Machine Learning to Understand & Predict Price of Air ticket/Flight

The Airline Companies is considered as one of the most enlightened industries using complex methods, strategies to allocate airline prices in a dynamic fashion. These industries are trying to keep their all-inclusive revenue as high as possible and boost their profit. Customers are seeking to get the minimum price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit. However, discrepancy between available seats and passenger demand usually tends to either the customer pay high value of money or airline company may loss.

Airlines companies are generally well-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

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

So, this blog involves collection of data for flight fares with other features and building a model to predict fares of flights.

A report says India's affable aeronautics industry is on a high development movement. *India is the third-biggest avionics showcase in 2020 and the biggest by 2030.* Indian air traffic is normal to cross the quantity of 100 million traveler's by 2017, whereas there were just 81 million passengers in

2015. Agreeing to Google, the expression" Cheap Air Tickets" is most sought in India. At the point when the white-collar class of India is presented to air travel, buyers searching at modest costs.

Any person who has booked a flight ticket previously knows how dynamically costs change. Aircraft uses advanced strategies called Revenue Management to execute a distinctive valuing strategy. The least expensive accessible ticket changes over a period the cost of a ticket might be high or low. This valuing method naturally modifies the toll as per the time like morning, afternoon or night. Cost may like change with the seasons like winter, summer and celebration seasons. The extreme goal of the carrier is to build its income yet on the opposite side purchaser is searching at the least expensive cost. Purchasers generally endeavor to purchase the ticket in advance to the take-off day.

From the customer point of view, determining the minimum price or the best time to buy a ticket is the key issue. The conception of "tickets bought in advance are cheaper" is no longer working (William Groves and Maria Gini, 2013). It is possible that customers who bought a ticket earlier pay more than those who bought the same ticket later. Moreover, early purchasing implies a risk of commitment to a specific schedule that may need to be changed usually for a fee. Most of the studies performed on the customer side focus on the problem of predicting optimal ticket purchase time using statistical methods.

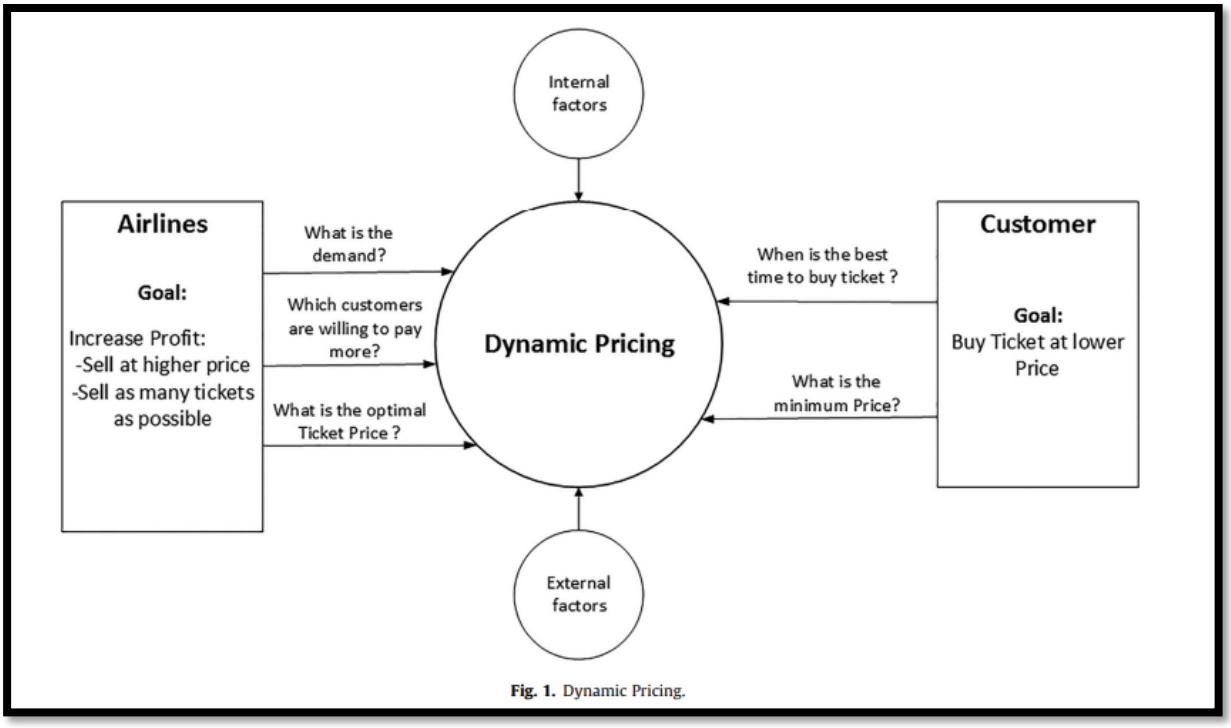


Fig. 1. Dynamic Pricing.

As noted by Y. Chen et al. (2015), *predicting the actual ticket price is a more difficult task than predicting an optimal ticket purchase time* due to various reasons: absence of enough datasets, external factors influencing ticket prices, dynamic behavior of air-ticket pricing, competition among airline companies, proprietary nature of airlines ticket pricing policies etc.

In Early days, prediction of the demand along a given route could help an airline company pre-plan the flights and determine appropriate pricing for the route. Existing demand prediction models generally try to predict passenger demand for a one-way flight/route and market share of an individual airline. Price discrimination allows an airline company to categorize customers based on their ability to pay and thus charge them different prices. Customers could be categorized into different groups based on various criteria such as business vs leisure, tourist vs normal traveler, profession etc. For example, business customers are willing to pay more as compared to leisure customers as they rather focus on quality service than price.

Flight Price Prediction Dataset

In this case study we are using the two types of datasets one is train dataset and another is test dataset. This fictional dataset available to download from GitHub and Kaggle. You can also download dataset from my GitHub profile here. This dataset consists of 10683 rows, 11 features describing each employee's background and characteristics and target variable. Price is our target variable to predict. As our target variable is continuous in nature, this case study falls into Regression Machine Learning Problem.

We have two objectives here:

1. Which are Key factors affecting Price.
2. Building Machine Learning Model for Predicting Price.

The Features considered initially for each flight are:

1. Air lines: The name of the airline.
2. Date of Journey: The date of the journey.
3. Source: The source from which the service begins
4. Destination: The Destination where the service ends.
5. Route: The route taken by the flight to reach the destination.
6. Dep Time: The time when the journey starts from the source.
7. Arrival time: Time of arrival at the destination.
8. Duration: Total duration of the flight.
9. Total Stop: Total stop between the source of destination.
10. Additional Info.: Information about the flight.
11. Price: The price of the Ticket.

Data Preparation: Load, Clean and Format

Let's begin with importing libraries for EDA and dataset itself.

```
import pandas as pd # for data wrangling purpose
import numpy as np # Basic computation library
import seaborn as sns # For Visualization
import matplotlib.pyplot as plt # ploting package
%matplotlib inline
import warnings # Filtering warnings
warnings.filterwarnings('ignore')
```

```
# Importing Flight Prediction dataset Excel file using pandas
df = pd.read_excel('Data_Train.xlsx',parse_dates=['Date_of_Journey'])
```

```
print('No of Rows:',df.shape[0])
print('No of Columns:',df.shape[1])
pd.set_option('display.max_columns', None) # This will enable us to see truncated columns
df.head()
```

```
No of Rows: 10683
No of Columns: 11
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	2019-09-06	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	2019-12-05	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	2019-01-03	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

Checking different datatypes in dataset: -

```
# Sort columns by datatypes
df.columns.to_series().groupby(df.dtypes).groups
```

```
{datetime64[ns]: ['Date_of_Journey'], int64: ['Price'], object: ['Airline', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arri
val_Time', 'Duration', 'Total_Stops', 'Additional_Info']}
```

Here we have 9 features with object datatypes and rest of one is Numeric feature with int64, and one has datetime64 its number series.

Feature Engineering:

In this step we mainly work on the data set and do some transformation like creating different bins of particular columns, clean the messy data so that it can be used in our ML model. This step is important because for a high prediction score you need to continuously make changes in it.

```
# Converting Dep_Time and Arrival_Time to datetime format
df['Dep_Time'] = pd.to_datetime(df['Dep_Time'])
df['Arrival_Time']=pd.to_datetime(df['Arrival_Time'])
```

Feature Extraction : We will create few new features for model building

1. Feature Engineering on 'Dep_Time' Column

```
# Extracting Hours from Dep_Time column
df['Dep_Hour']=pd.to_datetime(df['Dep_Time']).dt.hour

# Extracting Minutes from Dep_Time column
df['Dep_Min']=pd.to_datetime(df['Dep_Time']).dt.minute

# Dropping Dep_Time column
df.drop("Dep_Time",axis=1,inplace=True)
```

2. Feature Engineering on 'Arrival_Time' Column

```
# Extracting Arrival_Hour from Arrival_Time column
df['Arrival_Hour']=pd.to_datetime(df['Arrival_Time']).dt.hour

# Extracting Arrival_Min from Arrival_Time column
df['Arrival_Min']=pd.to_datetime(df['Arrival_Time']).dt.minute

# Dropping Arruval_Time column
df.drop("Arrival_Time",axis=1,inplace=True)
```

3. Feature Engineering on Date of Journey Column

- **We are gone extract day, month and year column from feature Date of Journey**

```
# Extracting Day from Date_of_journey column
df['Journey_Day'] = pd.to_datetime(df.Date_of_Journey,format="%d/%m/%Y").dt.day

# Extracting Month from Date_of_journey column
df['Journey_Month'] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.month

# Extracting Year from Date_of_journey column
df['Journey_Year'] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.year
```

- **Extracting weekday name.**

```
# Extracting WeekDay from Date_of_journey column
df['Week_Day'] = pd.to_datetime(df.Date_of_Journey,format="%d/%m/%Y").dt.day_name()

# Dropping Date_of_journey column
df.drop("Date_of_Journey",axis=1,inplace=True)
```

4. Feature Engineering on 'Duration' Column to extract flight duration in Minute

```
# Conversion of Duration column from hr & Minutes format to Minutes
df['Duration']=df['Duration'].str.replace('h','*60').str.replace(' ','+').str.replace('m','*1').apply(eval)

# convert this column into a numeric datatypes
df['Duration']=pd.to_numeric(df['Duration'])
```

Here we have created feature engineering and create you features which help us to get more information about the factors which are affecting the price of Flight.

Data Integrity Check:

Dataset can have missing values, duplicated entries and whitespaces. Now we will perform this integrity check of dataset.

1. Duplicate data

- **Since dataset is large, Let check for any entry which is repeated or duplicated in dataset.**

```
df.duplicated().sum()  # This will check the duplicate data for all columns.
```

```
222
```

Around 222 duplicate data rows. There is no point on training model on duplicated data, so we gone drop them.

```
df.drop_duplicates(keep='last',inplace= True)
```

```
df.shape
```

```
(10461, 16)
```

- **Let check if any whitespace, 'NA' or '-' exist in dataset.**

```
df.isin([' ','NA','-','?']).sum().any()
```

```
False
```

Statistical parameters like mean, median, quantile can give important details about database. Now is time to look at statistical Matrix of Dataset.

Here are few key Observation of Statistical Matrix:

- The Minimum flight ticket price is Rs.1759 and Maximum flight ticket price is Rs.79512. Outliers are present.
- The (mean > median) for Price, Journey_Day, Duration and Dep_Hour, which mean right skew data.
- The (mean < median) for Dep_Min, Arrival_Hour and Arrival_Min which mean left skew data.
- Presence of outliers in Duration & Price columns as we see huge difference in 75% and max.
- High value of std. deviation indicating spread of data.
- There are 12 airlines in dataset with maximum flight run by Jet Airways.
- The data of 128 route in dataset and we find maximum flight on route DEL → BOM → COK.

Exploratory data analysis

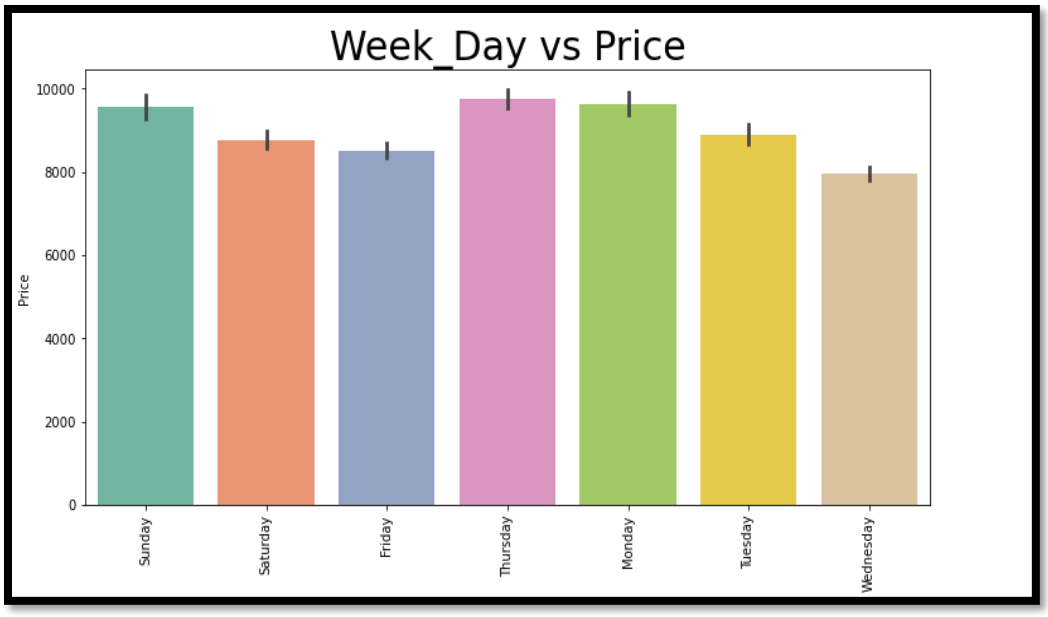
EXPLORATORY DATA ANALYSIS STATES TO THE CRITICAL PROCESS OF PERFORMING INITIAL INVESTIGATIONS ON DATA SO AS TO DISCOVER PATTERNS, TO SPOT ANOMALIES, TO TEST HYPOTHESIS AND TO CHECK ASSUMPTIONS WITH THE HELP OF SUMMARY STATISTICS AND GRAPHICAL REPRESENTATIONS.

Let's begin data exploration of Target variable Price:

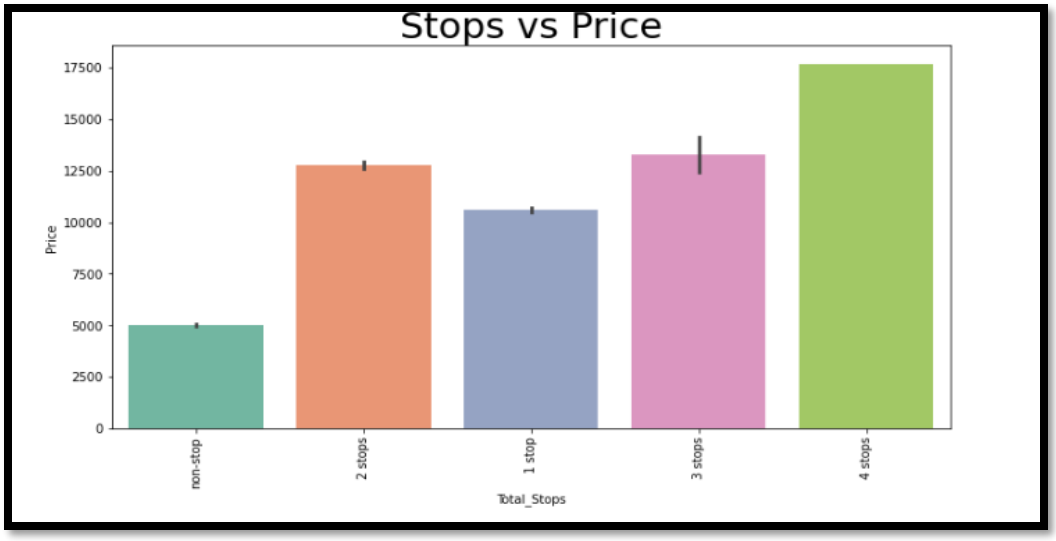
1. Exploring Relationship between feature 'Journey Month' and 'Price'

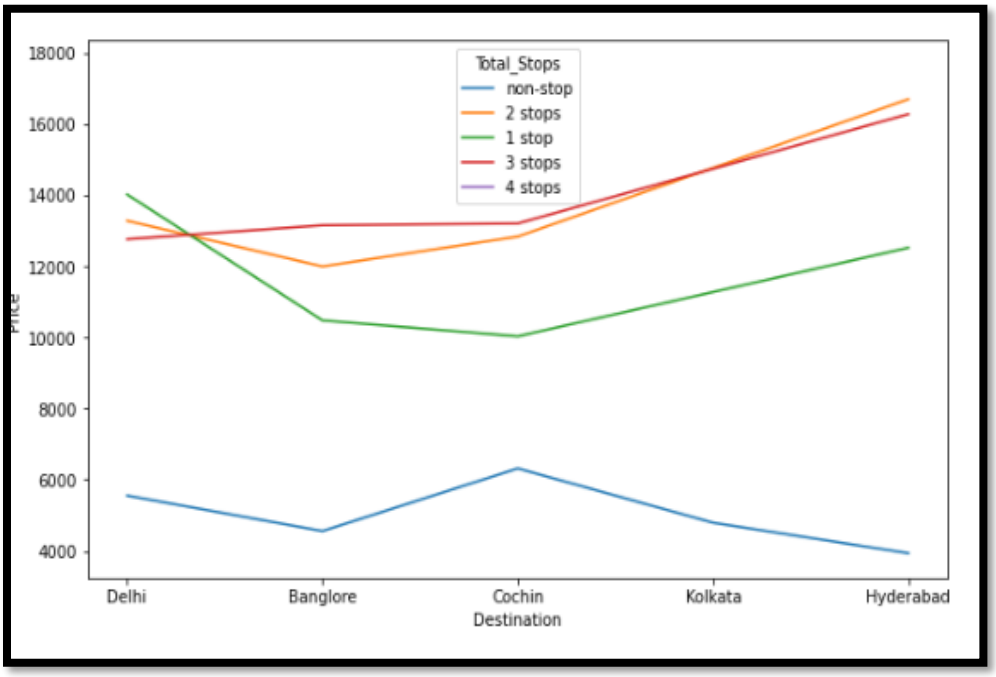
Here in the below graph, we get clear insights that in the month of January having highest fight ticket and very lowest flight tickets/fare price in the month of April.

2. Exploring Relationship between feature 'Week_Days' and 'Price'

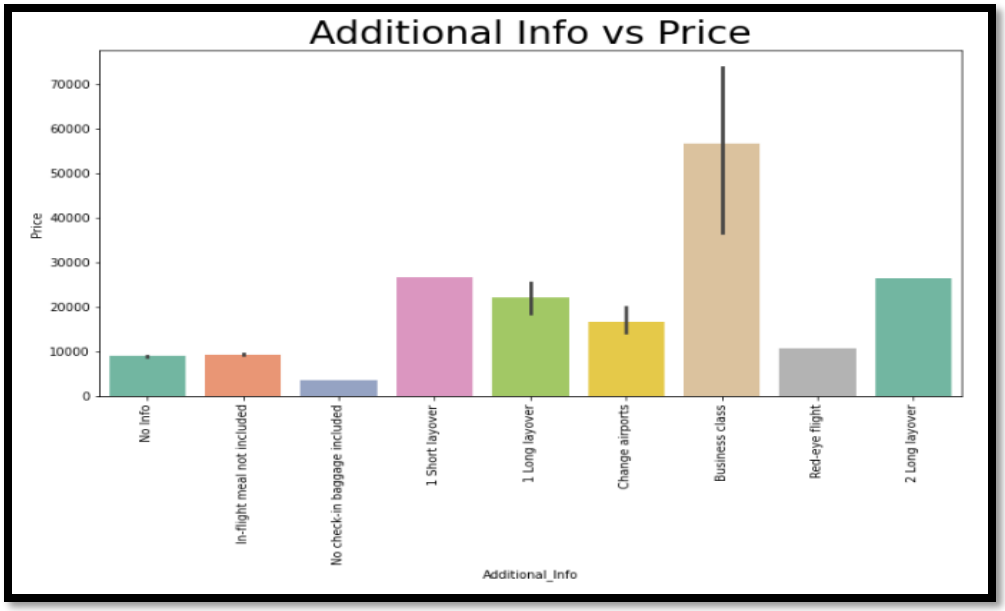


Here we have highest fare of Fight Tickets on Thursday, followed by Monday, Sunday and Tuesday respectively. Almost Rs. 9500.00 fare of air ticket had on thrusday and Rs 8000.00 fare on Wednesday.



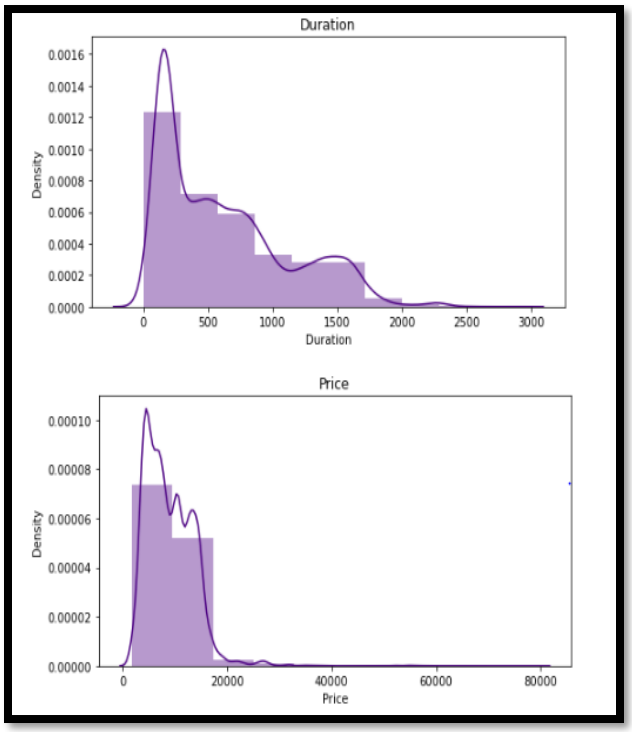


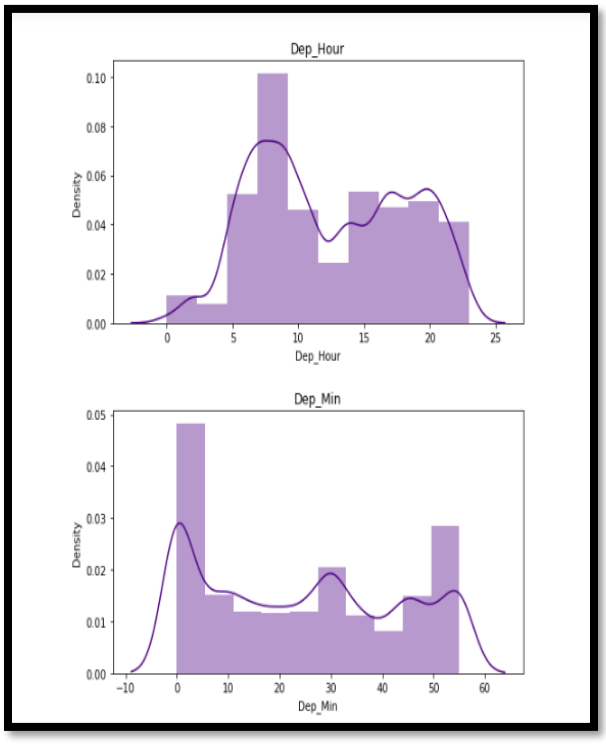
For non-stop flights price is less around Rs. 5000.0, for 4 stops flight the price is very high which is almost Rs.17500.00.

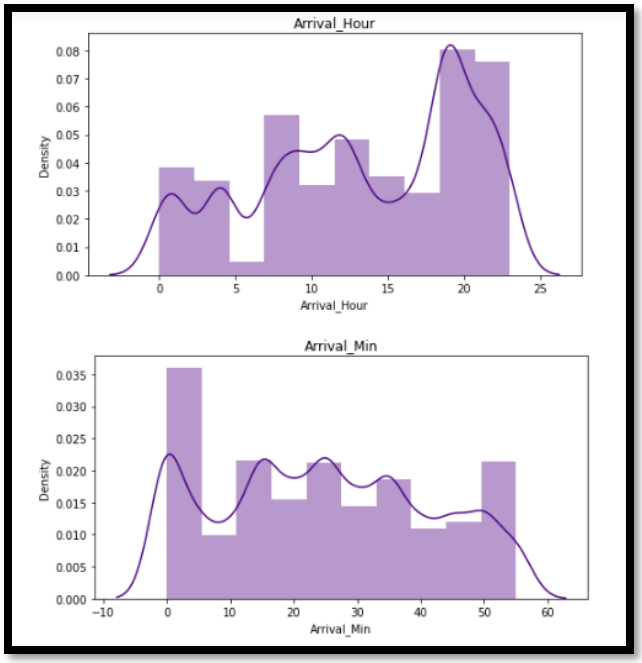


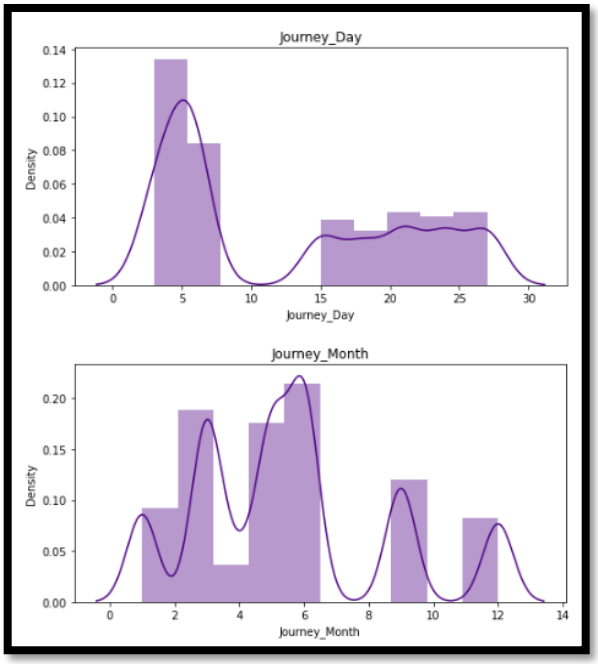
For Business class air ticket rates was nearabout Rs.60,000.00 and For ticket in which baggage checking was not included having almost Rs.5000.00 fare.

Let's Check Distribution plot for all numerical features to get more clear idea about data.



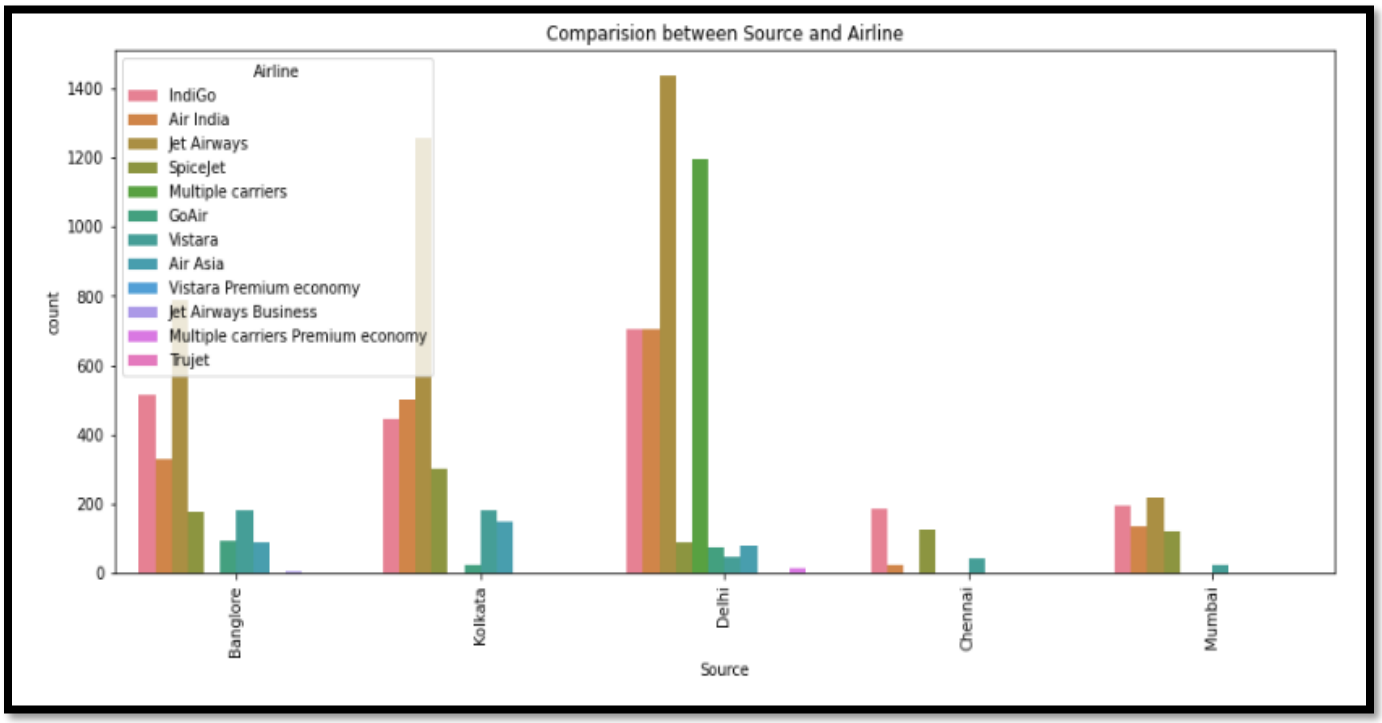






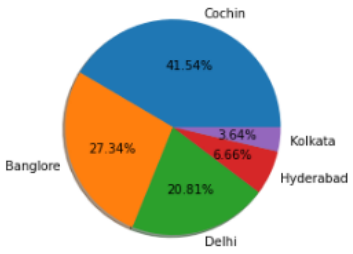
Here we get clear insights from all distribution plots as skewness ins present almost in all feature of numerical data.

Exploring Relationship between feature 'Source' and 'Airline'



In all sources Jet Airways is quite famous except Chennai only.

```
#Ploting pie chart for Destination column
labels='Cochin','Banglore','Delhi','Hyderabad','Kolkata'
fig, ax = plt.subplots()
ax.pie(df['Destination'].value_counts(), labels=labels, autopct='%1.2f%%', shadow=True)
plt.show()
```



Feature Engineering: Data Pre-processing

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

Feature Engineering is very important step in building Machine Learning model. Some machine learning projects succeed and some fail. What makes the difference? Easily the most important factor is the features used. In Feature engineering can be done for various reason. Some of them are mention below:

1. Feature Importance: An estimate of the usefulness of a feature
2. Feature Extraction: The automatic construction of new features from raw data (Dimensionality reduction Technique like PCA)
3. Feature Selection: From many features to a few which are useful
4. Feature Construction: The manual construction of new features from raw data (For example, construction of new column for month out date - mm/dd/yy)

There are Varity of techniques use to achieve above mention means as per need of dataset. Some of Techniques important are as below:

- Handling missing values
- Handling imbalanced data using SMOTE
- Outliers' detection and removal using Z-score, IQR
- Scaling of data using Standard Scalar or Minmax Scalar
- Binning whenever needed

- Encoding categorical data using one hot encoding, label / ordinal encoding
- Skewness correction using Boxcox or yeo-Johnson method
- Handling Multicollinearity among feature using variance inflation factor
- Feature selection Techniques:
  - ✓ Correlation Matrix with Heatmap
  - ✓ Univariate Selection – SelectKBest
  - ✓ ExtraTreesRegressor method

In this case study we will use some of the mention feature engineering Techniques one by one.

1. Dropping unnecessary features
   Features like Date_of_Journey, Arrival_Time,Dep_Time, which are unnecessary features as we have create 5 more new features from these, so we decided to drop it before proceeding further investigation.

```
# Dropping Dep_Time column
df.drop("Dep_Time",axis=1,inplace=True)
```

```
# Dropping Arruval_Time column
df.drop("Arrival_Time",axis=1,inplace=True)
```

```
# Dropping Date_of_journey column
df.drop("Date_of_Journey",axis=1,inplace=True)
```

2. Encoding Categorical & Ordinal Features

Label Encoding is deployed on Categorical data features to convert data into Numeric features.

```
Categorical_Features = ['Airline', 'Source', 'Destination', 'Route', 'Total_Stops', 'Additional_Info', 'Week_Day']
```

```
# Using Label Encoder on categorical variable
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for i in Categorical_Features:
    df[i] = le.fit_transform(df[i])
df.head()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Dep_Hour	Dep_Min	Arrival_Hour	Arrival_Min	Journey_Day	Journey_Mor
0	3	0	2	18	170	4	3	3897	22	20	1	10	24	
1	1	3	0	81	445	1	3	7662	5	50	13	15	5	
2	4	2	1	115	1140	1	3	13882	9	25	4	25	6	
3	3	3	0	88	325	0	3	6218	18	5	23	30	5	
4	3	0	2	29	285	0	3	13302	16	50	21	35	3	

Since now encoding is done we will move towards outliers' detection and removal.

1. Outliers' detection and removal

> Identifying outliers and bad data in your dataset is probably one of the most difficult parts of data clean-up, and it takes time to get right. Even if you have a deep understanding of statistics and how outliers might affect your data, it's always a topic to explore cautiously.
>
> - Page 167, Data Wrangling with Python, 2016

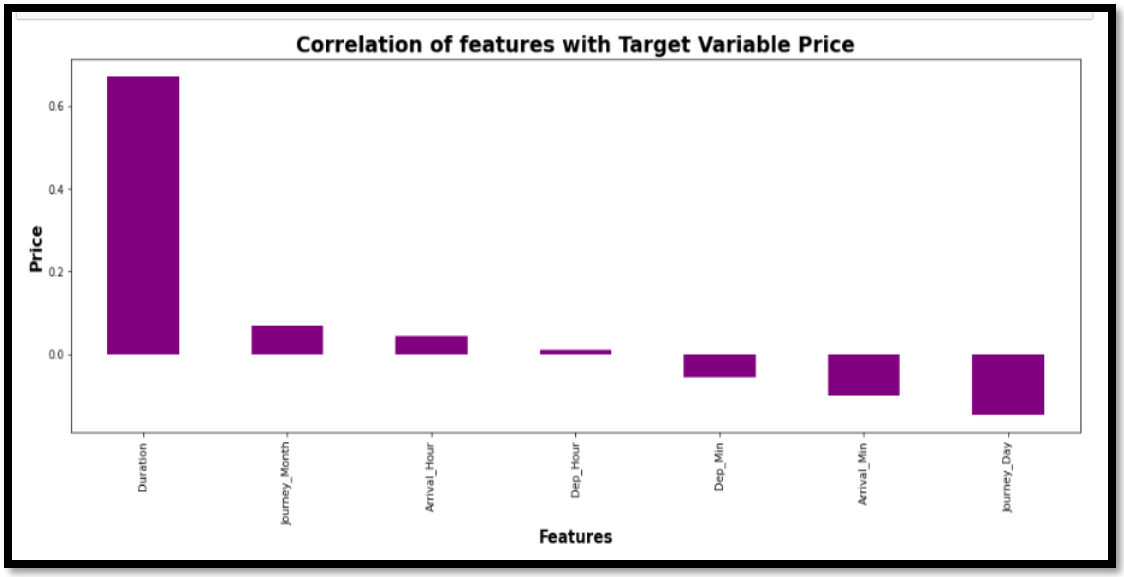
Machine learning algorithms are so, sensitive to range and distribution of attribute/feature values. Data outliers can spoil and mislead the training process which outcome are like, longer training times, less accurate models and ultimately poorer results. Outliers can be seen in boxplot of numerical feature. We did not added boxplot here as it will make this article length, I left it to reader to further investigate. Now we will use Z-score method for outliers' detection.

```
from scipy.stats import zscore
z = np.abs(zscore(df[Numerical_Features]))
threshold = 3
df1 = df[(z<3).all(axis = 1)]

print ("Shape of the dataframe before removing outliers: ", df.shape)
print ("Shape of the dataframe after removing outliers: ", df1.shape)
print ("Percentage of data loss post outlier removal: ", (df.shape[0]-df1.shape[0])/df.shape[0]*100)

df=df1.copy() # reassigning the changed dataframe name to our original dataframe name
```

```
Shape of the dataframe before removing outliers:  (10461, 15)
Shape of the dataframe after removing outliers:  (10305, 15)
Percentage of data loss post outlier removal:  1.4912532262689993
```

4. Correlation Heatmap

Correlation Heatmap shows at a glance which variables /features/ columns are highly positively correlated or highly negatively correlated , to what degree, in which direction, and alerts us to potential multicollinearity problems. The bar plot of correlation coefficient of target variable with independent features shown below



5. Scaling of data using Standard Scalar:

5. Standard Scaling

```
# Splitting data in target and dependent feature
X = df.drop(['Price'], axis =1)
Y = df['Price']
```

```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()
X_scale = scaler.fit_transform(X)
```

6. Multicollinearity between features

Variance Inflation factor imported from statsmodels. stats. Outliers__influence to check multicollinearity between features.

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif = pd.DataFrame()
vif["VIF values"] = [variance_inflation_factor(X_scale,i) for i in range(len(X.columns))]
vif["Features"] = X.columns
vif
```

	VIF values	Features
0	1.034072	Airline
1	1.428426	Source
2	1.691504	Destination
3	1.532537	Route
4	1.737057	Duration
5	2.391514	Total_Stops
6	1.121303	Additional_Info
7	1.033779	Dep_Hour
8	1.021015	Dep_Min
9	1.036344	Arrival_Hour
10	1.095872	Arrival_Min
11	1.107740	Journey_Day
12	1.099970	Journey_Month
13	1.012217	Week_Day

We can see that for all features Variance inflation factor in within permissible limit of 10. Multicollinearity not creating any threat here to our model building.

Machine Learning Model Building:

In this section we will build Supervised learning ML model-based classification algorithm. As objective is to predict attrition in 'Yes' or 'No' leads to fall problem in domain of classification algorithm. train_test_split used to split data with size of 0.33

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state=99, test_size=0.25)
print('Training Feature Matrix Size:', X_train.shape)
print('Training Target Vector Size :', Y_train.shape)
print('Test Feature Matrix Size:', X_test.shape)
print('Test Target Vector Size:', Y_test.shape)
```

```
Training Feature Matrix Size: (7213, 14)
Training Target Vector Size : (7213,)
Test Feature Matrix Size: (3092, 14)
Test Target Vector Size: (3092,)
```

First we will build base model using logistic regression algorithim. Best random state is investigated using for loop for random state in range of (50,500).

```
maxR2_score=0
maxRS=0
for i in range(50,500):
    X_train, X_test, Y_train, Y_test = train_test_split(X_scale, Y, random_state=i, test_size=.25)
    lin_reg=LinearRegression()
    lin_reg.fit(X_train,Y_train)
    y_pred=lin_reg.predict(X_test)
    R2=r2_score(Y_test,y_pred)
    if R2>maxR2_score:
        maxR2_score=R2
        maxRS=i
print('Best R2 Score is', maxR2_score ,'on Random_state', maxRS)
```

```
Best R2 Score is 0.496033105453895 on Random_state 492
```

Logistics regression model is train with random state 492. The evalution matrix along with classification report is as below :

Linear Regression

```
X_train, X_test, Y_train, Y_test = train_test_split(X_scale, Y, random_state= 492, test_size=0.25)
lin_reg= LinearRegression()
lin_reg.fit(X_train, Y_train)
y_pred = lin_reg.predict(X_test)
print('\033[1m'+ 'Error :'+ '\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test, y_pred))
print('Root Mean squared error :', np.sqrt(mean_squared_error(Y_test, y_pred)))
print('\033[1m'+' R2 Score :'+'\033[0m')
print(r2_score(Y_test,y_pred)*100)
```

```
Error :
Mean absolute error : 2242.4408662024616
Mean squared error : 8037441.4106131075
Root Mean squared error : 2835.0381673997103
 R2 Score :
49.6033105453895
```

As of now our base model is ready and we have R2 score of 49.60% and RMSE(Root Mean Sqaured error) 2835.0381.

We will train model with different Regression algorithm along with k-5 fold cross validation. The final evaluation matrix different classification algorithm is as shown table below:

ML Algorithm	R2 Score	CV Mean Score
Logistics Regression	49.60	0.46
Random Forest regressor	92.32	0.92
Decision Tree Regressor	87.16	0.86
Extra Tree regressor	92.45	0.92
XGBoost Regressor	93.16	0.93

(Min Value in column -Green, Max Value in column - Pink Colour)

We can see that XGBoost regressor giving us highest R2 score and CV mean score., so now we will perform the hyper parameter tunning on XGBoost Regressor algorithm to find the best parameter of this model.

Hyper Parameter Tuning : GridSearchCV

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

```
print(xgb.get_params())
```

```
{'objective': 'reg:squarederror', 'base_score': 0.5, 'booster': 'gbtree', 'colsample_bylevel': 1, 'colsample_bynode': 1, 'colsa
mple_bytree': 1, 'enable_categorical': False, 'gamma': 0, 'gpu_id': -1, 'importance_type': None, 'interaction_constraints': '',
'learning_rate': 0.300000012, 'max_delta_step': 0, 'max_depth': 6, 'min_child_weight': 1, 'missing': nan, 'monotone_constraint
s': '()', 'n_estimators': 100, 'n_jobs': 4, 'num_parallel_tree': 1, 'predictor': 'auto', 'random_state': 0, 'reg_alpha': 0, 're
g_lambda': 1, 'scale_pos_weight': 1, 'subsample': 1, 'tree_method': 'exact', 'validate_parameters': 1, 'verbosity': None}
```

```
parameter = {'n_estimators':[50,100,125],'gamma':np.arange(0,0.2,0.1),
             'booster' : ['gbtree','dart','gblinear'], 'max_depth':[4,6,8,10],
             'learning_rate' : [0.01, 0.1,0.3,0.5,1] }
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X_scale, Y, random_state= 492, test_size=0.25)
```

```
GCV = GridSearchCV(XGBRegressor(),parameter,verbose =10)
```

```
GCV.fit(X_train,Y_train)
```

Next step is to build final machine learning model over best params in Hyper parameter tuning.

Final Regression Model

```
Final_mod =  XGBRegressor(n_estimators=125 , booster= 'gbtree', learning_rate= 0.1, max_depth= 8, gamma= 0.0 )
Final_mod.fit(X_train,Y_train)
y_pred=Final_mod.predict(X_test)
print('\n')
print('\033[1m'+' Error in Final Model :' +'\033[0m')
print('Mean absolute error :', mean_absolute_error(Y_test,y_pred))
print('Mean squared error :', mean_squared_error(Y_test,y_pred))
print('Root Mean Squared Error:', np.sqrt(mean_squared_error(Y_test,y_pred)))
print('\n')
print('\033[1m'+' R2 Score of Final Model :'+'\033[0m')
print(r2_score(Y_test,y_pred))
print('\n')
```

```
 Error in Final Model :
Mean absolute error : 633.6503443927047
Mean squared error : 1033499.1686449125
Root Mean Squared Error: 1016.6116115040751


 R2 Score of Final Model :
0.9351971180965378
```

For the final model after implementing hyper Para tunning our R2 score is sightly increase to 93.51%.

At last, we will save final model with joblib library, so it can be deployed on cloud platform.

Final Regression For Train Dataset Saving Model ¶

```
: import joblib
  joblib.dump(Final_mod,'Flight_Price_Predication_Final.pkl')
: ['Flight_Price_Predication_Final.pkl']
```

Concluding Remarks on EDA and ML Model

- The standard deviation in the “Route” , “Deep Time” , “Arrival Time” , “Duration” and “Price” column is two high which means that the values in this columns are largely scattered and are not near to the mean value .
- The standard deviation of other columns is to high which shows us a normal distribution of data and less chances of having skewness.
- The value in target variable (“Price”)has its minimum price at 1759 and maximum price at 79512.The range is to high.
- The most negatively correlated column is that of the “Total Stops” which means more the number of stops less the flight price.
- The most positively correlated variable is “Route”.
- The variables “Route”, “Arrival Time”, “Source”, “Month”, “Deep Time” and positively correlated with the target variable and variables “Airline” , “Destination”, “Duration” , “Day” and “Total Stops” are negatively correlated.
- So, we build the Machine Learning Algorithm for the same, afterwards we have done hyper paratunning on our best model which have the R2 score of 93% and Mean cv score of 92% and save the final model using the parameter of hyper paratunning.
- Save final model in csv format as we need to predict the prices and in further steps we need to run the our prediction using the test data set.
- here for test data we have followed same steps which we have done for train dataset.
- predicted the flight prices on test dataset.

You can get code of this case study from my GitHub Profile.