



## Flight Price Prediction Project

Submitted by:

Khushboo Khatri

## **ACKNOWLEDGMENT**

I would like to appreciate FlipRobo to provide us with the opportunity to work on this Project. I would like to extend my gratitude towards our SME Shwetank Mishra, to help me out with his expertise.

I would like to thank Datatrained Mentors, for giving us the knowledge required to complete this project.

# INTRODUCTION

- **Business Problem Framing**

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).

So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

- **Conceptual Background of the Domain Problem.**

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

## Analytical Problem Framing

- **Mathematical/ Analytical Modeling of the Problem**

As we collect the data from site yatra.com, the dataset consists of mostly categorical columns, i.e., having Object datatypes.

So, we need to perform some sort of feature engineering, so that we did not loss any meaningful information.

And the first descriptive analysis/statistical analysis we did, includes the object data in which we get the count, unique values max min and other useful data.

- **Data Sources and their formats**

We scrape our data from yatra.com, in total we have 2787 rows and 10 columns.

Following are the columns and their respective data types.

Unnamed: 0	int64
AirLines	object
Source	object
Destination	object
Total_Stops	object
Departure_Time	object
Arrival_Time	object
Total_Duration	object
Date_Of_Journey	object
Flight_Price	object

We scrape our data from yatra.com, from there we scrape, Airlines detail, Source, Destination, Total number of Stops in flight, what is the departure time, what is the arrival time, date of journey, total duration and off course the Flight Price.

Here Flight Price is our label, while others are features.

- Data Preprocessing Done

Data preprocessing is a process of preparing raw data and making it suitable for machine learning model.

As we can see that almost all of our data is in the form of Object type, we did some feature engineering, some data cleaning, and encoding in order to prepare it for Machine learning model building.

We did data cleaning by dropping Unnamed: 0 column, and also by removing null values.

We feature engineered Price and make it numerical type data without losing any valuable information.

We also perform Feature engineering Total Duration, arrival time, departure time and Date of journey and extract meaningful information out of it.

We did mapping on Total stops columns to make it numerical column and also withhold the exact information.

We did some visualization, perform EDA both univariate and multivariate analysis.

Through Univariate analysis we are able to see the distribution of data and multivariate analysis helps us to observe the relationship among various features. And come up with these columns.

```
df.dtypes
AirLines      object
Source        object
Destination    object
Total_Stops    int64
Day           object
Dep_hour      int32
Dep_min       int32
Arrival_hour  int32
Duration_hours int64
Duration_mins int64
Price         int32
Date         int32
Month        int64
Arrival_Min  int32
dtype: object
```

Encoding is also most important part of preprocessing, as it encode categorical data into numerical columns.

In this dataset, we encode the three nominal column(Airlines, source and Destination) using Label Encoder.

For Day column we use mapping again, to map 7 days of the week as per my preference.

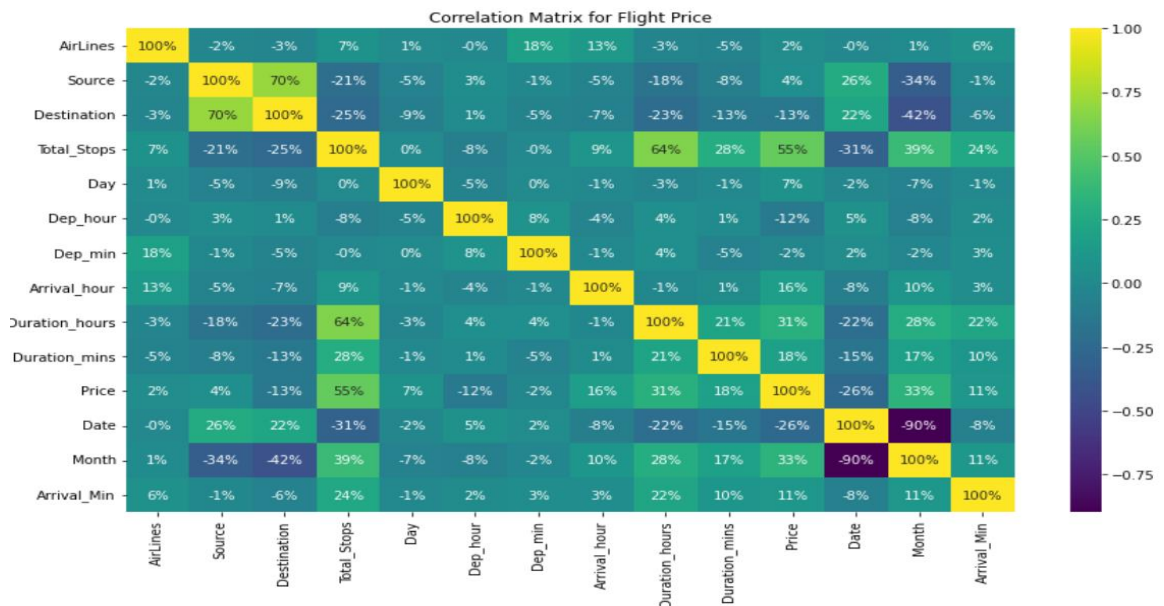
- **Data Inputs- Logic- Output Relationships**

From encoding, all our features(input data)/ independent variables are of numerical data type.

Our label/ Target column is Price, which is also numerical datatype. Since our entire dataset is in numerical form, we can say that its almost ready for model building.

However before proceeding in that direction, we need to scale our features.

And to understand the relation between Input variables i.e. features and Output i.e. Price, we plot correlation matrix.



From here we can observe that Total Stops is highly correlated with our label, Price. We can also see the relation among various independent features.

- **Hardware and Software Requirements and Tools Used**

As we know that this project is divided into two parts, first is data collection and the other is model building.

For Data Collection purpose, we scrape our data from yatra.com. And to do web scraping we use Selenium. Following are the set of libraries we import to perform the web scarping of yatra.com webpage.

```
# importing necessary libraries
import pandas as pd
import numpy as np
import selenium
from selenium import webdriver
import warnings
warnings.filterwarnings('ignore')
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
from selenium.webdriver.common.by import By

import time
```

And for Model Building these are the basic Libraries we starts with, and later we import other libraries as per our requirement.

```
» # importing important Libraries

import pandas as pd
import numpy as np

# visualization library
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# warning
import warnings
warnings.filterwarnings('ignore')


# Preprocessing and Normalization
from sklearn.preprocessing import StandardScaler

# Models
from sklearn.model_selection import train_test_split, GridSearchCV
```

As ours is a Regression Problem, we further import LinearRegression, RandomForestRegressor, Adaboost Regressor, Lasso, XGBoost regressor for Model Building. And also, randomizedSearchCV inorder to hyper tune our model.

## Model/s Development and Evaluation






- Identification of possible problem-solving approaches (methods)
  - ✚ **For Data Collection:** We scrape from yatra.com website.
    - We starts with getting the web driver
    - Getting the URL
    - Finding the web elements for the details we need to scrape
    - Scraping the details and store them in list
    - Forming a dataframe using those list
    - Saving the DataFrame in Excel form.

 **For Model Building:** we start with importing important libraries, followed by getting the dataset. Then describing the dataset, which followed by following steps.

- **Data Cleaning**
- **Feature Engineering And EDA**
- **Encoding**
- **Skewness and Outlier Detection and removal**
- **Scaling the feature columns**
- **Splitting the dataset into Test and Train dataset**
- **Building various Model**
- **We use  $r^2$  score as our primary evaluation metrics.**
- **And MAE(mean absolute error) and RMSE(root mean squared error) as our secondary model evaluation metrics.**
- **Lastly we Summarize and save the best model.**

- **Testing of Identified Approaches (Algorithms)**

In Flight Price data set, as it is Regression Problem, following are the Algorithms/Model we Build:

-  Linear Regression
-  RandomForest Regressor
-  Lasso
-  AdaBoost Regression
-  XGBOOST Regressor

We did cross validate all the models, and also perform hyper parameter tuning for most of them.

- **Run and Evaluate selected models:**

From below summary, we can see that XGBOOST Regressor is the best suitable model for our dataset.



## Summarizing each Model.

```
# Summarizing each model

MAE=[LR_MAE,RFR_MAE,LS_MAE,XGB_MAE,ADA_MAE]
R2=[LR_R2,RFR_R2,LS_R2,XGB_R2,ADA_R2]
RMSE=[LR_RMSE,RFR_RMSE,LS_RMSE,XGB_RMSE,ADA_RMSE]
Cross_score=[LR_CS,RFR_CS,LS_CS,XGB_CS,ADA_CS]

Model= pd.DataFrame({
    'Models':['Linear Regression', 'Random Forest Regressor', 'Lasso Regressor', 'XGBoost Regressor', 'AdaBoost Regressor'],
    'MAE': MAE, 'R^2': R2, 'RMSE': RMSE, 'Cross Validation Score': Cross_score })

Model.sort_values(by = 'R^2', ascending = False )

91]:
```

	Models	MAE	R^2	RMSE	Cross Validation Score
3	XGBoost Regressor	0.043011	0.8189	0.063871	0.783089
1	Random Forest Regressor	0.046946	0.7512	0.074878	0.712839
2	Lasso Regressor	0.084175	0.4308	0.113246	0.369833
0	Linear Regression	0.083572	0.4106	0.001494	0.376193
4	AdaBoost Regressor	0.102254	0.3532	0.120725	0.341451

Here is how we build the XGBOOST model, cross validate and perform hyperparameter tuning, in order to improve its efficiency.

## XGBOOST Regressor

```
import xgboost as XGB
#model
XGB = XGB.XGBRegressor()

#fit
XGB.fit(X_train,y_train)

#predict
y_pred=XGB.predict(X_test)
pred=XGB.predict(X_train)

print("----- Train score-----")
XGB_train_R2= round(r2_score(y_train, pred), 4)
XGB_train_MAE=(mean_absolute_error(y_train,pred))/(max(y)-min(y))
XGB_train_RMSE=(np.sqrt(mean_squared_error(y_train,pred)))/(max(y)-min(y))

print(f" R^2 Score : {XGB_train_R2}\n")
print(f" MAE avg score : {XGB_train_MAE}\n")
print(f" RMSE avg score : {XGB_train_RMSE}\n")

#score variables
XGB_MAE= (mean_absolute_error(y_test, y_pred))/(max(y)-min(y)) # we are calculating avg MAE
XGB_R2= round(r2_score(y_test, y_pred), 4)
XGB_RMSE=(np.sqrt(mean_squared_error(y_test,y_pred)))/(max(y)-min(y))
# calculating avg RMSE
print("\n-----Test Score-----\n")
print(f" R^2 Score : {XGB_R2}\n")
print(f" Mean Absolute Error avg : {XGB_MAE}\n")
print(f" Root Mean Squared Error avg: {XGB_RMSE}\n")
```

----- Train score-----

R^2 Score : 0.9852

MAE avg score : 0.012593469855886429

RMSE avg score : 0.018385788140429106

-----Test Score-----

R^2 Score : 0.8189

Mean Absolute Error avg : 0.043010858706145155

Root Mean Squared Error avg: 0.06387109966110599

#### ▶ # Cross Validation

```
scores= cross_val_score(XGB, X_train, y_train, scoring='r2', cv=10)
XGB_CS=scores.mean()
print("Cross validation score is : ", XGB_CS)
```

Cross validation score is : 0.7830885454831675

#### ▶ # Hyper parameter Tuning using GridSearchCV

```
parameter= {'gamma':range(0,10,2),
            'max_depth': range(4,14,2),
            'feature_selector': ['cyclic','shuffle','random','greedy']}

grid_search=GridSearchCV(estimator=XGB, param_grid= parameter, cv=5)

grid_search.fit(X_train,y_train)

grid_search.best_estimator_
```

```
XGBRegressor
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, feature_selector='cyclic', gamma=0, gpu_id=-1,
              grow_policy='depthwise', importance_type=None,
              interaction_constraints='', learning_rate=0.300000012, max_bin=256,
              max_cat_to_onehot=4, max_delta_step=0, max_depth=8, max_leaves=0,
              min_child_weight=1, missing=nan, monotone_constraints=('',),
              n_estimators=100, n_jobs=0, num_parallel_tree=1, predictor='auto',
              random_state=0, reg_alpha=0, ...)
```

```
prediction= grid_search.best_estimator_.predict(X_test)

print("Post tuning scores")

#score variables
R2= round(r2_score(y_test, prediction), 4)
MAE=(mean_absolute_error(y_test,prediction))/(max(y)-min(y))
RMSE=(np.sqrt(mean_squared_error(y_test,prediction)))/(max(y)-min(y))
print("-----Test Score-----")
print(f" R^2 Score : {R2}\n")
print(f" MAE avg score : {MAE}\n")
print(f" RMSE avg score : {RMSE}\n")
```

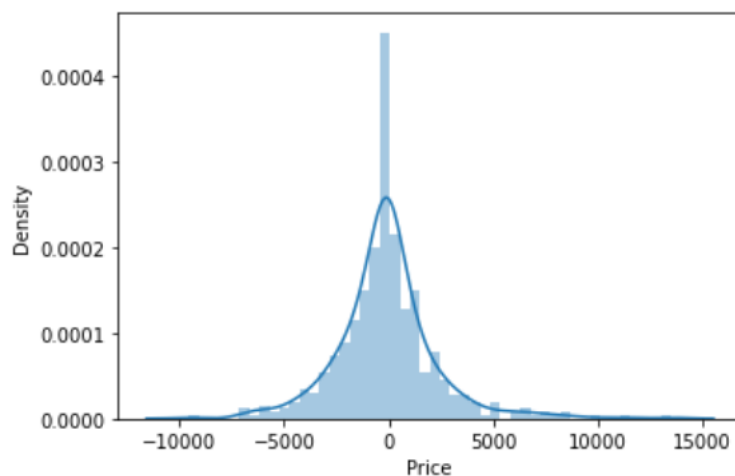
Post tuning scores  
-----Test Score-----  
R^2 Score : 0.8182  
  
MAE avg score : 0.042028275076095085  
  
RMSE avg score : 0.06400931125769403

```
train_pred=grid_search.best_estimator_.predict(X_train)
print("Post Tuning score for train data")
r2_score(y_train,train_pred)
```

Post Tuning score for train data  
0.9993976407090676

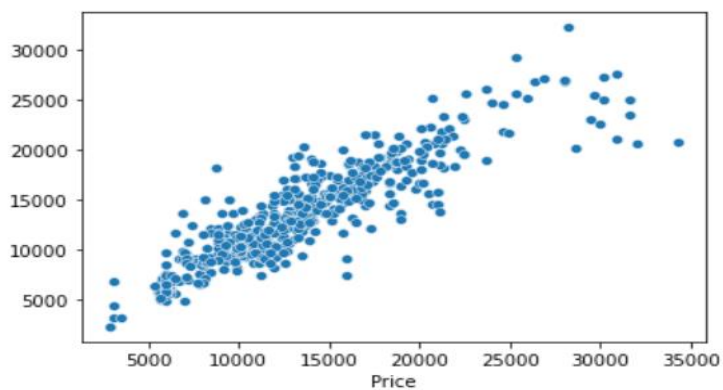
Let's do visualization, of our Best model.

```
► sns.distplot(y_test-prediction)
plt.show()
```



```
► sns.scatterplot(y_test,prediction)
```

`]: <AxesSubplot:xlabel='Price'>`



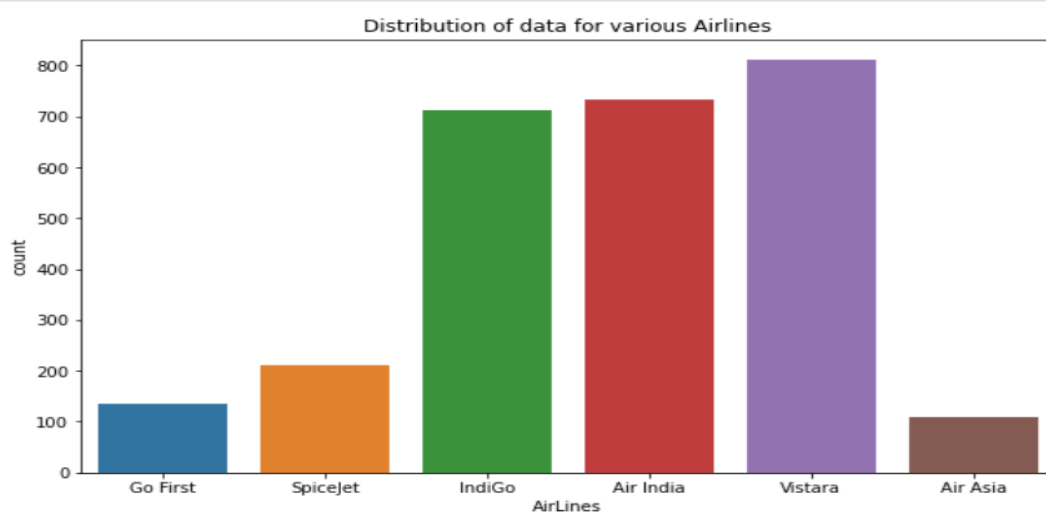
We can observe that both predicted value and actual value lies almost in the same line.

- Key Metrics for success in solving problem under consideration

- ✚ Here our Primary metrics is R2 score, as it gives us the goodness of fit measures, this statistic indicates the percentage of the variance in the dependent variable that the independent variables explain collectively.
- ✚ While our secondary metrics for this model is MAE and RMSE. The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It **measures accuracy for continuous variables**. While RMSE is the standard deviation of the residuals (prediction errors). It tells you **how concentrated the data is around the line of best fit**.

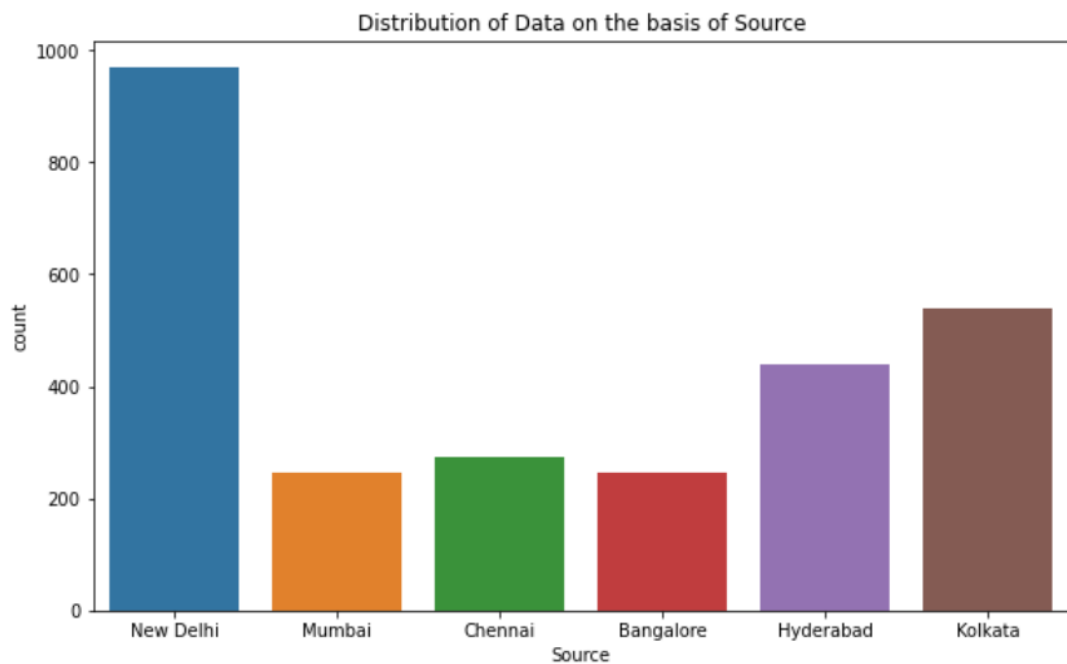
- Visualizations

```
# Let's do some Visualization
# Univariate Analysis.
plt.figure(figsize=(10,6))
sns.countplot(df['AirLines'])
plt.title("Distribution of data for various Airlines")
plt.show()
```



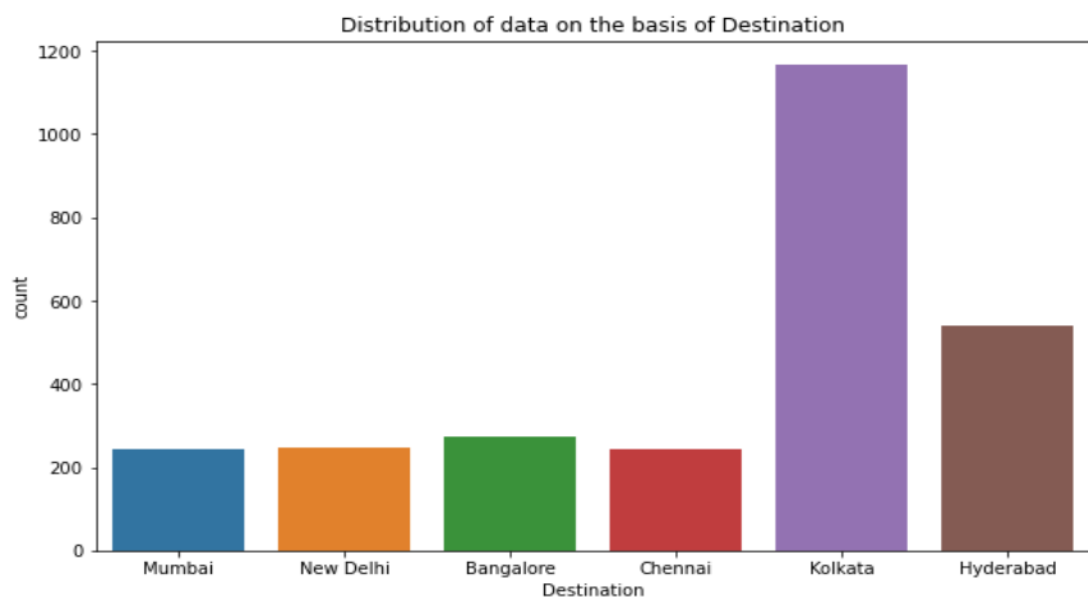
‘We can observe the distribution on data for various Airlines, Vistara has highest count while Air Asia has lowest count.

```
plt.figure(figsize=(10,6))
sns.countplot(df['Source'])
plt.title("Distribution of Data on the basis of Source")
plt.show()
```



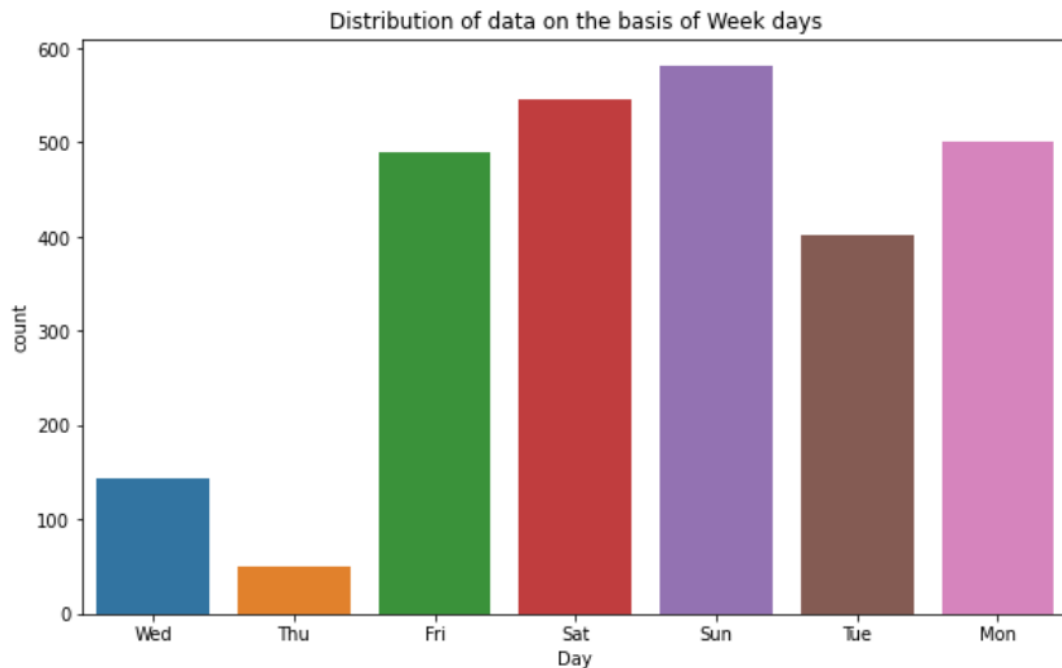
From above observation New Delhi has highest count as Source.

```
plt.figure(figsize=(10,6))
sns.countplot(df['Destination'])
plt.title("Distribution of data on the basis of Destination")
plt.show()
```



We can observe that Kolkata has highest count of data as destination.

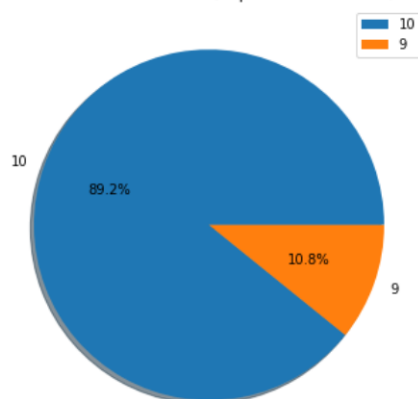
```
plt.figure(figsize=(10,6))
sns.countplot(df['Day'])
plt.title("Distribution of data on the basis of Week days")
plt.show()
```



We can see the distribution of data on the basis of week days. Sunday registered highest data.

```
plt.figure(figsize=(10,6))
plt.pie(df['Month'].value_counts(), labels=df['Month'].value_counts().index, shadow=True, autopct='%1.1f%%')
plt.title("Distribution of data on Month(September - October) Basis")
plt.legend()
plt.show()
```

Distribution of data on Month(September - October) Basis

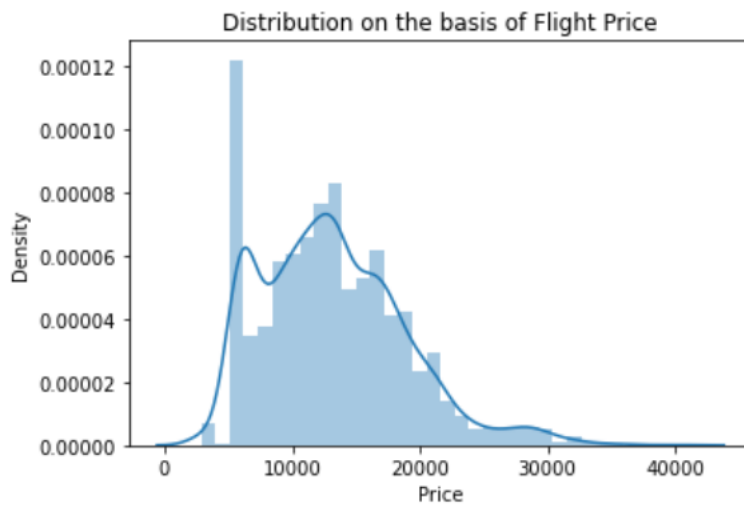


From above we can see that we got more data for October month compared to September.

```

# Distribution of Price column
sns.distplot(df['Price'])
plt.title('Distribution on the basis of Flight Price')
plt.show()

```

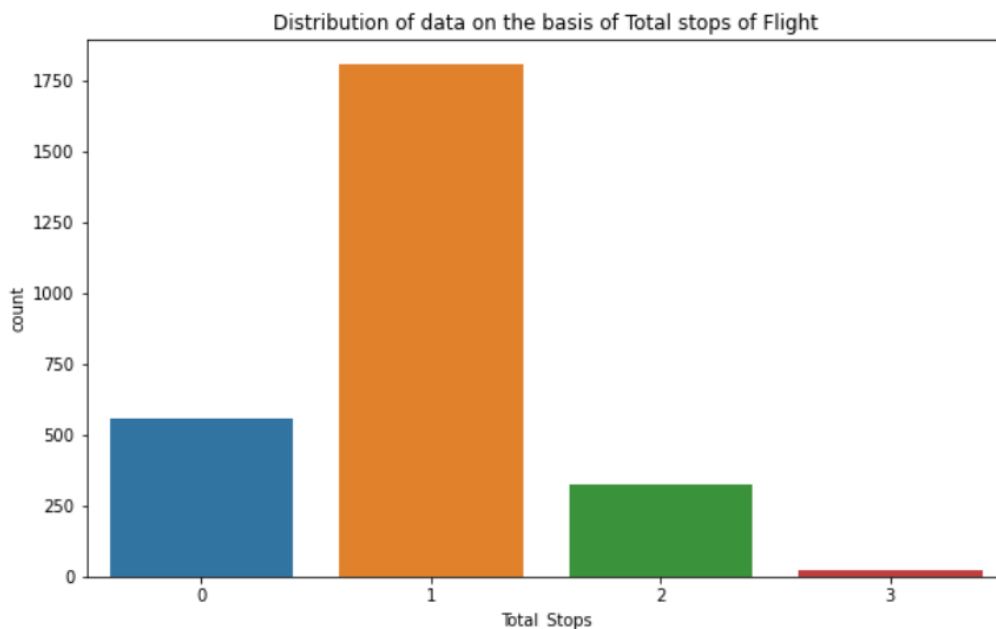


We can observe that the Price of Flight Ticket varies from somewhere 5000 to 30000. For our dataset, most of the flight tickets are on an average 5-7K.

```

# Distribution of data on the basis of total stops
plt.figure(figsize=(10,6))
sns.countplot(df['Total_Stops'])
plt.title("Distribution of data on the basis of Total stops of Flight")
plt.show()

```

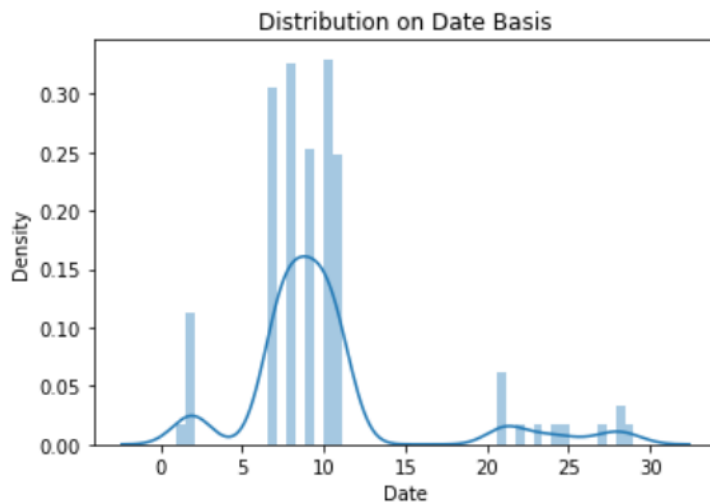


From above observation we see many flights got one stop, and we got least number of flights with 3+ stops.

```

# Distribution of data on the basis of date
sns.distplot(df['Date'])
plt.title("Distribution on Date Basis")
plt.show()

```

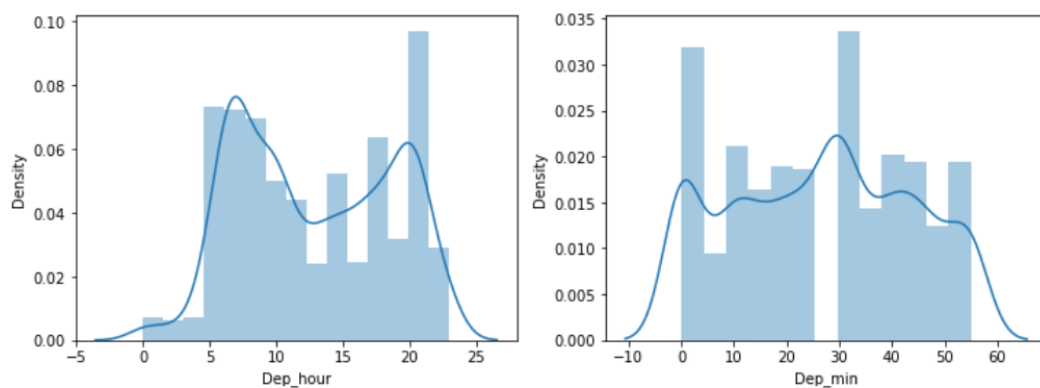


Most of the data available are either from starting dates or month end dates, We don't have data from middle of the month.

```

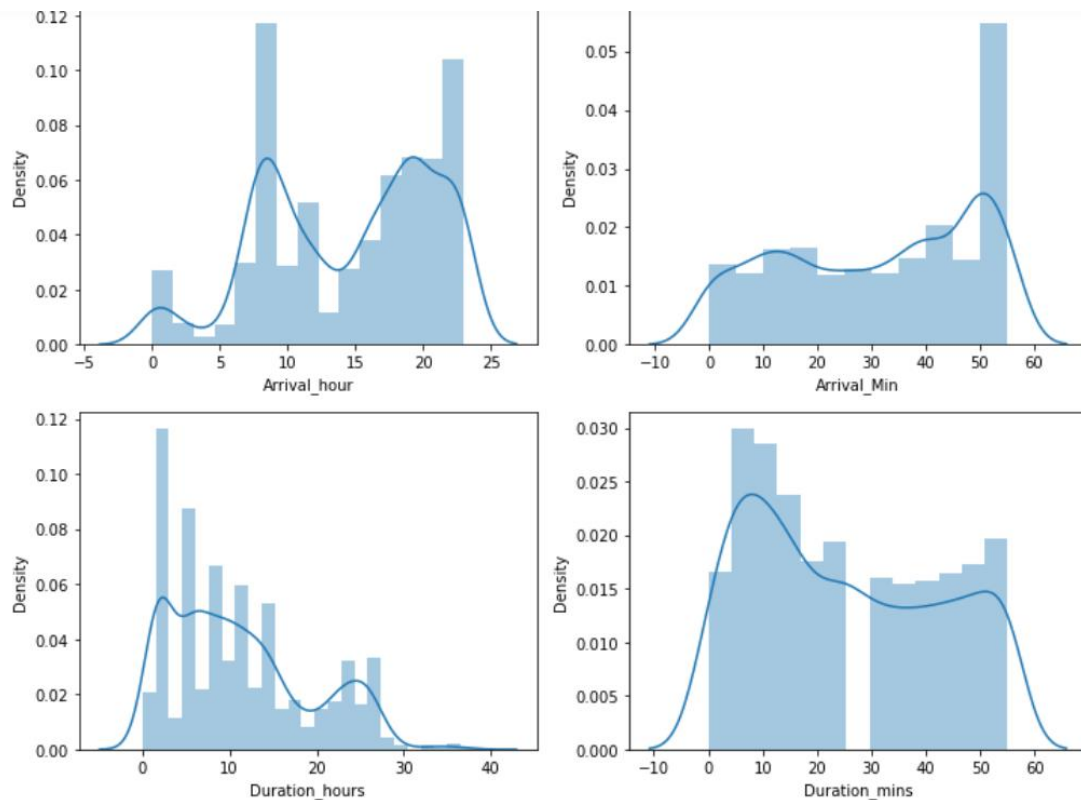
# Distribution for Dep_hour, Dep_min, Arrival_Hours, Arrival_Min, Duration_Hours, Duration_mins.
time_data=['Dep_hour', 'Dep_min', 'Arrival_hour', 'Arrival_Min', 'Duration_hours', 'Duration_mins']
plt.figure(figsize=(12,14), facecolor='white')
plotnumber=1
for column in df[time_data]:
    if plotnumber<=6:
        ax=plt.subplot(3,2,plotnumber)
        sns.distplot(df[column])
        plotnumber+=1
plt.show()

```



We can observe that our departure time data, which we feature engineer to make Dep\_hour and Dep\_min is discrete in nature, we can clearly see the distribution in the above plot.



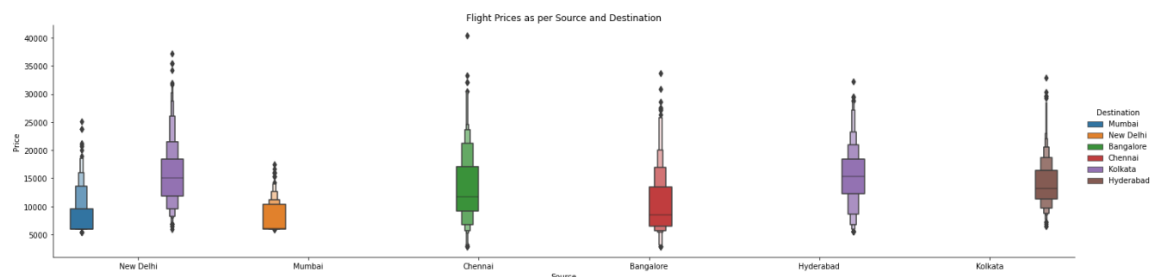


We can observe the data distribution on the basis of arrival time and total duration. The hours here are in 24 hrs presentation while mins are represented in 60 mins form. And we can see that data is distributed accordingly. And our now features are discrete in nature.

## Multivariate analysis

```
# Let's see how Price is related to source and destination.
sns.catplot(x='Source', y='Price', data=df, hue='Destination', kind='boxen', height=5, aspect=4)
plt.title("Flight Prices as per Source and Destination")
```

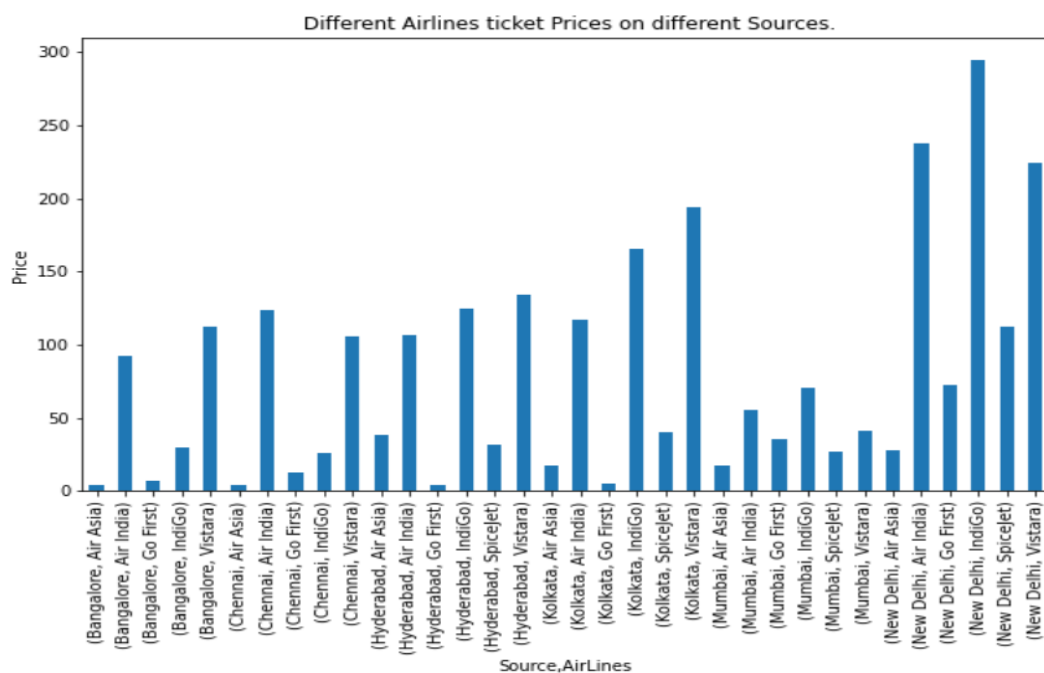
45]: Text(0.5, 1.0, 'Flight Prices as per Source and Destination')



We can observe that the average price for Kolkata flights are higher. while Delhi-Mumbai flights are comparatively cheap.

AirLines	Price
Air Asia	110
Air India	740
Go First	140
IndiGo	720
Spicejet	210
Vistara	820

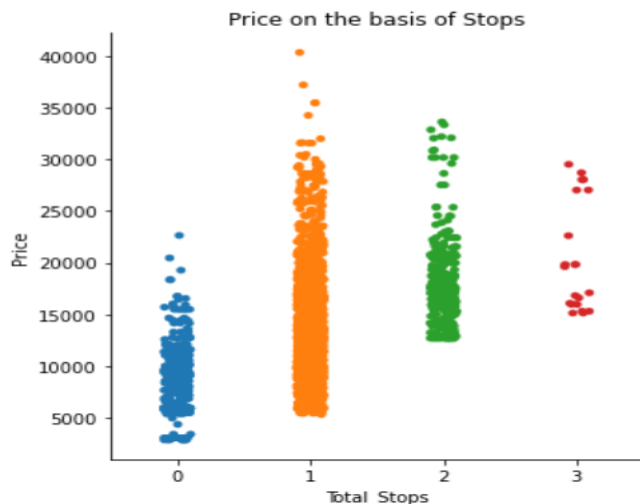
```
plt.figure(figsize=(10,6))
df.groupby(['Source', 'AirLines']).Price.count().plot.bar(ylim=0)
plt.ylabel('Price')
plt.title(" Different Airlines ticket Prices on different Sources.")
plt.show()
```



We can observe that New Delhi, Indigo has the highest Price Point while Chennai, AirAsia and Bangalore AirAsia has lowest Price point. Also we can observe that economical flight

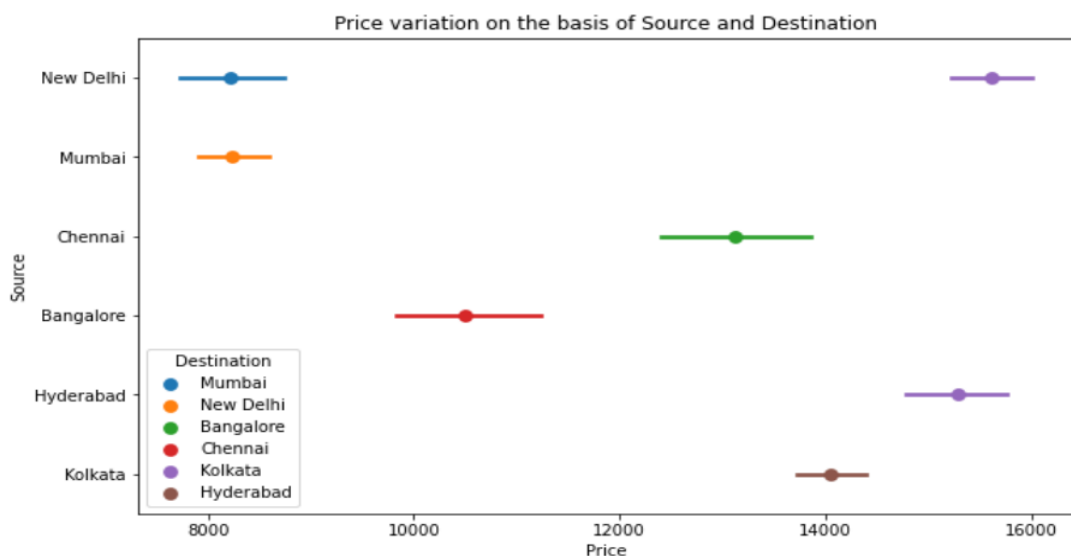
tickets are available for Air Asia, Go first while Indigo, Vistara, Aior India are comparatively costlier.

```
sns.catplot('Total_Stops', 'Price', data=df)
plt.title("Price on the basis of Stops")
]: Text(0.5, 1.0, 'Price on the basis of Stops')
```



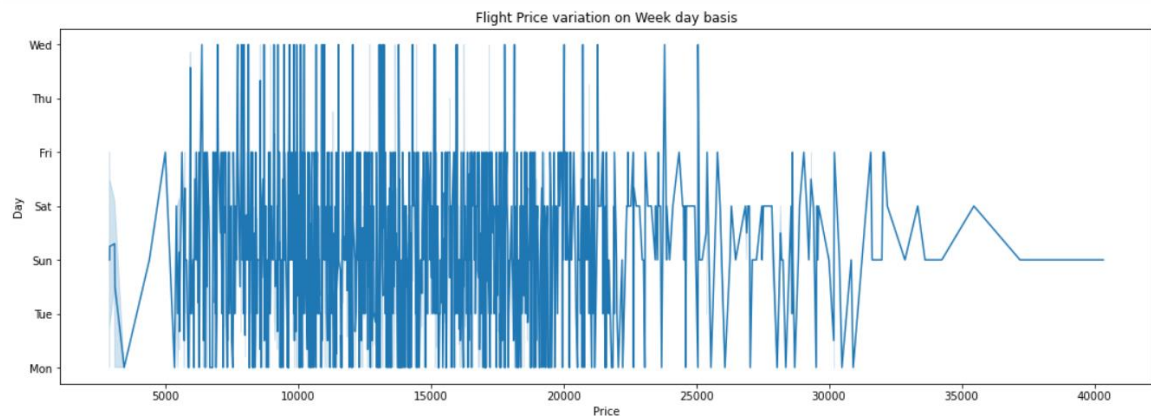
We can Observe that direct flights are cheaper than 1 stop or more stops flights. If the highest price for Non stop flight is around 23000, then we can say the highest price for stop flight is around 36-37K.

```
plt.figure(figsize=(10,6))
sns.pointplot(y='Source', x='Price', hue='Destination', data=df)
plt.title("Price variation on the basis of Source and Destination")
plt.show()
```



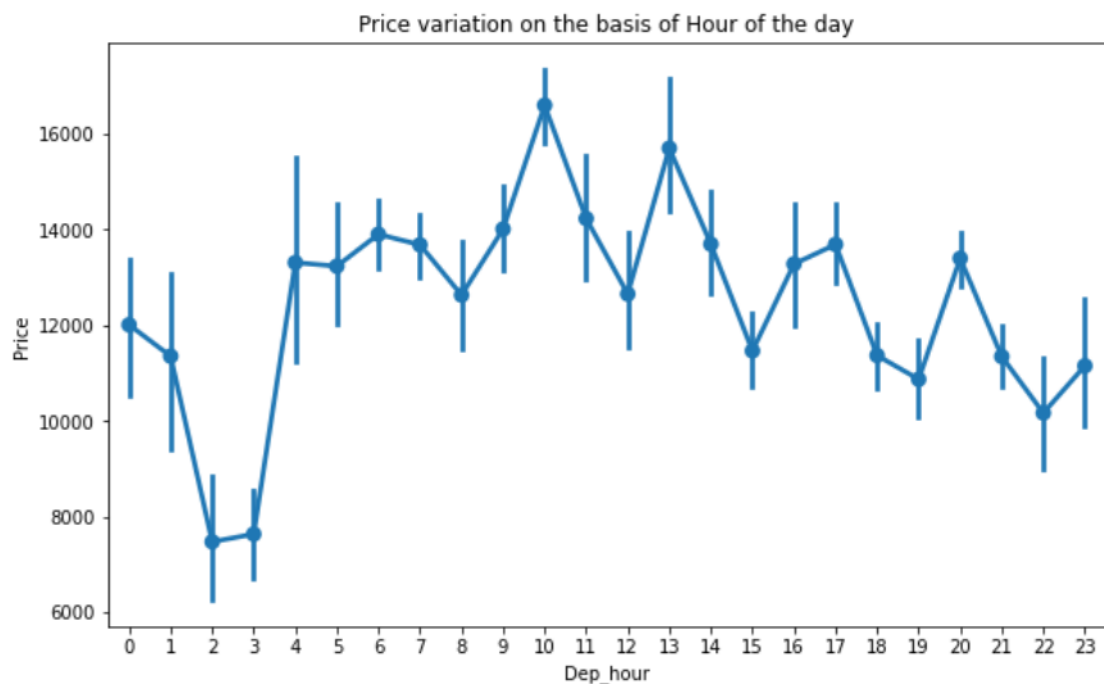
We can see that comparatively Delhi to kolkatta, Hyderabad-Kolkatta flights are more expensive in general. While Mumbai delhi, Delhi Mumbai flights are comparatively cheaper. In this plot, the x axis denotes the price, while y axis denotes Source, And colorful points in the plot denotes the destination source and price relationship.

```
plt.figure(figsize=(18,6))
sns.lineplot(x='Price', y='Day', data= df)
plt.title("Flight Price variation on Week day basis")
plt.show()
```



Although the weeks are not properly line up, we can clearly see that sunday's are expensive. We also observe that Friday, Saturday, Sunday and Monday are the days we have expensive tickets, while Tuesday, wednesday, Thursday are comparatively cheaper tickets.

```
plt.figure(figsize=(10,6))
sns.pointplot(x= 'Dep_hour', y='Price', data=df)
plt.title("Price variation on the basis of Hour of the day")
plt.show()
```

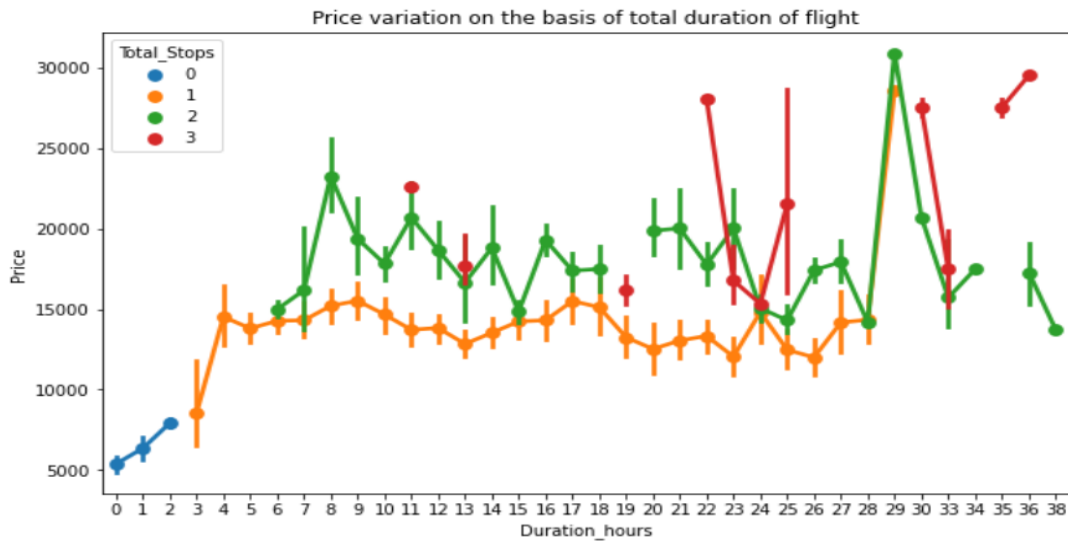


On the basis of this we can observe that the Flight Price varies with the hour of the day, generally Morning flights are expensive compared to late nights or midnight flights.

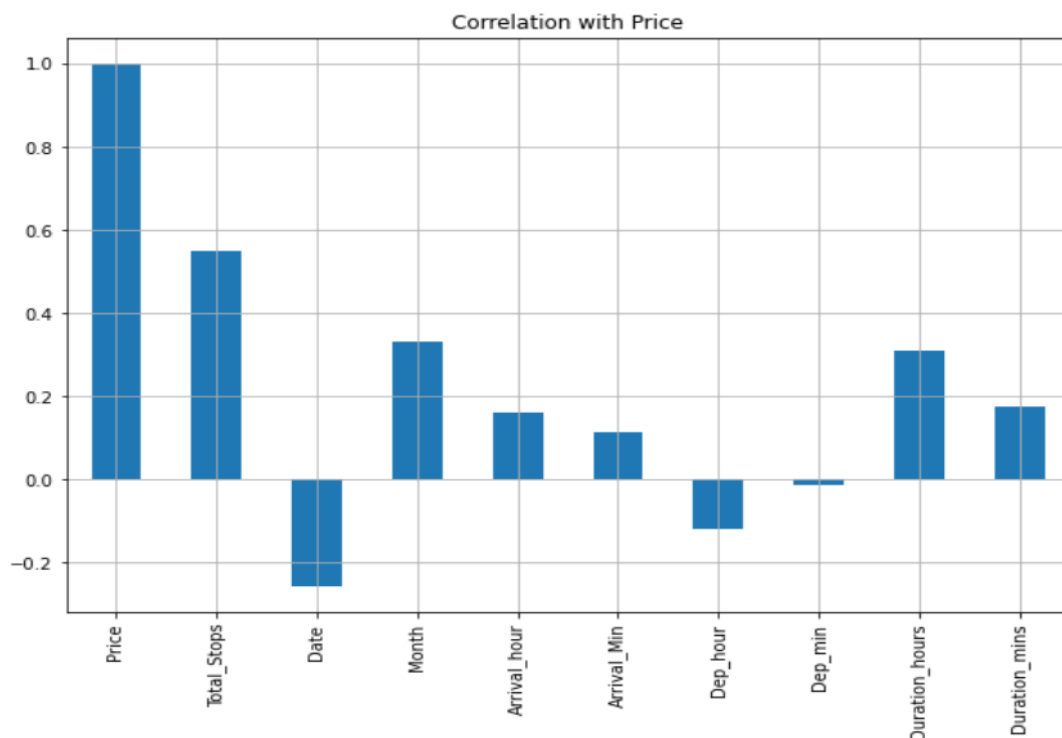
```

plt.figure(figsize=(10,6))
sns.pointplot(x='Duration_hours', y='Price', data=df,hue='Total_Stops')
plt.title("Price variation on the basis of total duration of flight")
plt.show()

```



From above observation we can say that Longer duration flights charge more money, i.e., the Flight ticket Prices are increases. We also observe that the flights with more number of stops also cost more, if its longer duration flights.



We can observe that Date, Dep\_hour and Depmin are inversely related to the flight Price. While Total\_stops, Month, Arrival\_hour, arrival\_Min, Duration\_Hours and Duration mins are positively correlated with flight Price. Inverse correlation means when value of one increases the value for other decreases and vice versa. While positive correlation means if one increases the other is also increases.

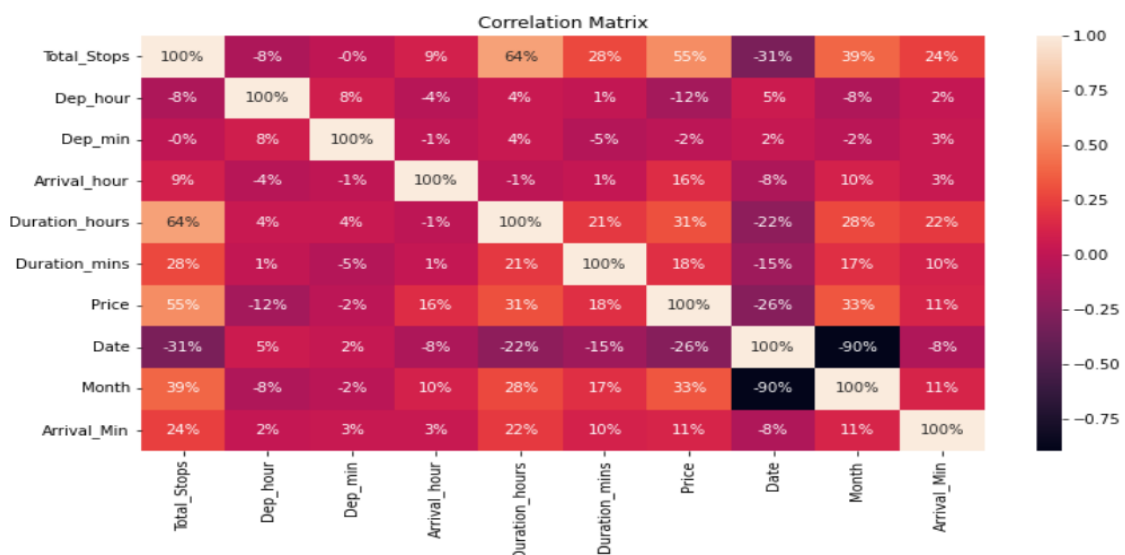
Here we can see, that Price of Flight Ticket is highly correlated with Total Stops, while least correlated with Dep\_min.

```
# Visualizing statistical description of dataset using Heatmap.
plt.figure(figsize=(14,7))
sns.heatmap(df.describe(), annot=True, fmt='.2f')
plt.show()
```



From above statistical description of data we can observe that, Price is our label or target variable. We have 2712 rows of data for all the features present in the dataset. Our dataset looks decent, with highest number of 3 total stops, all hourly details are in 23:00 terms and minutely details have max of 55 seems logical. In my opinion we can move ahead with this dataset.

```
# Visualizing correlation matrix using HeatMap.
plt.figure(figsize=(12,6))
sns.heatmap(df.corr(), annot=True, fmt='%.0%')
plt.title("Correlation Matrix")
plt.show()
```



We can Observe that Total\_stops is highly correlated Price, while Dep\_min is least correlated with Price. This correlation Matrix only consists of numerical features. We need to encode the the categorical data before moving ahead.

- **Interpretation of the Results**

By visualization we can see that, the label i.e., Price is greatly impacted by the features. Some features show positive correlation while other show negative correlation.

In short, we can say that Price column is dependent variable which is definitely affected by other independent variables.

Price shows highest correlation with total Stops, we also observe that Day and Departure Time also plays a significant role as the deciding factor for Flight Price.

We also see Airlines, is also one of the important features which shows strong correlation with Price.

## **CONCLUSION**

- **Key Findings and Conclusions of the Study.**

- ✚ Price shows highest correlation with total Stops, we also observe that Day and Departure Time also plays a significant role as the deciding factor for Flight Price.

- ✚ We also see Airlines, is also one of the important features which shows strong correlation with Price.

- ✚ We also observe that, as per the dataset we collect Ensemble technique and advanced boosting techniques handle our model effectively, as compared to linear model or regularization model.

- ✚ Also, Source and Destination Feature is the key for determining the flight ticket prices.

- ✚ About date and month, as we don't have sufficient information for throughout the year, what little we have we can say that ticket price varies with respect to the duration when you are booking the ticket and when you actually travel.

- **Learning Outcomes of the Study in respect of Data Science**

As we collect data from real time website, so its very import to cover all the features in current page in short interval , as page gets expired real quick, and there are chances of data miss match, or one need to run the entire code again just to tally the data.

In part of model building, as the entire data columns are object data type, so it is really important to treat each column with proper understanding. A slight mistake can lead to important information loss.

There are certain features which are important like date of journey, as it helps us to relate how ticket prices varies w.r.t. date of journey. But in our dataset we don't find such wide variety regarding it.

- **Limitations of this work and Scope for Future Work**

Model requires retraining after certain duration of time, as per current aviation scenario, these airlines are working in domestic domain. In future, things would change.

Also, our data consist of domestic travel only, one can volume up the data by including International Travel also. One more thing can be done, is to include variety of date of journey, as it helps to predict for year long data.