

Flight Price Prediction **Report**

# Problem Definition :

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable. Here you will be provided with prices of flight tickets for various airlines between March and June of 2019 and between various cities.

Size of training set: 10683 records

Size of test set: 2671 records

## FEATURES

Airline: The name of the airline.

Date\_of\_Journey: The date of the journey

Source: The source from which the service begins.

Destination: The destination where the service ends.

Route: The route was taken by the flight to reach the destination.

Dep\_Time: The time when the journey starts from the source.

Arrival\_Time: Time of arrival at the destination.

Duration: Total duration of the flight.

Total\_Stops: Total stops between the source and destination.

Additional\_Info: Additional information about the flight

(Target Variable) Price: The price of the ticket

**Working first on Train Data**

# Data Analysis :

## Data Shape

**There are 10683 Rows and 11 Columns in the train data.**

## Data Info

## This method prints a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and columns, non-null values and memory usage.

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 10683 entries, 0 to 10682  Data columns (total 11 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Airline 10683 non-null object  1 Date\_of\_Journey 10683 non-null object  2 Source 10683 non-null object  3 Destination 10683 non-null object  4 Route 10682 non-null object  5 Dep\_Time 10683 non-null object  6 Arrival\_Time 10683 non-null object  7 Duration 10683 non-null object  8 Total\_Stops 10682 non-null object  9 Additional\_Info 10683 non-null object  10 Price 10683 non-null int64 |

**dtypes: int64(1), object(10)**

**memory usage: 918.2+ KB**

## Data types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are classes and variables are instances (objects) of these classes.

|  |
| --- |
| Airline object  Date\_of\_Journey object  Source object  Destination object  Route object  Dep\_Time object  Arrival\_Time object  Duration object  Total\_Stops object  Additional\_Info object  Price int64  dtype: object |

**All the columns are in String Data types except for the last column, which also happens to be our Target Variable "Price".**

## Data Description :

**Data.Describe()**

The describe () function computes a summary of statistics on the DataFrame columns. This function gives the mean, std, and IQR values. And, function excludes the character columns and given summary about numeric columns.

|  |
| --- |
| Price  count  10683.000000  mean  9087.064121  std  4611.359167  min  1759.000000  25%  5277.000000  50%  8372.000000  75%  12373.000000  max  79512.000000 |

**In the column "Price", the Mean or the Average is 9087.06, Smallest and the Biggest values are 1759.0 & 79512.0 respectively**.

## 

## Checking For Null Values :

**Data.Isnull().sum()**

Null values appear when some fields are left blank for some records of your Data.IsNull() method is useful, to evaluate whether any value is missing in a data set. To determine how many missing values exist in the collection, in which case you can use .sum() chained on.

|  |
| --- |
| Airline 0  Date\_of\_Journey 0  Source 0  Destination 0  Route 1  Dep\_Time 0  Arrival\_Time 0  Duration 0  Total\_Stops 1  Additional\_Info 0  Price 0  dtype: int64 |

**There are 2 Null Values present in the data. One in the "Destination" column and the other in the "Total\_stops"**

# 

# 

# EDA Concluding Remarks :

1. **In the column "Price", the Mean or the Average is 9087.06, Smallest and the Biggest values are 1759.0 & 79512.0 respectively**.
2. **There are 2 Null Values present in the data. One in the "Destination" column and the other in the "Total\_stops"**
3. **All the columns are in String Data types except for the last column, which also happens to be our Target Variable "Price".**

# Pre-processing Pipeline :

## Data Cleaning :

Data Cleaning attempts to impute missing values, smooth out noise, resolve inconsistencies, removing outliers in the data.

df=df.dropna()

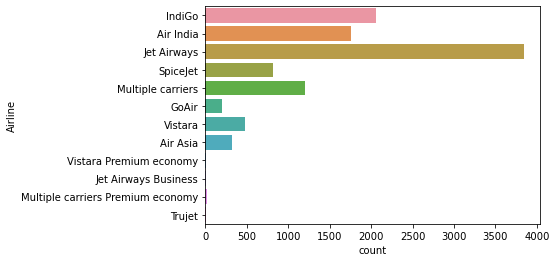
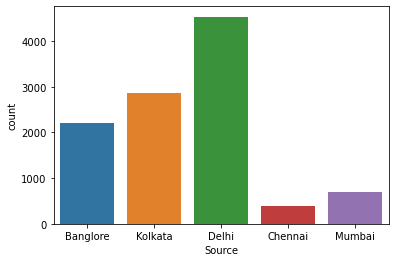
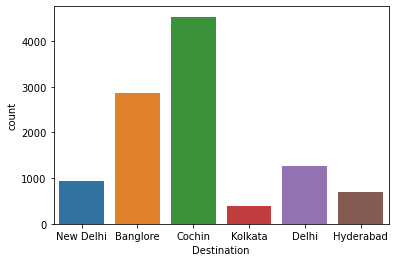
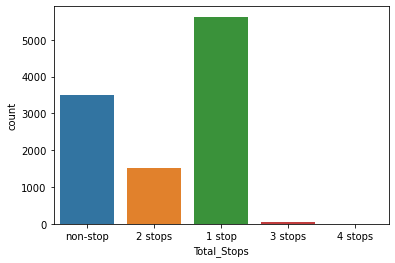
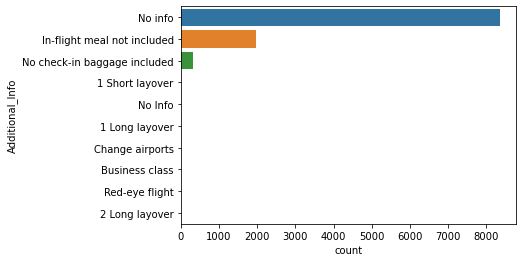
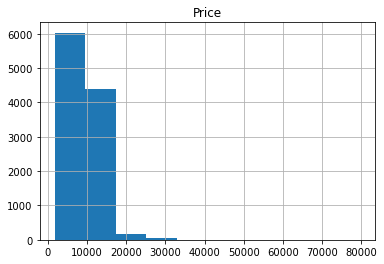
dropna() is an inbuilt DataFrame function that is used to remove rows and columns with Null/None/NA values from DataFrame.After Dropping the Null Values in the data there was a loss of one row containing the null values.

(10683, 11)➡(10682, 11)

## Data Visualization :

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends, and correlations that might not otherwise be detected can be exposed.

## **Visualisation Observations :**

1. **As per the visualization, 'Jet Airways' has the highest number of Flights and 'Vistara premium Economy', 'Jet Airways Business' and, 'Trujet' has the least number of Flights.**
2. **'Delhi' is the most frequent Boarding Airport, more than 4000 Flights were Boarded from Delhi, and 'Chennai' being the least boarded Airport where only around 500 Flights were boarded by their passengers.**
3. **More than 4000 Flights were completed in "Cochin", and the least flights were destined for "Kolkata" i.e. 500.**
4. **Most Flights are having 1 Stop i.e. more than 5000 flights, And there are nearly no flights with 4 Stops. And around 3500 flights are Non-Stop.**
5. **Well, For additional information more than 8000 flights have 'No-Info', But there are still 9 different Additional pieces of information provided for the rest of the flights.**
6. **More than 6000 flights are priced between 1000-9000, Around 4400 Flights are priced between 9000-17000, Around 100 Flights are priced Around 17000-25000 and About 100 flights are priced Around 25000-32000.**

## **Checking for Correlation:**

Correlation analysis is a statistical method used to evaluate the strength of the relationship between two quantitative variables. A high correlation means that two or more variables have a strong relationship with each other, while a weak correlation means that the variables are hardly related.

|  |
| --- |
| Price 1.000000  Route 0.164149  Arrival\_Time 0.027358  Source 0.015999  Dep\_Time 0.002931  Date\_of\_Journey -0.036907  Airline -0.039565  Additional\_Info -0.065463  Destination -0.071122  Duration -0.144280  Total\_Stops -0.571221  Name: Price, dtype: float64 |

**Variable 'Route' is Highly correlated to the Target Variable, Whereas 'Total\_Stops' is Least Correlated to the target variable “Price”.**

## Data Transformation :

## **Encoding the data:**

An [Ordinal Encoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html) is used to encode categorical features into an ordinal numerical value (ordered set). This approach transforms categorical value to numerical value in ordered sets.

|  |
| --- |
| from sklearn.preprocessing import OrdinalEncoder  enc=OrdinalEncoder()  for i in df.columns:  if df[i].dtypes=="object":  df[i]=enc.fit\_transform(df[i].values.reshape(-1,1)) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Airline** | **Date\_of\_Journey** | **Source** | **Destination** | **Route** | **Dep\_Time** | **Arrival\_Time** | **Duration** | **Total\_Stops** | **Additional\_Info** | **Price** |  | | **0** | 3.0 | 24.0 | 0.0 | 5.0 | 18.0 | 211.0 | 233.0 | 240.0 | 4.0 | 8.0 | 3897 | | **1** | 1.0 | 6.0 | 3.0 | 0.0 | 84.0 | 31.0 | 906.0 | 336.0 | 1.0 | 8.0 | 7662 | |

Transforming categorical data into numeric data for processing the data further.

## **Skewness :**

Skewness is a measure of the asymmetry of the probability distribution of a random variable about its mean. In other words, skewness tells you the amount and direction of skew (departure from horizontal symmetry). The skewness value can be positive or negative, or even undefined.

|  |
| --- |
| Airline 0.731057  Date\_of\_Journey -0.070718  Source -0.424023  Destination 1.244046  Route -0.501911  Dep\_Time 0.194914  Arrival\_Time -0.606676  Duration -0.213498  Total\_Stops 0.631532  Additional\_Info -1.779689  Price 1.812405  dtype: float64 |

**Keeping +/-(0.5) as the range of skewness. There are 6 columns out of the range of skewness.**

|  |
| --- |
| from sklearn.preprocessing import power\_transform  df.new=power\_transform(x)  df.new=pd.DataFrame(df.new,columns=x.columns) |

**Using Power Transform to fix the Skewness in the data.**

## Building Machine Learning Models :

The goal in this step is to develop a benchmark model that serves us as a baseline, upon which we will measure the performance of a better and more tuned algorithm.

## **Finding best random state :**

|  |
| --- |
| for i in range(0,100):  x\_train, x\_test, y\_train, y\_test=train\_test\_split(x, y, test\_size=0.2, random\_state=i)  dtr.fit(x\_train,y\_train)  pred\_train=dtr.predict(x\_train)  pred\_test=dtr.predict(x\_test)  print (f"At randon state {i}, the training accuracy is:- {r2\_score(y\_train, pred\_train)}")  print (f"At random state {i}, the testing accuracy is:{r2\_score(y\_test,pred\_test)}")  print("\n") |

|  |
| --- |
| At randon state 94, the training accuracy is:- 0.9956634247320154  At random state 94, the testing accuracy is:0.8567252461837069 |

**Best Accuracy is for Decision Tree Regressor model at Random State 94.**

## **Creating Train Test Split :**

|  |
| --- |
| x\_train, x\_test, y\_train, y\_test=train\_test\_split(x, y, test\_size=0.2, random\_state=94) |

## **Cross-validation of the model :**

|  |
| --- |
| from sklearn.model\_selection import cross\_val\_score  for j in range (2,10):  cv\_score=cross\_val\_score (lr , x, y, cv=j)  cv\_mean=cv\_score.mean()  print(f" At cross fold {j} the cv score is {cv\_mean} and accuracy score for training is {Train\_accuracy} and accuracy score for testing is {Test\_accuracy}")  print("\n") |

**Since the number of folds don't have such impact on the accuracy and cv\_score. So cv=5 is selected.**

## **Regularization:**

To mitigate the problem of overfitting and underfitting Regularization Methods are used: Lasso

|  |
| --- |
| from sklearn.linear\_model import Lasso  parameters = {'alpha' : [.0001, .001, .01, .1, 1, 10], 'random\_state' :list (range(0,10))}  ls= Lasso()  clf= GridSearchCV(ls, parameters)  clf.fit(x\_train, y\_train)  print (clf.best\_params\_) |

|  |
| --- |
| {'alpha': 1, 'random\_state': 0} |

We get Best Parameter 'alpha': 1

|  |
| --- |
| cv\_score=cross\_val\_score (ls , x, y,cv=5)  cv\_mean=cv\_score.mean()  cv\_mean |

|  |
| --- |
| 0.2590803037317851 |

## 

## **Ensemble Technique :**

|  |
| --- |
| from sklearn.ensemble import RandomForestRegressor  rf=RandomForestRegressor (criterion="mse", max\_features="auto")  rf.fit(x\_train, y\_train)  rf.score(x\_train,y\_train)  pred\_decision =rf.predict(x\_test)  rfs = r2\_score(y\_test, pred\_decision)  print('R2 core:' , rfs\*100 )    rfscore = cross\_val\_score( rf,x,y,cv=5)  rfc = rfscore.mean()  print('Cross Val Score:',rfc\* 100) |

Random Forest Regressor gives us the maximum accurate prediction of 88%

## **Saving Model:**

|  |
| --- |
| import pickle  filename = 'FPP\_train\_project.pkl'  pickle.dump(rf, open( filename, 'wb' )) |

## 

## 

## **Conclusion :**

|  |
| --- |
| loaded\_model = pickle.load(open('FPP\_train\_project.pkl', 'rb'))  result = loaded\_model.score(x\_test, y\_test)  print(result) |

0.8873199262028327

|  |
| --- |
| conclusion=pd.DataFrame([loaded\_model.predict(x\_test)[:], pred\_decision[:]], index=["Predicted", "orginal"])  conclusion |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Predicted** | 2281.39 | 10352.51 | 12801.78 | 29344.24 | 5228.0 | 10733.81 | 4036.57 | 6380.54 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **orginal** | 2281.39 | 10352.51 | 12801.78 | 29344.24 | 5228.0 | 10733.81 | 4036.57 | 6380.54 |

Working on Test Data

For testing data set will do all process like data cleaning, EDA process, visualtization.

Will not create a model instead, will load the model created with training dataset and will predict the price.

# Data Analysis :

## Data Shape

**There are 2671 Rows and 10Columns in the train data.**

## Data Info

## This method prints a concise summary of a DataFrame. This method prints information about a DataFrame including the index dtype and columns, non-null values, and memory usage.

|  |
| --- |
| <class 'pandas.core.frame.DataFrame'>  RangeIndex: 2671 entries, 0 to 2670  Data columns (total 10 columns):  # Column Non-Null Count Dtype  --- ------ -------------- -----  0 Airline 2671 non-null object  1 Date\_of\_Journey 2671 non-null object  2 Source 2671 non-null object  3 Destination 2671 non-null object  4 Route 2671 non-null object  5 Dep\_Time 2671 non-null object  6 Arrival\_Time 2671 non-null object  7 Duration 2671 non-null object  8 Total\_Stops 2671 non-null object  9 Additional\_Info 2671 non-null object |

**dtypes: object(10)**

**memory usage: 208.8+ KB**

## Data types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are classes and variables are instances (objects) of these classes.

|  |
| --- |
| Airline object  Date\_of\_Journey object  Source object  Destination object  Route object  Dep\_Time object  Arrival\_Time object  Duration object  Total\_Stops object  Additional\_Info object  dtype: object |

**All the values are in the 'Object' data type throughout the test Data.**

## Checking For Null Values :

**Data.Isnull().sum()**

Null values appear when some fields are left blank for some records of your Data.IsNull() method is useful, to evaluate whether any value is missing in a data set. To determine how many missing values exist in the collection, in which case you can use .sum() chained on.

|  |
| --- |
| Airline 0  Date\_of\_Journey 0  Source 0  Destination 0  Route 0  Dep\_Time 0  Arrival\_Time 0  Duration 0  Total\_Stops 0  Additional\_Info 0  dtype: int64 |

**There are no Null Values in the Data.**

# EDA Concluding Remarks :

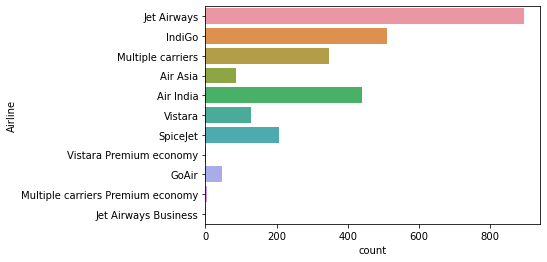
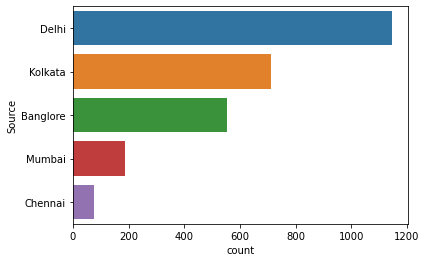
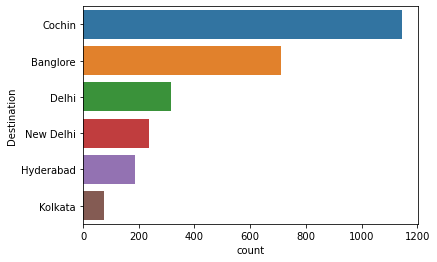
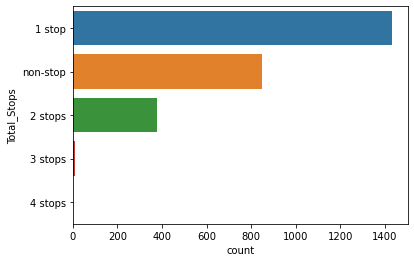
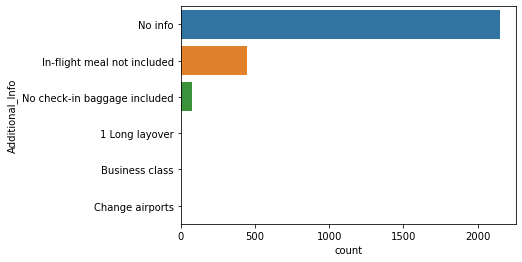
1. **All the columns are in String Datatypes.**
2. **There are no Null Values in the Data.**

# Pre-processing Pipeline :

## Data Visualization :

Data visualization is the discipline of trying to understand data by placing it in a visual context so that patterns, trends, and correlations that might not otherwise be detected can be exposed.

## **Visualisation Observations :**

1. **As per the visualization, 'Jet Airways' has the highest number of Flights and 'Vistara premium Economy', 'Jet Airways Business' and, 'Trujet' has the least number of Flights.**
2. **'Delhi' is the most frequent Boarding Airport, more than 1100 Flights were Boarded from Delhi, and 'Chennai' being the least boarded Airport where only around 50 Flights were boarded by their passengers.**
3. **More than 1100 Flights were completed in "Cochin", and the least flights were destined for "Kolkata" i.e. 50.**
4. **Most Flights are having 1 Stop i.e. more than 1400 flights, And there are nearly no flights with 4 Stops. And around 900 flights are Non-Stop.**
5. **Well, For additional information more than 2000 flights have 'No-Info', But there are still 5 different Additional pieces of information provided for the rest of the flights.**

## Data Transformation :

## **Encoding the data:**

An [Ordinal Encoder](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html) is used to encode categorical features into an ordinal numerical value (ordered set). This approach transforms categorical value to numerical value in ordered sets.

|  |
| --- |
| **from sklearn.preprocessing import LabelEncoder**  **le=LabelEncoder()**  **for i in df.test.columns:**  **if df.test[i].dtypes=="object":**  **df.test[i]=le.fit\_transform(df.test[i].values.reshape(-1,1))** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Airline** | **Date\_of\_Journey** | **Source** | **Destination** | **Route** | **Dep\_Time** | **Arrival\_Time** | **Duration** | **Total\_Stops** | **Additional\_Info** |  |
| **0** | **4** | **39** | **2** | **1** | **76** | **138** | **150** | **10** | **0** | **5** |
| **1** | **3** | **10** | **3** | **0** | **65** | **31** | **353** | **248** | **0** | **5** |

Transforming categorical data into numeric data for processing the data further.

## **Skewness :**

Skewness is a measure of the asymmetry of the probability distribution of a random variable about its mean. In other words, skewness tells you the amount and direction of skew (departure from horizontal symmetry). The skewness value can be positive or negative, or even undefined.

|  |
| --- |
| Airline 0.483882  Date\_of\_Journey -0.044181  Source -0.424997  Destination 1.263630  Route -0.281807  Dep\_Time 0.178928  Arrival\_Time -0.523661  Duration -0.275404  Total\_Stops 0.672623  Additional\_Info -1.719119  dtype: float64 |

Keeping +/-(0.5) as the range of skewness. There are 3 columns out of the range of skewness.

## Loading Train Model :

|  |
| --- |
| fitted\_model=pickle.load(open('FPP\_train\_project.pkl','rb')) |

|  |
| --- |
| predictions=fitted\_model.predict(df.tnew) |

**Predicted Flight ticket prices**

|  |
| --- |
| array([ 4652.68333333, 9106.45 , 13641.0725 , ...,  7354.92 , 9612.68 , 12045.0425 ]) |

## Concluding Remarks :

In this type of problem Feature Engineering is the most crucial think . We can see how we have handled the categorical and numerical data and also how we build different ML model on the same dataset . We also check the r2 score of the model so that we can understand how it should perform in our test dataset . At last we can also further improve the Model by Tunning different parameters which are being used in the model .