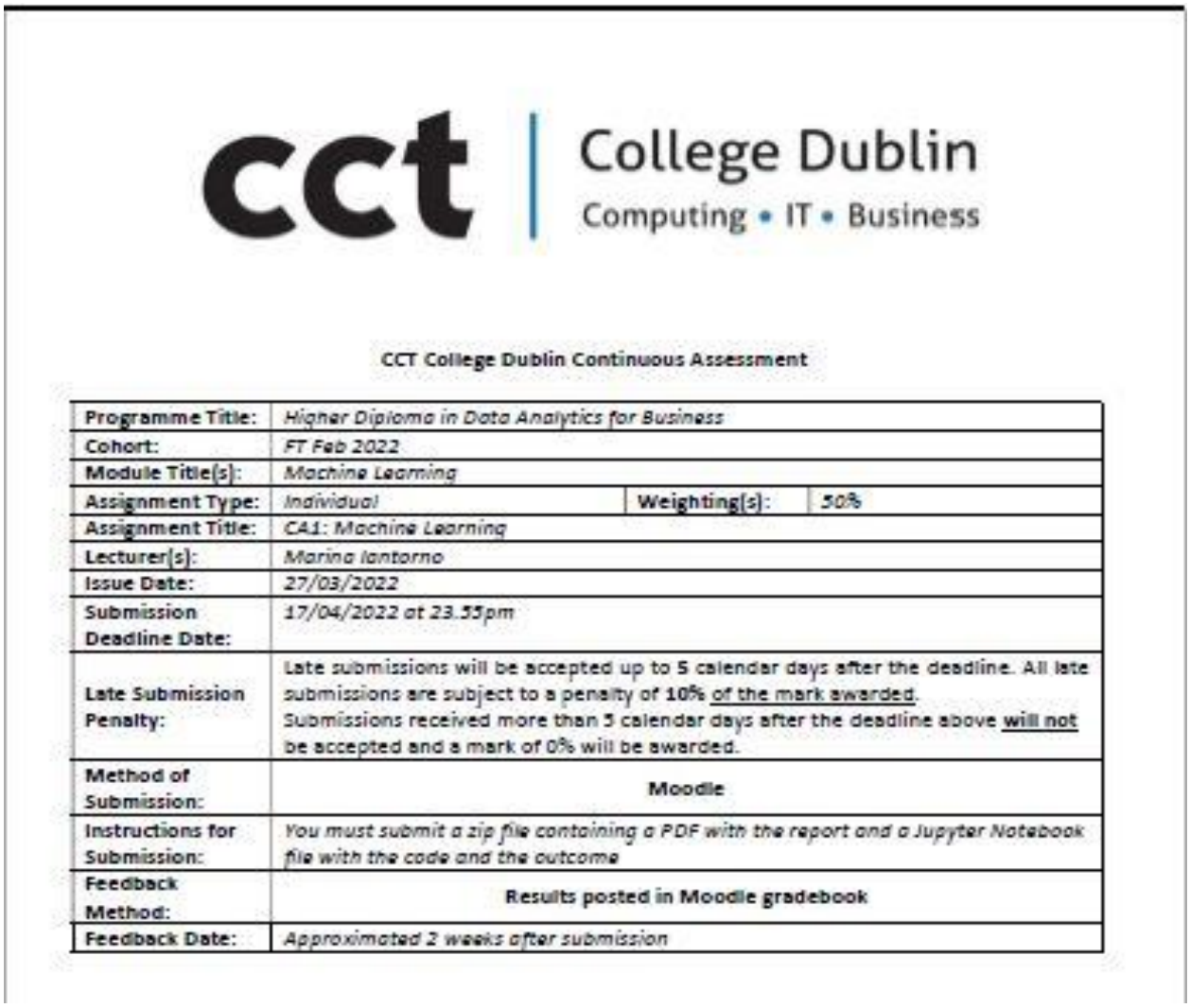
Assignment title:

MachLearn_HDipData_Feb22_FT CA

Submitted by: Muhammad Nauman

Student ID:2022097



cct | College Dublin
Computing • IT • Business

CCT College Dublin Continuous Assessment

Programme Title:	*Higher Diploma in Data Analytics for Business*		
Cohort:	*FT Feb 2022*		
Module Title(s):	*Machine Learning*		
Assignment Type:	*Individual*	**Weighting(s):**	*50%*
Assignment Title:	*CA1: Machine Learning*		
Lecturer(s):	*Marina Iantorno*		
Issue Date:	*27/03/2022*		
Submission Deadline Date:	*17/04/2022 at 23.55pm*		
Late Submission Penalty:	Late submissions will be accepted up to 5 calendar days after the deadline. All late submissions are subject to a penalty of 10% of the mark awarded. Submissions received more than 5 calendar days after the deadline above will not be accepted and a mark of 0% will be awarded.		
Method of Submission:	**Moodle**		
Instructions for Submission:	*You must submit a zip file containing a PDF with the report and a Jupyter Notebook file with the code and the outcome*		
Feedback Method:	**Results posted in Moodle gradebook**		
Feedback Date:	*Approximated 2 weeks after submission*		

Analysis and predictions on Flight prices

Abstract.

Determining flight prices is very challenging task now a days for aeroplane companies' and customers both as flight prices are dynamically changes.

The focus of this project is to apply data science and machine learning techniques that 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.

Introduction

Project scenario

In India a company have a business of air travel and tours. It provides air tickets of many airlines companies to people across different cities. It decides its flight prices manually due to which these problems are generated:

- As flight prices changes day to day due to multiple factors.
- Labour cost of company is increased as many people have involved in this prediction process .Company pays high salary to these people.
- The process of predictions takes huge time.
- Accuracy of predictions is not good.
- Company suffers loss in some cities due to incorrect pricing.
- Customers have trouble in deciding prices

To overcome above mentioned problems this Air Travel company decides to make a robust prediction system for its customers and managers which have following features:

- If changes happened in prices of fuel or any factors of change in flight prices then prices of flight changes updated automatically.
- It helps people In deciding prices by inputting facilities.
- It is an automated system.

Data Description:

The dataset which I am going to use here is dataset of flight prices in India. It is openly available on Kaggle.

Dataset source: https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh

This dataset has 11 features which are following:

1. **Airline:**
   This feature contains names of airline 12 companies of India
2. **Date_of_Journey:**
   This feature contains day, month and year of departure.
3. **Source:**

This feature contains cities names of India from where air journey starts.

4. **Destination**:
5. This feature contains cities names of India from where air journey ends.

6. **Route:**
   This feature contains different paths of passing aeroplane as airlines fly over many ci ties .arrow diagram is used here to show paths.
   There are 129 routs of aeroplane.

7. **Dep_Time:**
   This feature contains time of departure of flight in hours and minutes.
8. **Arrival_Time:**
   This feature contains time of arriving of flight in hours and minutes.
9. **Duration:**
   This feature represents The time taken by a flight. It is difference of arrival time and departure time.
10. **Total_Stops**:
    This feature tells that how many places on which aeroplane lands during it journey.
11. **Additional_Info:**
    This feature tells some extra information about a flight
12. **Price:**
    This is targeted feature which represent price of air flights in Indian rupees (INR)

Reason for Analysis:

The reasons of doing this Analysis are following:

- To extract information, facts and figures from raw data
- To understand pattern of data.
- To know about trends.
- To make visualizations to represents data in the form of graphs and charts. As a pictu re is worth than 1000 words.
- To show what relationships different feature are having among them.
- To make an accurate Machine Learning Model which can be used to predict flight pri ces on the basis of input data?

Technical requirements:

There are two main tools which are used to do analysis which are following:

- **Python:**
  Python is programming language which is used here .I have to perform this data analysis usi ng python
- **Jupiter notebook:**
  Jupiter notebook is tool on have to I have to write python script

Importing libraries and dataset

The first step is to import all required libraries.so here I am importing four Major Libraries which are here:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

1)**pandas**:
pandas is used for data crunching.
2)**numpy** :
It is used for scientific computing it provides some mathematics and statistics like functions.
3)**Matplotlib:**
It is used for data Visualization For example to make Charts and graphs.
4)**Seaborn**:
It is also used for data Visualization For example to make Charts and graphs.

so these are four major python libraries which I have imported to use.

Import csv file and create a data frame

Now here I am loading data . As data is in excel file so I am importing Excel File with .xlsx file format.

```
train_data = pd.read_excel(r"E:\ML Projects\Flight-Price-Prediction-master\Flight-Price-Prediction-master\Data_Train.xlsx")
```

```
pd.set_option('display.max_columns', None)
```

```
train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

As here excel data is loaded into a data frame 'train data'.
Data Frame is also like a table.
At the end first 5 records are displayed.

Size/shape of data and general information about data

```
train_data.shape
```

```
(10683, 11)
```

```
train_data.info()
```

```
<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 
```

There are 11 features or columns 10683 rows or records in dataset.
Except Price all features are not in numerical form.

Checking for any missing values

```
train_data.isnull().sum()
```

```
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
```

As shown above there are only two missing or null values:

Route feature has one null value and Total stops feature has one null value.

Removing/dropping missing values

```
train_data.dropna(inplace = True)
train_data.shape
```

```
(10682, 11)
```

As two rows which have missing are removed so now dataset have 10682 rows

Feature Engineering

Making a new feature journey day

```
train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey, format="%d/%m/%Y").dt.day
train_data["Journey_day"]
```

```
0        24
1         1
2         9
3        12
4         1
         ..
10678     9
10679    27
10680    27
10681     1
10682     9
Name: Journey_day, Length: 10682, dtype: int64
```

Here a new feature Journey day is created by extracting day from feature “Date of journey “

Making a new feature Journey month

```
train_data["Journey_month"] = pd.to_datetime(train_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
train_data["Journey_month"]
```

```
0        3
1        5
2        6
3        5
4        3
        ..
10678    4
10679    4
10680    4
10681    3
10682    5
Name: Journey_month, Length: 10682, dtype: int64
```

Here a new feature Journey month is created by extracting month from feature “Date of journey “

Making a new feature Journey year

```
train_data["Journey_year"]=pd.to_datetime(train_data["Date_of_Journey"], format = "%d/%m/%Y").dt.year
train_data["Journey_year"]
```

```
0        2019
1        2019
2        2019
3        2019
4        2019
         ...
10678    2019
10679    2019
10680    2019
10681    2019
10682    2019
Name: Journey_year, Length: 10682, dtype: int64
```

```
train_data["Journey_year"].unique()
```

```
array([2019], dtype=int64)
```

Here a new feature Journey year is created by extracting year from feature “Date of journey “.

As there is only one year involved here which is 2019.

Removing Date o Journey and Journey year feature

```
# Since we have converted Date_of_Journey column into integers, Now we can drop as it is of no use.

train_data.drop(["Date_of_Journey","Journey_year"], axis = 1, inplace = True)
```

Here date of journey is converted to two more features so it is not in use and Journey Year has only one value which is also not required so that is why two features are removed.

Exploring departure time feature

```
train_data["Dep_Time"]
# Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time
```

```
0        22:20
1        05:50
2        09:25
3        18:05
4        16:50
         ...
10678    19:55
10679    20:45
10680    08:20
10681    11:30
10682    10:55
Name: Dep_Time, Length: 10682, dtype: object
```

Departure time feature shows hours and minutes so it is needed to extract hours and minutes from this feature

Making a new feature departure hour.

```
# Extracting Hours
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour
train_data["Dep_hour"]
```

```
0        22
1         5
2         9
3        18
4        16
         ..
10678    19
10679    20
10680     8
10681    11
10682    10
Name: Dep_hour, Length: 10682, dtype: int64
```

Here a new feature departure hour is created by extracting hours from departure time feature.

Making a new feature departure minute

```
# Extracting Minutes
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute
train_data["Dep_min"]
```

```
0        20
1        50
2        25
3         5
4        50
         ..
10678    55
10679    45
10680    20
10681    30
10682    55
Name: Dep_min, Length: 10682, dtype: int64
```

Here a new feature departure minute is created by extracting minutes from departure time feature.

Removing Departure time feature

```
train_data.drop(["Dep_Time"], axis = 1, inplace = True)
```

Here departure time feature is converted to two more features so it is not in use. That is why departure time feature is removed.

Exploring Arrival Time feature

```
train_data['Arrival_Time']
# Arrival time is when the plane pulls up to the gate.
# Similar to Date_of_Journey we can extract values from Arrival_Time
```

```
0        01:10 22 Mar
1               13:15
2        04:25 10 Jun
3               23:30
4               21:35
              ...
10678           22:25
10679           23:20
10680           11:20
10681           14:10
10682           19:15
Name: Arrival_Time, Length: 10682, dtype: object
```

Arrival time feature contains hours and minutes.so it is needed to extract hours and minutes.

Making a new feature arrival hour

```
# Extracting Hours
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour
train_data["Arrival_hour"]
```

```
0         1
1        13
2         4
3        23
4        21
         ..
10678    22
10679    23
10680    11
10681    14
10682    19
Name: Arrival_hour, Length: 10682, dtype: int64
```

Here a new feature arrival hour is created by extracting hours from arrival time feature.

Making a new feature arrival minute

```
# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute
train_data["Arrival_min"]
```

```
0        10
1        15
2        25
3        30
4        35
         ..
10678    25
10679    20
10680    20
10681    10
10682    15
Name: Arrival_min, Length: 10682, dtype: int64
```

Here a new feature arrival minute is created by extracting minutes from arrival time feature.

Removing arrival time feature

```
# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
```

Arrival time feature is converted to two more features so it is not in use. That is why arrival time feature is removed.

Exploring duration feature

```
train_data["Duration"]
```

```
0        2h 50m
1        7h 25m
2           19h
3        5h 25m
4        4h 45m
          ...
10678    2h 30m
10679    2h 35m
10680        3h
10681    2h 40m
10682    8h 20m
Name: Duration, Length: 10682, dtype: object
```

Here duration feature is difference between arrival time and departure time .It contains hours and minutes. It is needed to convert this in minutes only

Making lists of minutes and hours

```
# Time taken by plane to reach destination is called Duration
# It is the differnce betwwen Departure Time and Arrival time


# Assigning and converting Duration column into list
duration = list(train_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"   # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))   # Extracts only minutes from duration
```

Here above two lists duration minutes and duration hours is created by separating other characters in duration feature.

Making a new feature duration minutes

```
train_data["Duration_mins"] = np.array(duration_hours)*60+np.array(duration_mins)
train_data["Duration_mins"]
```

```
0         170
1         445
2        1140
3         325
4         285
         ...
10678     150
10679     155
10680     180
10681     160
10682     500
Name: Duration_mins, Length: 10682, dtype: int32
```

Here above duration minutes feature is created by multiplying hours with sixty and then by adding product to duration minutes.

Removing duration feature

```
train_data.drop(["Duration"], axis = 1, inplace = True)
```

Duration feature is removed as it is converted to duration minutes feature so it is not in use so that is why it is removed.

Exploratory data analysis

Extracting qualitative features

```
qual_features=[feature for feature in train_data.columns if len(train_data[feature].unique())<10]
qual_features
```

```
['Source', 'Destination', 'Total_Stops', 'Journey_month']
```

```
for feature in qual_features:
    print(feature)
    print(train_data[feature].unique())
```

```
Source
['Banglore' 'Kolkata' 'Delhi' 'Chennai' 'Mumbai']
Destination
['New Delhi' 'Banglore' 'Cochin' 'Kolkata' 'Delhi' 'Hyderabad']
Total_Stops
['non-stop' '2 stops' '1 stop' '3 stops' '4 stops']
Journey_month
[3 5 6 4]
```

Here above qualitative features are those features which are non-numerical features having less than 10 categories.

These features are also called categorical features.

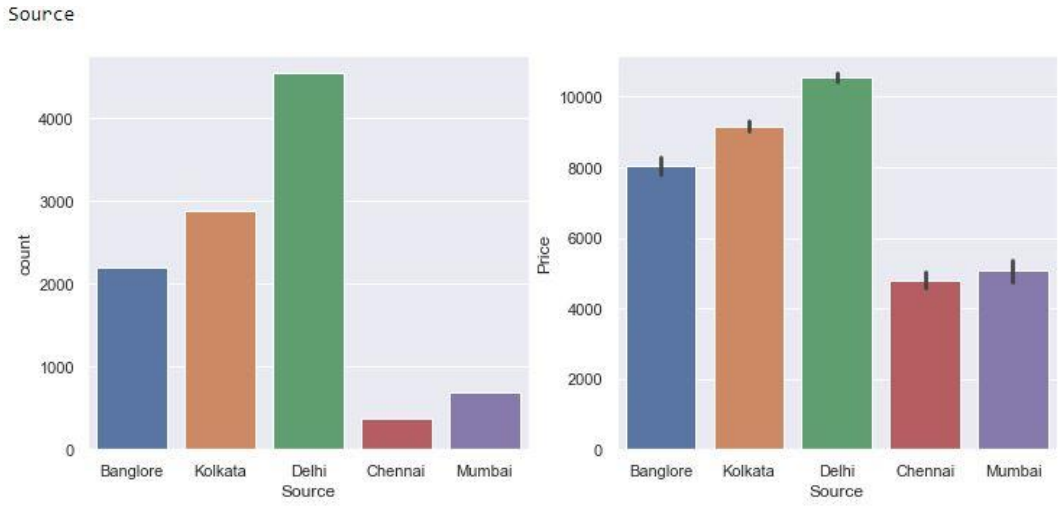
'Source', 'Destination', 'Total Stops', 'Journey month' are four categorical features here.

Here unique values of every categorical feature are also shown.

Exploring categorical features using bar charts

```
for feature in qual_features:
    print(feature)
    plt.figure(figsize=(12,5))
    plt.subplot(1, 2, 1)
    sns.countplot(x=feature,data=train_data)
    plt.subplot(1, 2, 2)
    sns.barplot(x=feature,y='Price',data=train_data)
    plt.show()
```

Relationship of source with prices

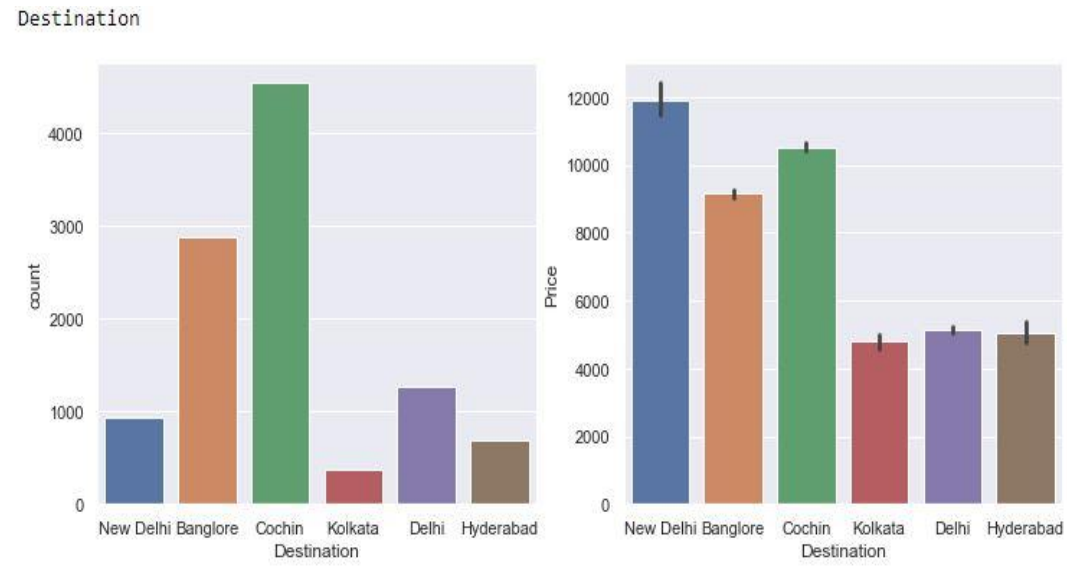


Left bar chart shows that Delhi is top city of departure in most of flights.

Right bar charts shows that Delhi is top city of departure which have very high price.

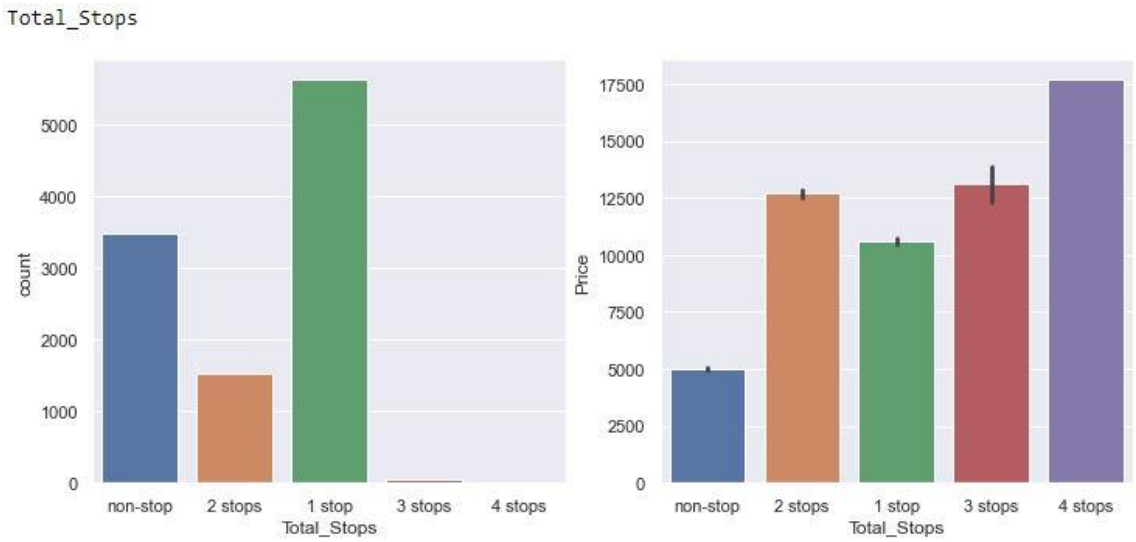
A Positive relationship is shown as by looking both graphs

Relationship of destination with prices



Left bar chart shows that New Delhi is city of arrival with less flight operations but right chart shows reverse relation of flight operations with price as New Delhi is top city with price

Relationship of source with prices

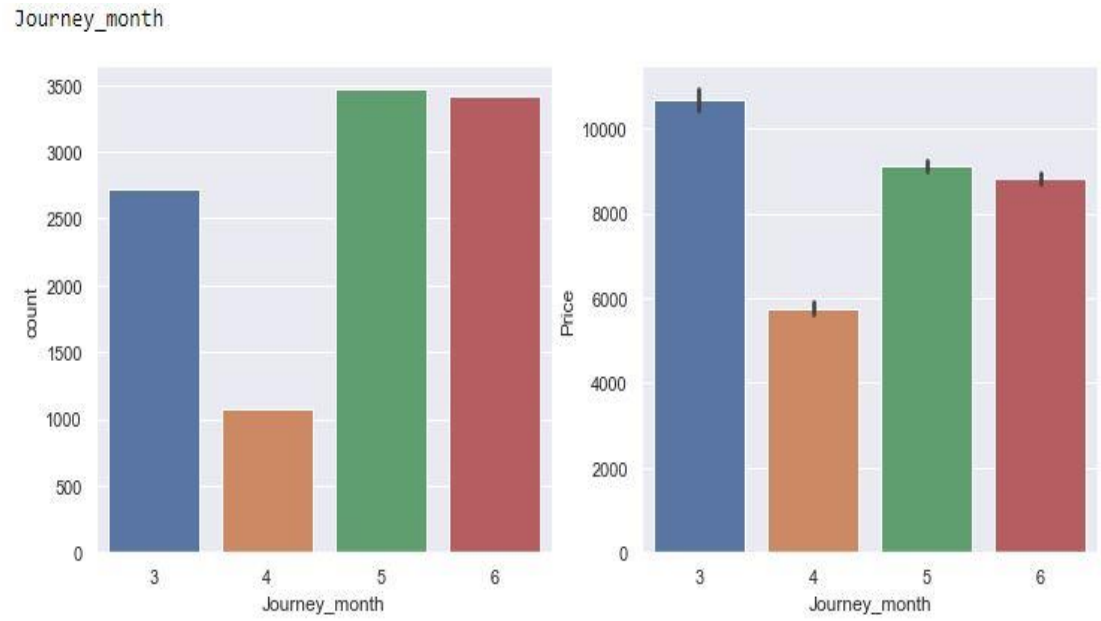


Left bar chart shows that 1-stop is used in most of flight operations but right chart shows that 4 stop is most expensive

The relation of number of stops and prices is positive as non-stop have very less price and 4 stops has top prices

But relationship of flight operations and stops is not positive as 4 stops is very less used in flight operations but very high paid

Relationship of journey month with prices



The above left and right bar chart shows positive relationship between flights of journey month and prices associated with journey month.

Quantitative features:

```
quant_features=[feature for feature in train_data.columns if train_data[feature].dtype!='O' and
               len(train_data[feature].unique())>10 ]
quant_features
```

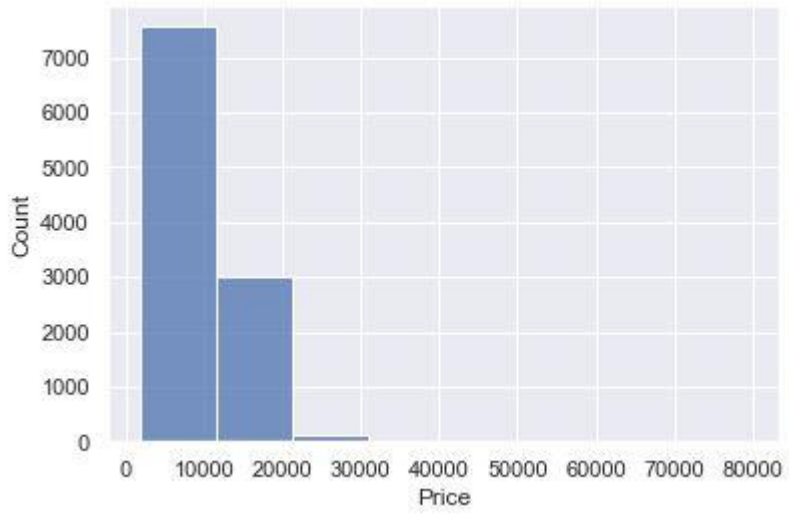
```
['Price',
 'Dep_hour',
 'Dep_min',
 'Arrival_hour',
 'Arrival_min',
 'Duration_mins']
```

There are six features which are in numerical form

Showing distribution of values in quantitative features using histogram

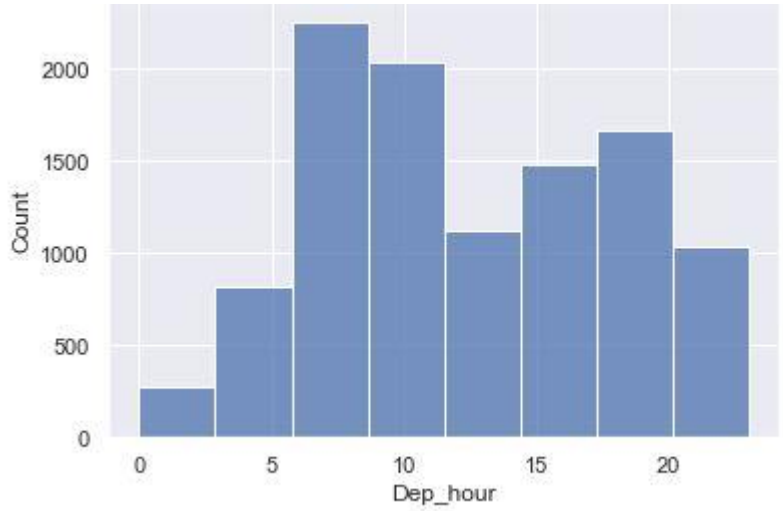
```
for feature in quant_features:
    sns.histplot(train_data[feature],bins=8)
    plt.show()
```

Distribution of values in price feature



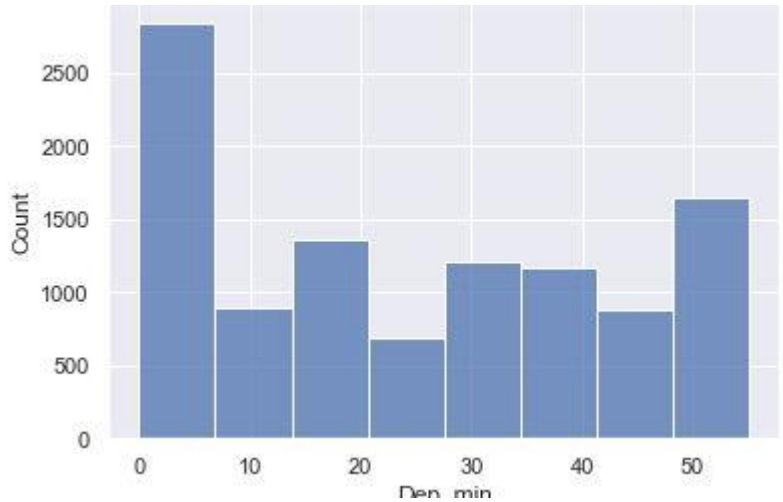
As 10000 is most usable price in flights as shown in above histogram

Distribution of values in departure hour feature



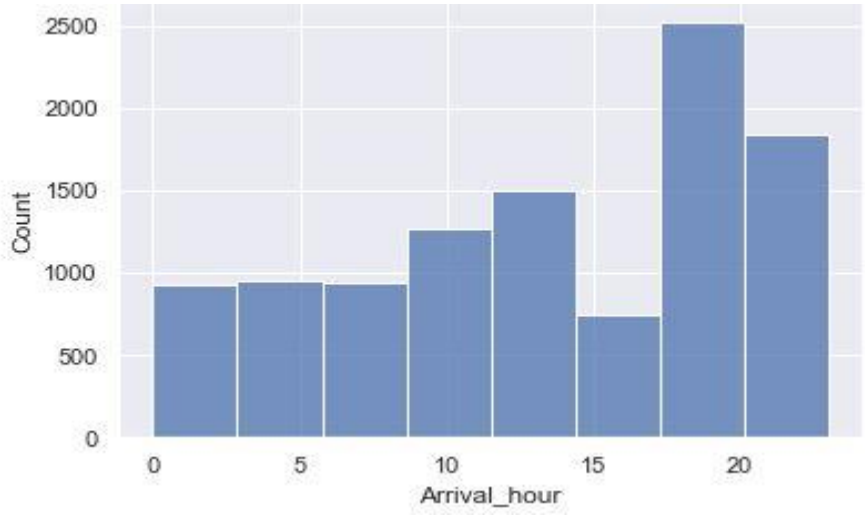
Above histogram shows that most of flight operations occurs at morning 5am to 10am.

Distribution of values in Departure min feature



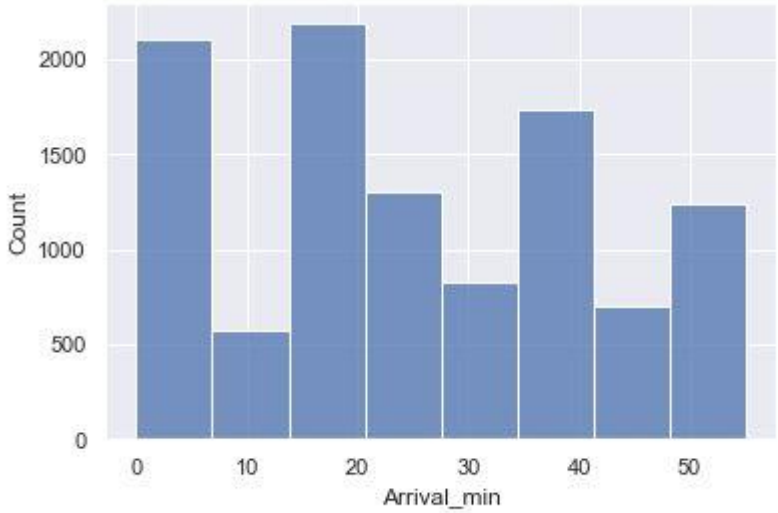
As 0-10 minutes is mostly occurred which is shown I above histogram

Distribution of values in Arrival hour feature



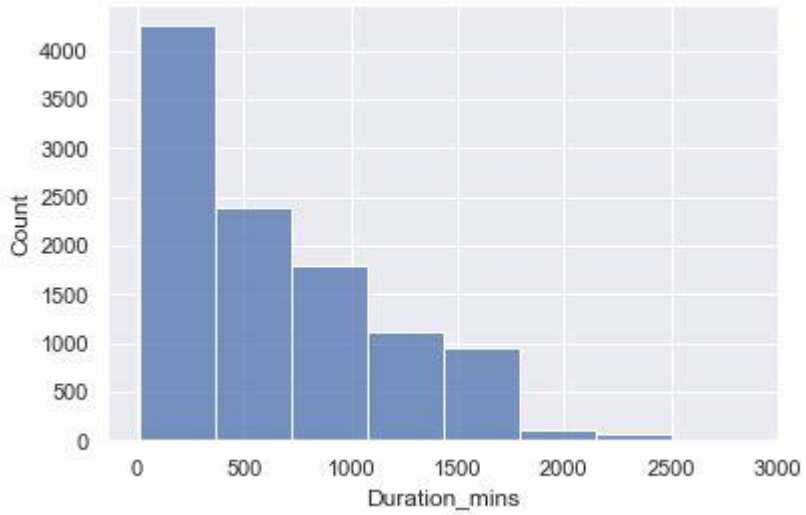
Above histogram shows that 8pm night is arrival time of most of the flights.

Distribution of values in arrival minute feature



Above histogram shows that range of 15-20 minutes is arrival time of most of the flights.

Distribution of values in duration minute feature



Above bar chart shows that most of the flight operations takes 0-500 minutes of duration during flight.

```
high_catagary_features= [feature for feature in train_data.columns if train_data[feature].dtype=='O' and
               len(train_data[feature].unique())>=10 ]
high_catagary_features
```

```
['Airline', 'Route', 'Additional_Info']
```

```
for feature in high_catagary_features:
    print(feature)
    print(len(train_data[feature].unique()))
```

```
Airline
12
Route
128
Additional_Info
10
```

There are three features which have high categories which are:

`'Airline', 'Route', 'Additional Info'`

Handling categorical features using One hot encoding

```
# As Airline is Nominal Categorical data we will perform OneHotEncoding
Airline = train_data[["Airline"]]
Airline = pd.get_dummies(Airline, drop_first= True)
Airline.head()
```

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airline_Trujet	Airline_Vistara	Airline_Vista Premiu econor
0	0	0	1	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	0	0	

Here airline feature is converted to 11 binary features.

```
# As Source is Nominal Categorical data we will perform OneHotEncoding

Source = train_data[["Source"]]

Source = pd.get_dummies(Source, drop_first= True)

Source.head()
```

	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

Here source feature is converted to 4 binary features.

```
# As Destination is Nominal Categorical data we will perform OneHotEncoding

Destination = train_data[["Destination"]]

Destination = pd.get_dummies(Destination, drop_first = True)

Destination.head()
```

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

Here destination feature is converted to 5 binary features.

Removing route and additional info feature

```
# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other

train_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

As these features not in use so these are removed

Replacing values in total stops feature

```
# As this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding keys

train_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)
```

Here categories are removed with numerical values.

Concatenating data frames

```
# Concatenate dataframe --> train_data + Airline + Source + Destination

data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)
```

As the result of one hot encoding there are three datasets which are generated so these are added by features in train data

Removing three features "Airline", "Source" and "Destination"

```
data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
```

As these features are converted so it is not used so that is why it is removed

```
data_train.shape
```

```
(10682, 29)
```

There are 29 features which are involved here.

```
data_train.columns
```

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_mins',
       'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
       'Airline_Jet Airways', 'Airline_Jet Airways Business',
       'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

```
len(data_train.columns)
```

```
29
```

There are 29 features involved in data training

Preparing dataset for regression

Splitting of data into dependant and independent

```
X = data_train.drop(['Price'],axis=1)
```

```
X.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour', 'Dep_min',
       'Arrival_hour', 'Arrival_min', 'Duration_mins', 'Airline_Air India',
       'Airline_GoAir', 'Airline_IndiGo', 'Airline_Jet Airways',
       'Airline_Jet Airways Business', 'Airline_Multiple carriers',
       'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
       'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

Price is removed as it is included as target feature.

```
y = data_train.iloc[:,1]
y.head()
```

```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

Visualizing feature importance

```
# Important feature using ExtraTreesRegressor

from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
```

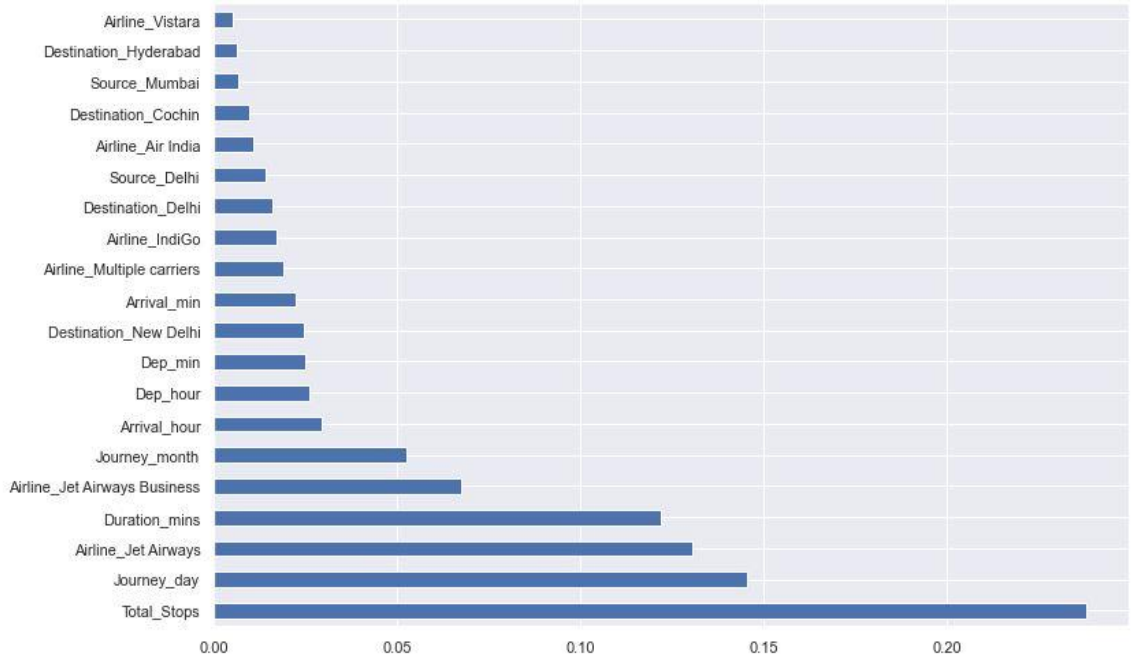
```
ExtraTreesRegressor()
```

```
print(selection.feature_importances_)
```

```
[2.37760854e-01 1.45654852e-01 5.27378895e-02 2.63484751e-02
 2.52228922e-02 2.93346507e-02 2.22870206e-02 1.21812824e-01
 1.08242962e-02 2.02376034e-03 1.73084252e-02 1.30623103e-01
 6.74541759e-02 1.91913938e-02 8.43092991e-04 2.69640392e-03
 1.04551889e-04 5.41192810e-03 6.24148047e-05 4.18400989e-04
 1.43836103e-02 3.27274008e-03 6.87783246e-03 9.87352803e-03
 1.59994405e-02 6.28089648e-03 5.35813253e-04 2.46547335e-02]
```

```
#plot graph of feature importances for better visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```



This bar chart is showing importance of each feature with respect to target feature price. As by looking here it is shown total stops and journey day having high feature importance.

Splitting of data into train and test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((8545, 28), (2137, 28), (8545,), (2137,))
```

Here dataset is divided into train and test part at the ratio of 20% .test part is used for evaluation of model.

Machine learning Model Building

```
from sklearn.linear_model import LinearRegression
LR= LinearRegression()
LR.fit(X_train, y_train)
```

```
LinearRegression()
```

```
from sklearn.tree import DecisionTreeRegressor
DTR= DecisionTreeRegressor()
DTR.fit(X_train, y_train)
```

```
DecisionTreeRegressor()
```

```
from sklearn.ensemble import RandomForestRegressor
RFR= RandomForestRegressor()
RFR.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

Here following three regression algorithms are used:

- Linear Regression
- Decision Tree Regression
- Random Forest Regression

Data training phase is also done here.

Predictions

Selecting values for predictions

```
Total_Stops                                    1
Journey_day                                    6
Journey_month                                  5
Dep_hour                                       7
Dep_min                                       30
Arrival_hour                                  19
Arrival_min                                    0
Duration_mins                                690
Airline_Air India                              0
Airline_GoAir                                  0
Airline_IndiGo                                 0
Airline_Jet Airways                            0
Airline_Jet Airways Business                   0
Airline_Multiple carriers                      1
Airline_Multiple carriers Premium economy      0
Airline_SpiceJet                               0
Airline_Trujet                                 0
Airline_Vistara                                0
Airline_Vistara Premium economy                0
Source_Chennai                                 0
Source_Delhi                                   1
Source_Kolkata                                 0
Source_Mumbai                                  0
Destination_Cochin                             1
Destination_Delhi                              0
Destination_Hyderabad                          0
Destination_Kolkata                            0
Destination_New Delhi                          0
```

Here above 5th row of test data is selected to build predictions.

Building predictions

```
arr=np.array(X_test.iloc[5],ndmin=2)
```

```
LR.predict(arr)
```

```
array([11188.30128002])
```

```
DTR.predict(arr)
```

```
array([9554.5])
```

```
RFR.predict(arr)
```

```
array([9583.40166667])
```

```
y_test.iloc[5]
```

```
10529
```

Here are predictions of three models and actual value.so there is difference of almost 1000 exists here.

Accuracy scores of machine learning models

```
LR.score(X_test, y_test)
```

```
0.6196858941274607
```

```
DTR.score(X_test, y_test)
```

```
0.7775633933630828
```

```
RFR.score(X_test, y_test)
```

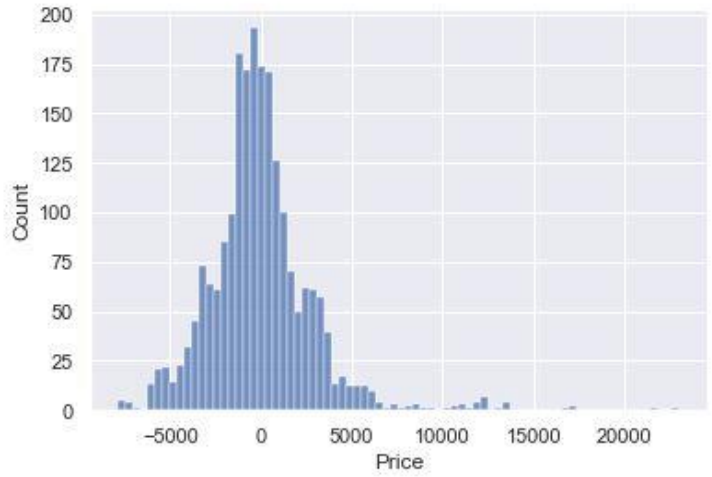
```
0.8195056130519595
```

Here are accuracy scores of three different models.

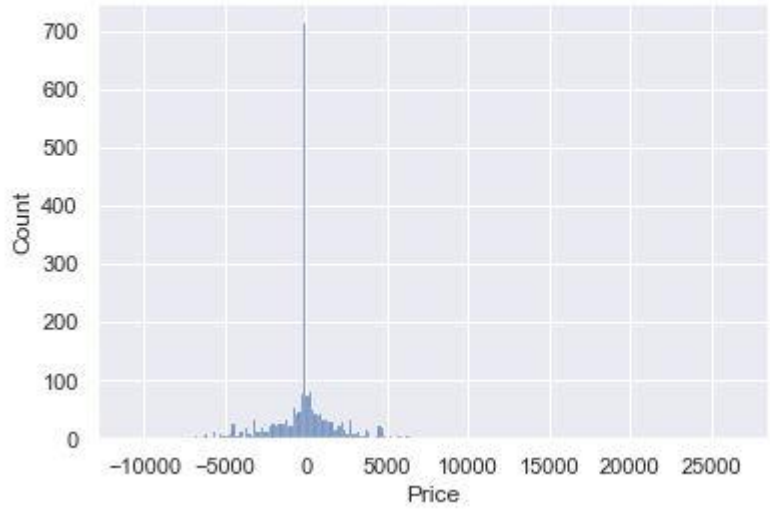
Random Forest Regression is model with best accuracy.

Visualizing difference between actual and predictions using histogram

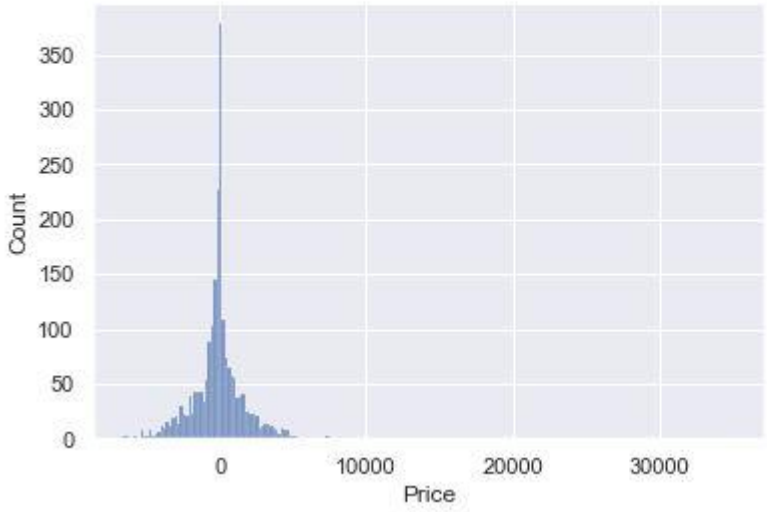
```
LR_pred=LR.predict(X_test)
sns.histplot(y_test-LR_pred)
plt.show()
```



```
DTR_pred=DTR.predict(X_test)
sns.histplot(y_test-DTR_pred)
plt.show()
```




```
RFR_pred=RFR.predict(X_test)
sns.histplot(y_test-RFR_pred)
plt.show()
```



As by comparing above three histograms it is clear that Random Forest Regression is performing very well.

Mean Absolute Error

```
from sklearn.metrics import mean_absolute_error
print('MAE:',mean_absolute_error(y_test, LR_pred))
```

```
MAE: 1972.9471133405893
```

```
from sklearn.metrics import mean_absolute_error
print('MAE:', mean_absolute_error(y_test, DTR_pred))
```

```
MAE: 1294.1262283575106
```

```
from sklearn.metrics import mean_absolute_error
print('MAE:', mean_absolute_error(y_test, RFR_pred))
```

```
MAE: 1155.0829385438042
```

Here is Mean Absolute Error of three models .It shows that Random Forest Regression has very less MAE error which shows that it is very accurate.

Root Mean Square Error

```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,LR_pred)))
```

```
RMSE: 2863.6260530482887
```

```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,DTR_pred)))
```

```
RMSE: 2190.020694162644
```

```
from sklearn.metrics import mean_squared_error
print('RMSE:', np.sqrt(mean_squared_error(y_test,RFR_pred)))
```

```
RMSE: 1972.7721960654
```

Here are root mean square errors of three models.

As root mean square error of random forest regression is very less than other models so it also shows that it performing very well.

Preparing dataset for classification

Making a new categorical feature for performing classification

```
data_train['Price_condition']=(train_data['Price']>=train_data['Price'].median()).astype(int)
data_train['Price_condition']
```

```
0        0
1        0
2        1
3        0
4        1
        ..
10678    0
10679    0
10680    0
10681    1
10682    1
```

Here a new feature is developed by taking median as threshold of price so above median is considered as expensive and below median is considered as cheaper than above median is converted to 1 and below median is converted to 0.

Preparing independent data by Removing two features

```
X=data_train.drop(['Price_condition','Price'],axis=1)
```

As price condition is targeted feature so it is removed with its parent feature price

Preparing dependant feature

```
y=data_train['Price_condition']
y
```

```
0        0
1        0
2        1
3        0
4        1
        ..
10678    0
10679    0
10680    0
10681    1
10682    1
```

Splitting data into train and test

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

Here above 20 % data is splatted as test data and 80% data is reserved for training purpose.

Building classification models.

```
from sklearn.tree import DecisionTreeClassifier
DTC= DecisionTreeClassifier()
DTC.fit(X_train,y_train)
```

```
DecisionTreeClassifier()
```

```
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

Here two classification models are used which are following:

- Decision Tree Classification
- Random Forest Classification

Build predictions

```
arr=np.array(X_test.iloc[10],ndmin=2)
```

```
DTC.predict(arr)
```

```
array([0])
```

```
RFC.predict(arr)
```

```
array([0])
```

Here are predictions of two classification models on the basis of row 10 of test data.so both giving same result.

Evaluating classification models

Accuracy score

```
DTC.score(X_test,y_test)
```

```
0.9218530650444549
```

```
RFC.score(X_test,y_test)
```

```
0.9284043051006083
```

As both models has a high and same accuracy rate.

F1_score

```
from sklearn import metrics
score=metrics.accuracy_score(y_test,y_pred)
pre=metrics.precision_score(y_test, y_pred)
recall=metrics.recall_score(y_test, y_pred)
f1_score=2*pre*recall/(pre+recall)
print("F1_score:",f1_score)
```

```
Accuracy: 0.9218530650444549
Precision: 0.9242009132420091
Recall: 0.9233576642335767
F1_score: 0.923779096303058
```

```
from sklearn import metrics
y_pred=RFC.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test,y_pred))
print("Precision:",metrics.precision_score(y_test,y_pred))
print("Recall:",metrics.recall_score(y_test,y_pred))

from sklearn import metrics
score=metrics.accuracy_score(y_test,y_pred)
pre=metrics.precision_score(y_test, y_pred)
recall=metrics.recall_score(y_test, y_pred)
f1_score=2*pre*recall/(pre+recall)
print("F1_score:",f1_score)
```

```
Accuracy: 0.9284043051006083
Precision: 0.9206066012488849
Recall: 0.9416058394160584
F1_score: 0.9309878213802436
```

As by examining F1 score it is clear that Random Forest Classification has better F1_score

Classification report

```
from sklearn.metrics import classification_report
y_pred=DTC.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92      1041
           1       0.92      0.92      0.92      1096

    accuracy                           0.92      2137
   macro avg       0.92      0.92      0.92      2137
weighted avg       0.92      0.92      0.92      2137
```

```
from sklearn.metrics import classification_report
y_pred=RFC.predict(X_test)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      1041
           1       0.92      0.94      0.93      1096

    accuracy                           0.93      2137
   macro avg       0.93      0.93      0.93      2137
weighted avg       0.93      0.93      0.93      2137
```

Here this classification report shows precision recall and F1_score of each category

So in both models category has same F1_score

Conclusions:

On the basis of analysis and visualization these are conclusions.

Missing values

There are only two missing values in dataset so that is why two rows are removed.

Feature Engineering:

There are seven new features which are created in feature engineering phase.

Relationships:

Source and journey month has direct flight prices.

Destination and total stops have reverse relationship with flight prices

Best model:

Random Forest Regression is my best choice because as it is best performing model due to following reasons:

- Accuracy of random forest regression is more than other models
- Mean Absolute Error and Root Mean Square Error is very less than other

Reference:

Dataset at kaggle:

https://www.kaggle.com/datasets/nikhilmittal/flight-fare-prediction-mh

Article on this topic ''How To Perform Exploratory Data Analysis -A Guide for Beginners''

https://www.analyticsvidhya.com/blog/2021/08/how-to-perform-exploratory-data-analysis-a-guide-for-beginners/

Machine learning practice:

https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/code

Exploratory data analysis:

https://medium.com/@hammad173216/my-first-exploratory-data-analysis-project-in-python-on-sales-analysis-2e48880d4094