:*

*Vistara and Air India are the only airlines that have free meal service on their flights but also have a very high no meal record*

*Also, the eCash meal options do not work for Vistara and Air India flights and they have very high flight prices compared to their competitors*

*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*

*And 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 layovers even in a domestic environment*

*SpiceJet 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()

x = "Meal_Availability"
plt.figure(figsize = (8,8))
sns.barplot(data = df, y = y, x = x)
plt.title("Prices according to different Meal options\n", fontsize = 20)
plt.show()
```

*Output:*


*Observation:*

*Airfares in Vistara and Air India are pretty high when compared to other airlines as they provide free meal service which probably is just meal cost included in the tickets*

*Flight prices when departing from cities like Kolkata and Chennai 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 Kolkata and Chennai have high price range however the highest recorded is for Lucknow city and the rest fall in similar price bracket again not making any huge difference in savings*

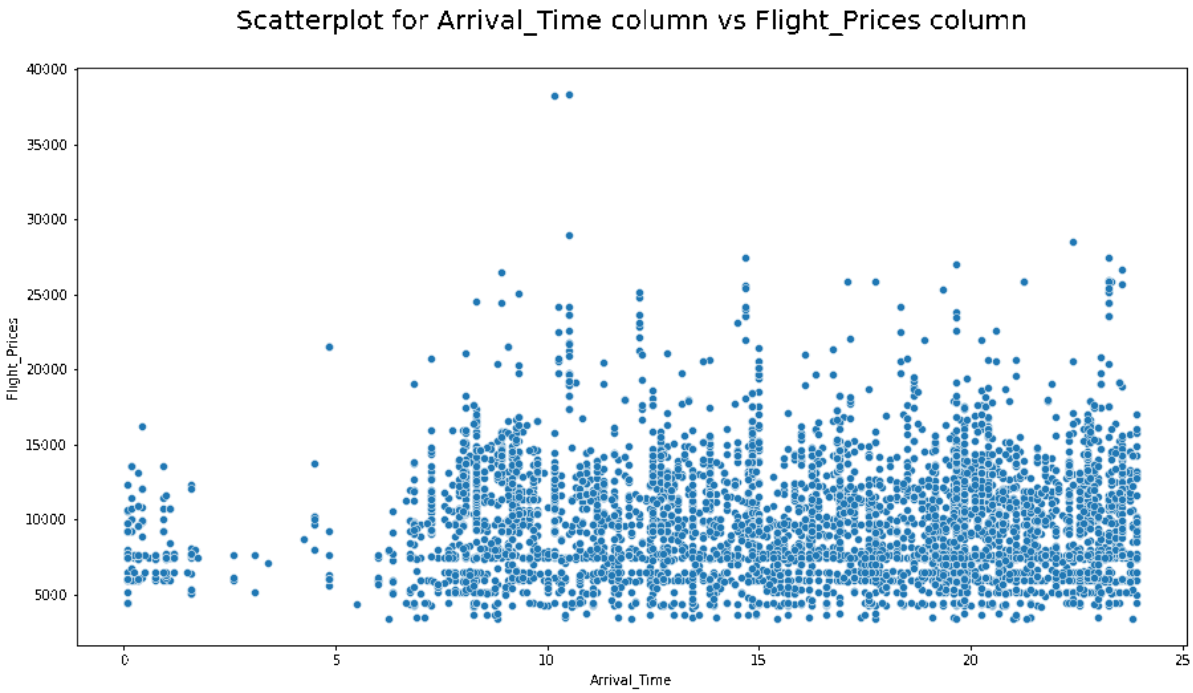
*When we consider the layovers for pricing situation then obviously direct flights are cheaper when compared to flights that have 1 or more stops*

*As per the data collected, we see that free meal service flight tickets cost the highest as compared to no meal service flight tickets and the lowest price observed are for eCash meal service wherein the frequent flyers can use their flight points as discounts on the ticket prices*

*Code:*

```
y = "Flight_Prices"

x = "Departure_Time"
plt.figure(figsize = (15,8))
sns.scatterplot(x=x,y=y,data=df)
plt.title(f"Scatterplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()

x = "Arrival_Time"
plt.figure(figsize = (15,8))
sns.scatterplot(x=x,y=y,data=df)
plt.title(f"Scatterplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()

x = "Flight_Duration"
plt.figure(figsize = (15,8))
sns.scatterplot(x=x,y=y,data=df)
plt.title(f"Scatterplot for {x} column vs {y} column\n", fontsize = 20)
plt.show()
```

*Output:*




*Observation:*

*In the above scatter plot, we see the comparison where flight prices are the highest during peak hours as compared to late hours or specifically graveyard timings for departure*

*We can see a similar trend in flight price scatter plot details when it comes to arrival timings where we see a decrease in prices between 1:00 hrs and 6:00 hrs*

*Finally, as the overall flight duration increases the flight prices increase too and that makes direct flight depart and arrive in a relatively shorter period of time*

*Code:*

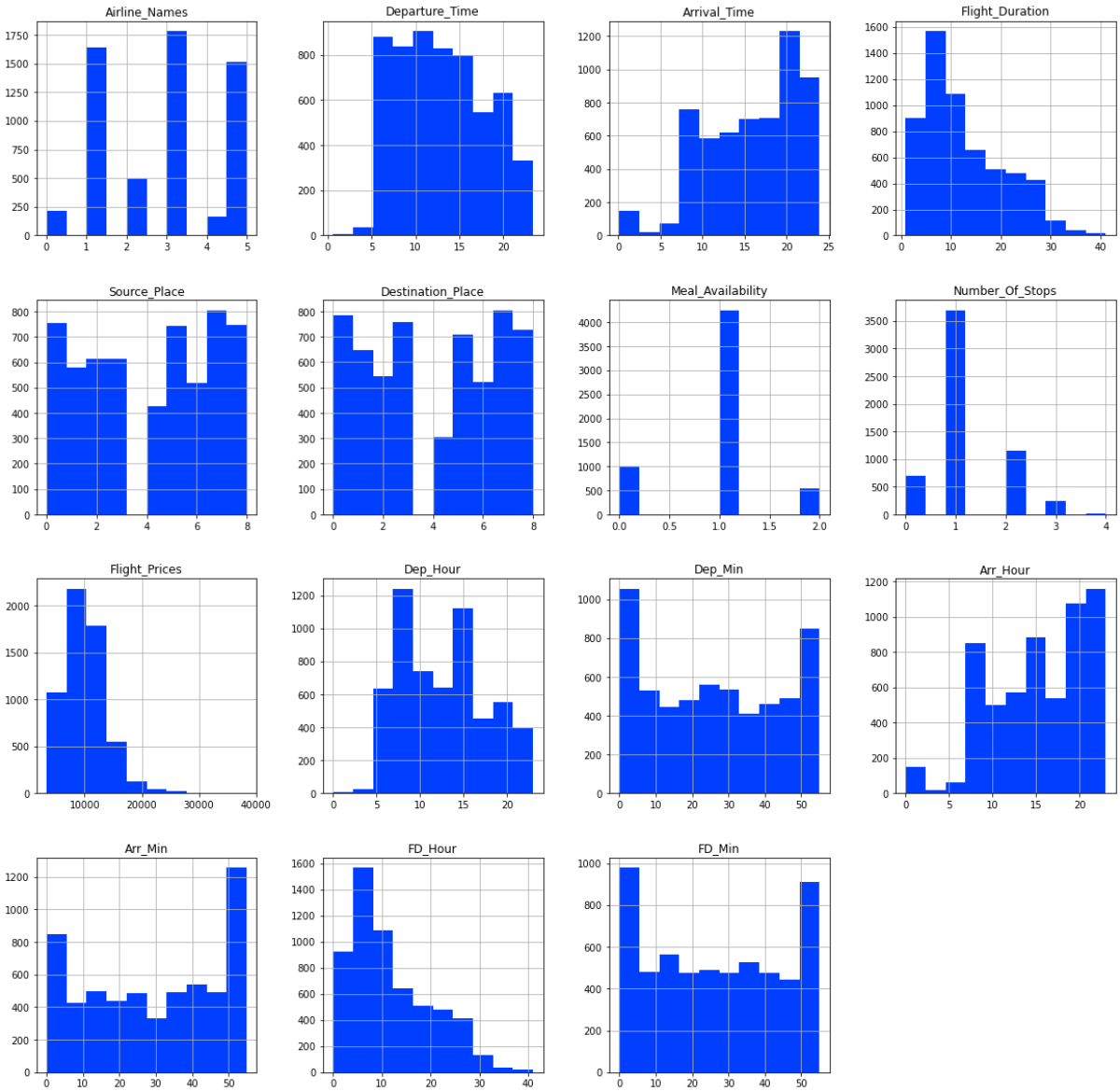
```
print("************************************ Pair Plot with Meal Type Legend ************************************")
sns.pairplot(df, hue='Meal_Availability', diag_kind="kde", kind="scatter", palette="Set1", height=3.5)
plt.show()
```

*Output:*

*Code:*

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

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

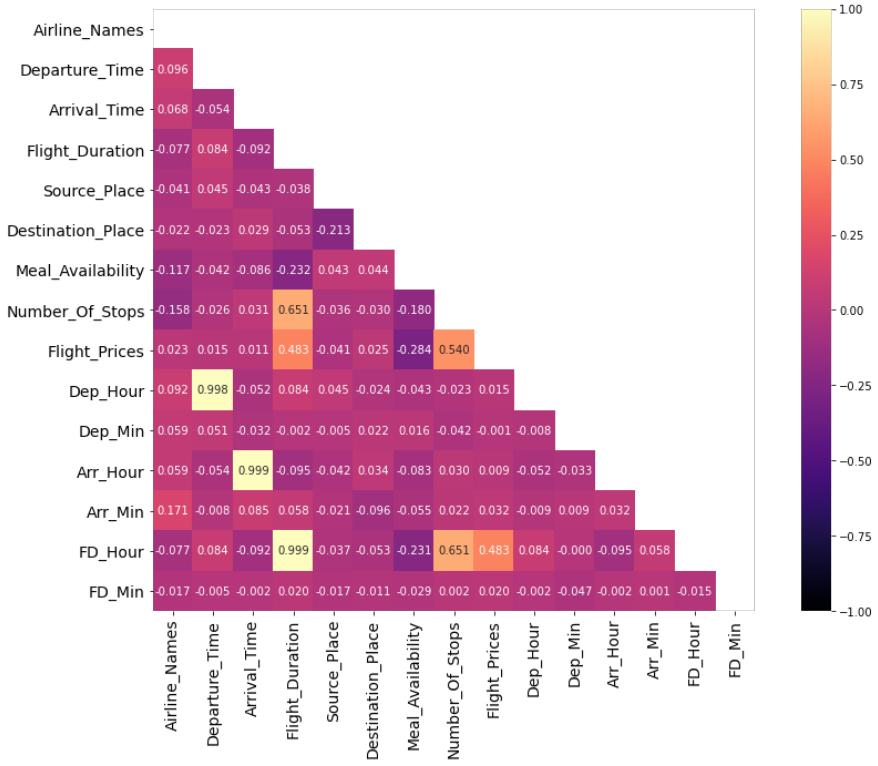
*Output:*




*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="magma", mask=upper_triangle)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

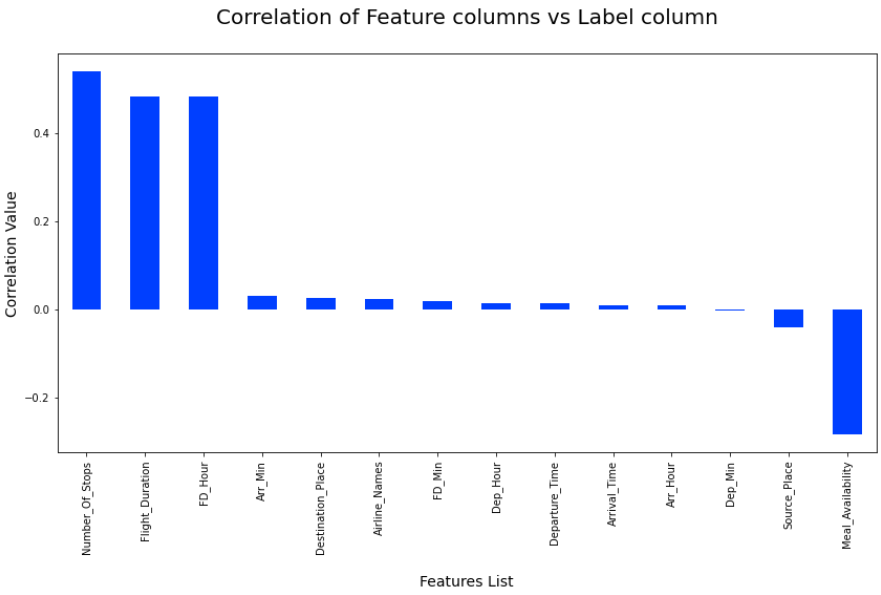
*Output:*




## *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()
```

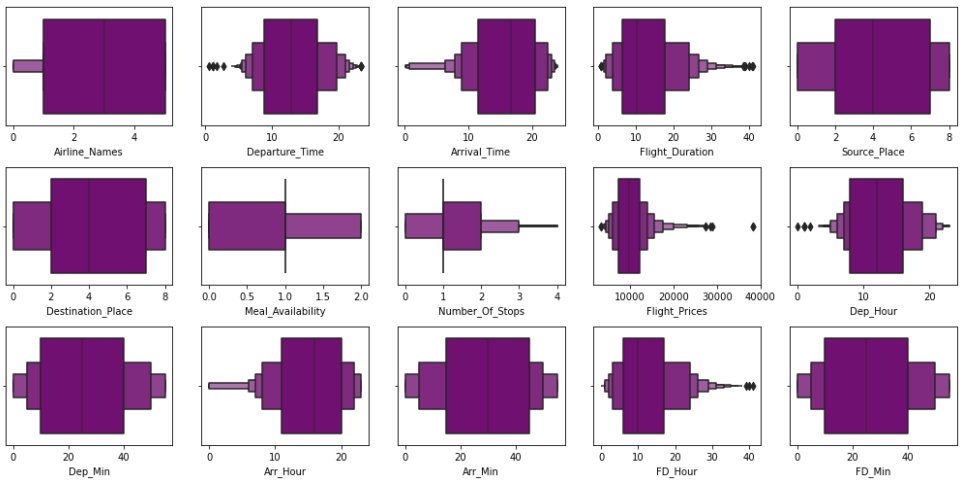
## *Output:*



## *Outliers Code:*

```
plt.figure(figsize=(14,7))
outl_df = df.columns.values
for i in range(0, len(outl_df)):
    plt.subplot(3, 5, i+1)
    ax = sns.boxenplot(df[outl_df[i]], color='purple')
    plt.tight_layout()
```

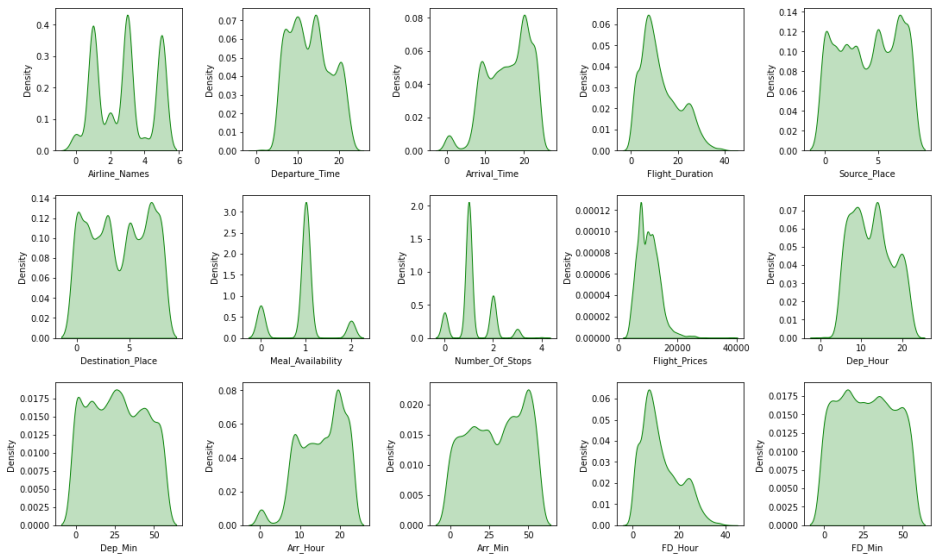
## *Output:*




## *Skewness Code:*

```
fig, ax = plt.subplots(ncols=5, nrows=3, figsize=(15,9))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.distplot(value, ax=ax[index], hist=False, color="g", kde_kws={"shade": True})
    index += 1
plt.tight_layout(pad=0.8, w_pad=0.8, h_pad=2.0)
plt.show()
```

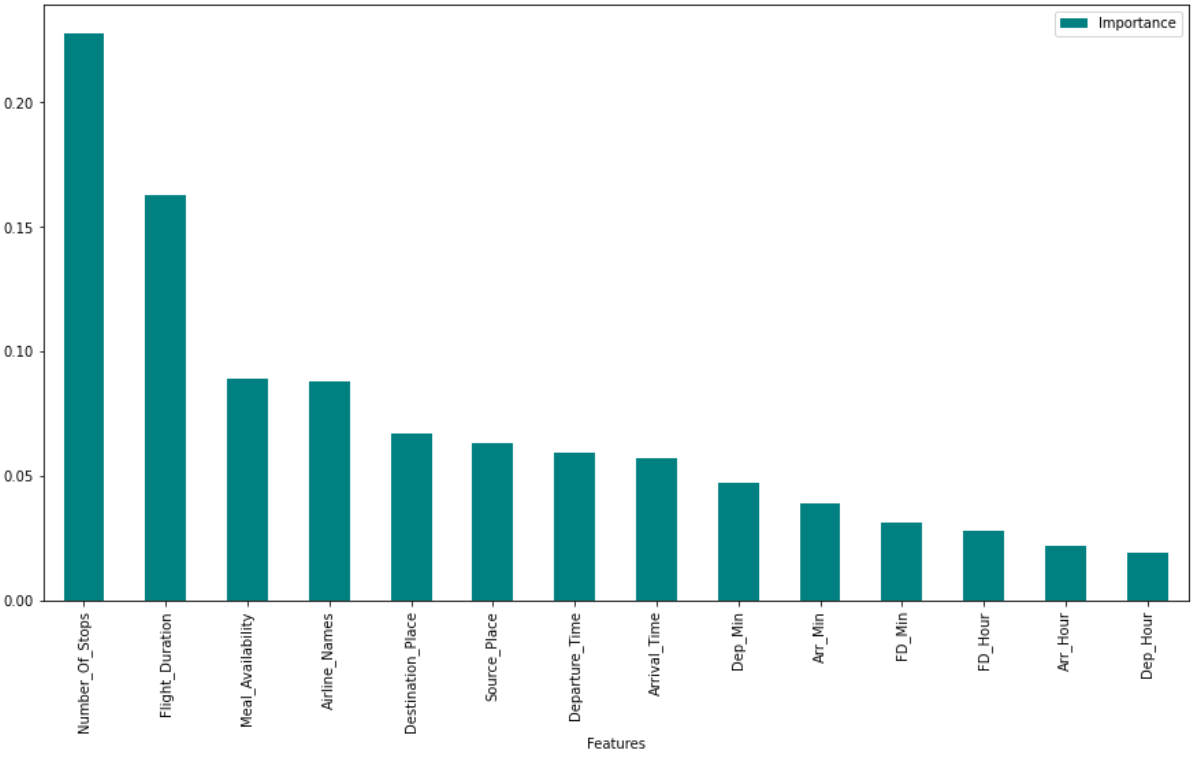
## *Output:*




## *Importance Graph 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*

  *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.*

*THANK YOU*