19CSE304 – Foundation of Data Science

Case Study Document

Group 14 - **Flight Fare Prediction**

Roll number	Name
CB.EN.U4CSE19602	Adarsh jayan
CB.EN.U4CSE19615	Dev divyendh
CB.EN.U4CSE19626	Karthik shriram
CB.EN.U4CSE19649	Sri chakra teja

IMPORTING AND LOADING DATASET

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

```
[2] from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[3] df = pd.read_excel('/content/drive/MyDrive/flight.xlsx')
df.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

DESCRIPTIVE STATISTICS

```
df.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
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

```
[7] df.describe()
```

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000

PREPROCESSING

```
[8] df.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
```

```
[9] df.dropna(inplace=True)
```

```
[10] df.drop(columns='Additional_Info',axis=1,inplace=True)
```

```
[11] df.columns
```

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops', 'Price'],
      dtype='object')
```

FEATURE ENGINEERING

Converting the column [Duration]

```
df['Duration'].astype(str)
d=list(df['Duration'])
duration=[]
def dur():
  for i in range(len(d)):
    z=d[i]
    if len(z.split(sep=' '))!=2:
      if 'h' in z:
        z=z.split(sep='h')[0]
        x=int(z)
        duration.append(60*x)
      else:
        z=z.split(sep='m')[0]
        x=int(z)
        duration.append(x)
    else:
      h=z.split(sep=' ')[0]
      h=h.split(sep='h')[0]
      h=int(h)
      m=z.split(sep=' ')[1]
      m=m.split(sep='m')[0]
      m=int(m)
      duration.append(60*h+m)
```

```
[13] type(d)
```

```
list
```

```
[14] dur()
     df['duration(mins)']=duration
```

```
[13] type(d)

     list
```

```
[14] dur()
     df['duration(mins)']=duration
```

```
[15] df.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Price	duration(mins)
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	3897	170
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	7662	445
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	13882	1140
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	6218	325
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	13302	285

```
[16] df.drop(columns='Duration',axis=1,inplace=True)
```

```
[17] df['Total_Stops'].value_counts()

     1 stop      5625
     non-stop    3491
     2 stops     1520
     3 stops       45
```

Converting the column [total stops]

```
df.loc[(df ['Total_Stops']=='non-stop'), "Number_of_stops"]=0
df.loc[(df ['Total_Stops']=='1 stop'), "Number_of_stops"]=1
df.loc[(df ['Total_Stops']=='2 stops'), "Number_of_stops"]=2
df.loc[(df ['Total_Stops']=='3 stops'), "Number_of_stops"]=3
df.loc[(df ['Total_Stops']=='4 stops'), "Number_of_stops"]=4
```

Converting the column [Date of journey]

```
[19] df['Date_of_Journey']=pd.to_datetime(df['Date_of_Journey'])
```

```
[20] df['date'] = df['Date_of_Journey'].dt.day
```

```
[21] df['month'] = df['Date_of_Journey'].dt.month
```

```
[22] df['year'] = df['Date_of_Journey'].dt.year
```

Converting the columns [dep time, arrival time]

```
[23] df['Dep_Time']=pd.to_datetime(df['Dep_Time'])
```

```
[24] df['Arrival_Time']=pd.to_datetime(df['Arrival_Time'])
```

```
[25] df['dep(hour)']=df['Dep_Time'].dt.hour
```

```
[26] df['dep(minute)']=df['Dep_Time'].dt.minute
```

```
[27] df['arr(hour)']=df['Arrival_Time'].dt.hour
```

```
[28] df['arr(min)']=df['Arrival_Time'].dt.minute
```

```
[29] df.drop(columns=['Date_of_Journey','Dep_Time','Arrival_Time','Total_Stops'],axis=1,inplace=True)
```

Updated dataframe

```
df.head()
```

	Airline	Source	Destination	Route	Price	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
0	IndiGo	Banglore	New Delhi	BLR → DEL	3897	170	0.0	24	3	2019	22	20	1	10
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7662	445	2.0	5	1	2019	5	50	13	15
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	13882	1140	2.0	6	9	2019	9	25	4	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	6218	325	1.0	5	12	2019	18	5	23	30
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	13302	285	1.0	3	1	2019	16	50	21	35

PANDAS FUNCTIONS

Sort

```
df.sort_values(by=['Number_of_stops'],ascending=False)
```

	Airline	Source	Destination	Route	Price	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
9182	Air India	Banglore	New Delhi	BLR → CCU → BBI → HYD → VGA → DEL	17686	1770	4.0	3	1	2019	5	50	11	20
7031	Air India	Kolkata	Banglore	CCU → GAU → IMF → DEL → BLR	14015	805	3.0	5	12	2019	9	50	23	15
3945	Air India	Mumbai	Hyderabad	BOM → BLR → CCU → BBI → HYD	14260	1175	3.0	3	12	2019	16	50	12	25
5996	Air India	Kolkata	Banglore	CCU → GAU → IMF → DEL → BLR	15145	1040	3.0	24	3	2019	5	55	23	15
5947	Air India	Banglore	New Delhi	BLR → HBX → BOM → AMD → DEL	10573	715	3.0	3	3	2019	12	0	23	55
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

```
df.sort_values(by=['date', 'month','year'])
```

	Airline	Source	Destination	Route	Price	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	13302	285	1.0	3	1	2019	16	50	21	35
7	Jet Airways	Banglore	New Delhi	BLR → BOM → DEL	22270	1265	1.0	3	1	2019	8	0	5	5
56	Air India	Banglore	New Delhi	BLR → BOM → AMD → DEL	17345	905	2.0	3	1	2019	8	50	23	55
123	Air India	Delhi	Cochin	DEL → BOM → COK	27430	1215	1.0	3	1	2019	23	0	19	15
268	Air India	Chennai	Kolkata	MAA → CCU	19630	135	0.0	3	1	2019	11	40	13	55

Group by

```
print(df.groupby(['Airline']).count())
```

```
                                   Source  Destination  ...  arr(hour)  arr(min)
Airline                                                 ...
Air Asia                              319          319  ...        319       319
Air India                            1751         1751  ...       1751      1751
GoAir                                 194          194  ...        194       194
IndiGo                               2053         2053  ...       2053      2053
Jet Airways                          3849         3849  ...       3849      3849
Jet Airways Business                    6            6  ...          6         6
Multiple carriers                    1196         1196  ...       1196      1196
Multiple carriers Premium economy      13           13  ...         13        13
SpiceJet                              818          818  ...        818       818
Trujet                                  1            1  ...          1         1
Vistara                               479          479  ...        479       479
Vistara Premium economy                 3            3  ...          3         3
```

Inference: We were able to find out the number of journeys that were taken through each kind of Airline.

```
print(df.groupby(['Source','Destination']).count())
```

```
                      Airline  Route  Price  ...  dep(minute)  arr(hour)  arr(min)
Source   Destination                         ...
Banglore Delhi           1265   1265   1265  ...         1265       1265      1265
         New Delhi        932    932    932  ...          932        932       932
Chennai  Kolkata          381    381    381  ...          381        381       381
Delhi    Cochin          4536   4536   4536  ...         4536       4536      4536
Kolkata  Banglore        2871   2871   2871  ...         2871       2871      2871
Mumbai   Hyderabad        697    697    697  ...          697        697       697

[6 rows x 12 columns]
```

Inference: We were able to count the number of travels based on Sourse and Destination pairs

Pivot table

```
table = pd.pivot_table(df, values=['duration(mins)', 'Number_of_stops'], index=['Source','Destination'],
                       aggfunc={'duration(mins)': np.mean,'Number_of_stops': np.mean})
table
```

Source	Destination	Number_of_stops	duration(mins)
Banglore	Delhi	0.000000	171.695652
	New Delhi	0.790773	654.077253
Chennai	Kolkata	0.000000	139.619423
Delhi	Cochin	1.209436	817.852734
Kolkata	Banglore	0.860676	747.248346
Mumbai	Hyderabad	0.157819	191.714491

Inference: With the help of pivot table, we were able to find the mean of duration and mean of number of stops w.r.t Source and destination.

Melt

```
table = pd.melt(df, id_vars=['Airline'], value_vars=['Source'],
        var_name='source/dest', value_name='place')
table
```

	Airline	source/dest	place
0	IndiGo	Source	Banglore
1	Air India	Source	Kolkata
2	Jet Airways	Source	Delhi
3	IndiGo	Source	Kolkata
4	IndiGo	Source	Banglore
...	...	...	...
10677	Air Asia	Source	Kolkata
10678	Air India	Source	Kolkata
10679	Jet Airways	Source	Banglore
10680	Vistara	Source	Banglore
10681	Air India	Source	Delhi

10682 rows × 3 columns

Inference: With the help of melt table, we were able to conclude the airlines and their source(starting point) of travel

Crosstab

```
pd.crosstab(df['Airline'],df['Source'])
```

Source	Banglore	Chennai	Delhi	Kolkata	Mumbai
Airline					
Air Asia	89	0	80	150	0
Air India	332	25	746	512	136
GoAir	93	0	76	25	0
IndiGo	523	184	705	445	196
Jet Airways	788	0	1586	1256	219
Jet Airways Business	4	0	2	0	0
Multiple carriers	0	0	1196	0	0
Multiple carriers Premium economy	0	0	13	0	0
SpiceJet	181	128	87	300	122
Trujet	0	0	0	0	1
Vistara	185	43	45	183	23
Vistara Premium economy	2	1	0	0	0

Inference: With the help of crosstab, we were able to find the count of travels whose source is one of [Bangalore, chennai, Delhi, Kolkata, Mumbai] w.r.t Unique Airline.

```
pd.crosstab(df['Airline'],df['Destination'])
```

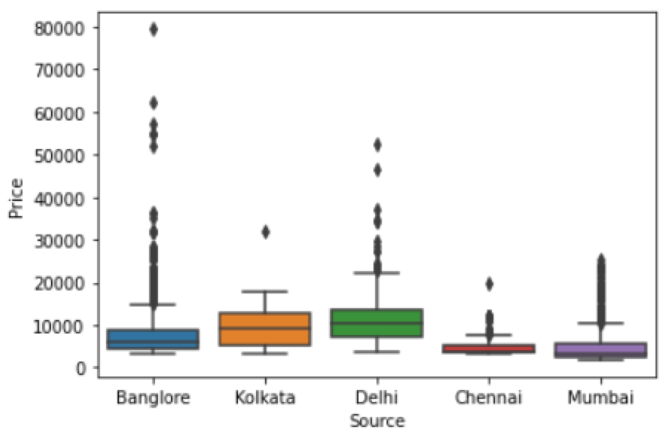
Destination / Airline	Banglore	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
Air Asia	150	80	71	0	0	18
Air India	512	746	120	136	25	212
GoAir	25	76	69	0	0	24
IndiGo	445	705	366	196	184	157
Jet Airways	1256	1586	370	219	0	418
Jet Airways Business	0	2	0	0	0	4
Multiple carriers	0	1196	0	0	0	0
Multiple carriers Premium economy	0	13	0	0	0	0
SpiceJet	300	87	137	122	128	44
Trujet	0	0	0	1	0	0
Vistara	183	45	131	23	43	54
Vistara Premium economy	0	0	1	0	1	1

Inference: With the help of crosstab, we were able to find the count of travels whose destination is one of [Bangalore, Cochin, Delhi, New Delhi, Kolkata, Hyderabad] w.r.t Unique Airline.

OUTLIER DETECTION

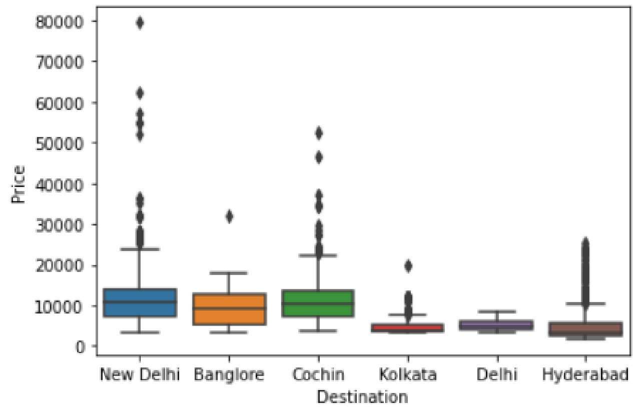
```
sns.boxplot(x=df['Source'],y=df['Price'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff67d6ecc50>
```



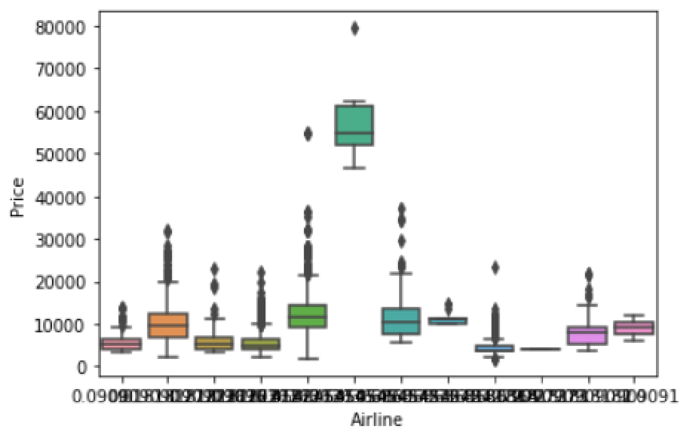
```python
sns.boxplot(x=df['Destination'],y=df['Price'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff67d219550>



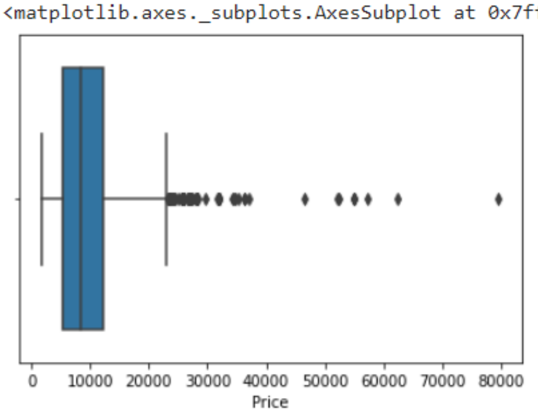

```python
sns.boxplot(x=df['Airline'],y=df['Price'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff671b91990>




```python
sns.boxplot(x=df['Price'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7ff65d670690>



```
q1=df['Price'].quantile(0.25)
q3=df['Price'].quantile(0.75)
IQR=q3-q1
display(IQR)
```

```
7096.0
```

```
lower_limit = q1 - 1.5*IQR
display(lower_limit)
```

```
-5367.0
```

```
upper_limit = q3 + 1.5*IQR
display(upper_limit)
```

```
23017.0
```

```
range = df['Price'].max() - df['Price'].min()
display('max:',df['Price'].max())
display('min:',df['Price'].min())
display('range:',range)
```

```
'max:'
79512
'min:'
1759
'range:'
77753
```

```
df['Price'].mean()
```

```
9087.21456656057
```

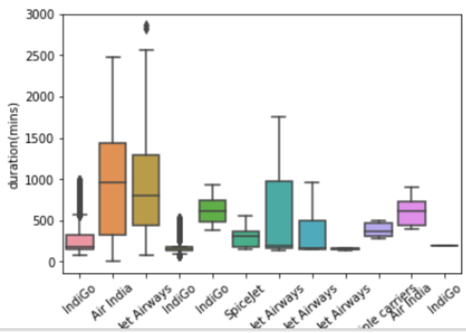
```
df.describe()
```

	Price	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
count	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.0	10682.000000	10682.000000	10682.000000	10682.000000
mean	9087.214567	643.020502	0.824190	12.682925	5.534731	2019.0	12.491013	24.409287	13.349186	24.690601
std	4611.548810	507.830133	0.675229	8.803800	2.987626	0.0	5.748820	18.767801	6.859317	16.506808
min	1759.000000	5.000000	0.000000	3.000000	1.000000	2019.0	0.000000	0.000000	0.000000	0.000000
25%	5277.000000	170.000000	0.000000	5.000000	3.000000	2019.0	8.000000	5.000000	8.000000	10.000000
50%	8372.000000	520.000000	1.000000	6.000000	5.000000	2019.0	11.000000	25.000000	14.000000	25.000000
75%	12373.000000	930.000000	1.000000	21.000000	6.000000	2019.0	18.000000	40.000000	19.000000	35.000000
max	79512.000000	2860.000000	4.000000	27.000000	12.000000	2019.0	23.000000	55.000000	23.000000	55.000000

VISUALISATION

```
g=sns.boxplot(x=df['Airline'],y=df['duration(mins)'])
g.set_xticklabels(labels=df['Airline'],rotation=40)
```

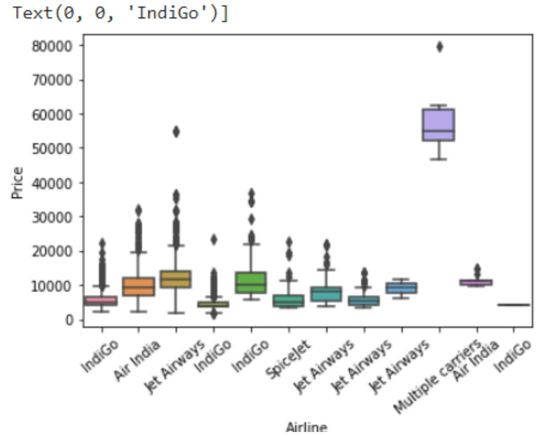
```
[Text(0, 0, 'IndiGo'),
 Text(0, 0, 'Air India'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'IndiGo'),
 Text(0, 0, 'IndiGo'),
 Text(0, 0, 'SpiceJet'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Multiple carriers'),
 Text(0, 0, 'Air India'),
 Text(0, 0, 'IndiGo')]
```



Inference from the graphs is based on outlier detection from the following graphs
1) Boxplot between duration and airline----> For Indigo and Jet Airways we can see there are values that are deviating from the relationship

```
g=sns.boxplot(x=df['Airline'],y=df['Price'])
g.set_xticklabels(labels=df['Airline'],rotation=40)
```

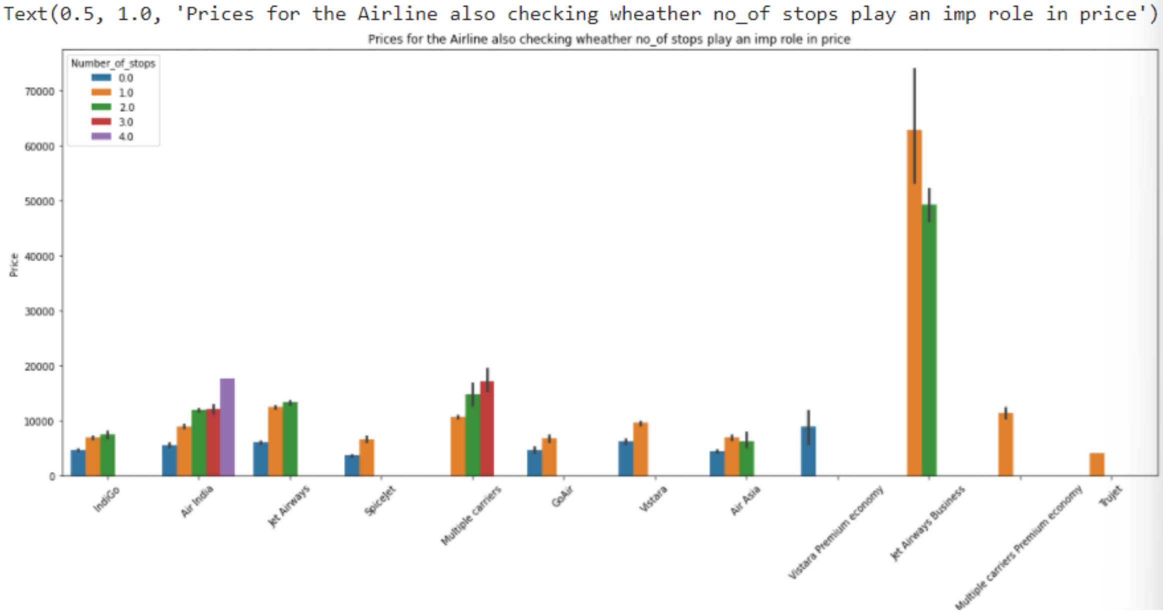
```
[Text(0, 0, 'IndiGo'),
 Text(0, 0, 'Air India'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'IndiGo'),
 Text(0, 0, 'IndiGo'),
 Text(0, 0, 'SpiceJet'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'Multiple carriers'),
 Text(0, 0, 'Air India'),
 Text(0, 0, 'IndiGo')]
```



Inference For Outlier detection for the following graph shows that most of the relationships are having outliers, in the sense price can be high for different routes or places

```
plt.figure(figsize=(20,8))
g=sns.barplot(x=df['Airline'],y=df['Price'],hue=df['Number_of_stops'])
plt.xticks(rotation=45)
#g.set_xticklabels(labels=df['Airline'],rotation=40)
g.set_title('Prices for the Airline also checking wheather no_of stops play an imp role in price')
```

```
Text(0.5, 1.0, 'Prices for the Airline also checking wheather no_of stops play an imp role in price')
```



Inference from the bar plot here is to find whether Number of stops causes whether increase or either decrease in the price of the fare

Here we are finding the number of values for a brief idea

```
df['Source'].value_counts()
```

```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

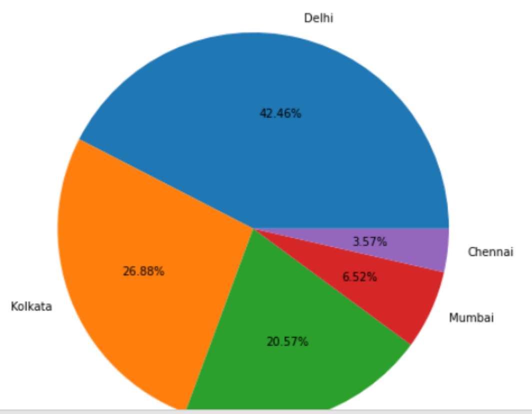
```
df['Destination'].value_counts()
```

```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,2])
ax.axis('equal')
data=[4536,2871,2197,697,381]
diff=('Delhi','Kolkata','Banglore','Mumbai','Chennai')
ax.pie(data, labels =diff,autopct='%1.2f%%')
plt.title('percentage of flights that are present in that particular source ')
```

```
Text(0.5, 1.0, 'percentage of flights that are present in that particular source ')
```

percentage of flights that are present in that particular source

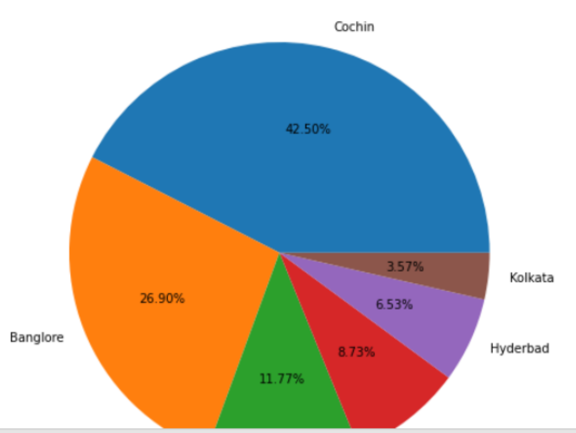


In this pie plot we are trying to find percentage of source locations, basically in the sense we are trying to find the number of flights that are present in a particular source

```
fig = plt.figure()
ax = fig.add_axes([0,0,1,2])
ax.axis('equal')
data=[4536,2871,1256,932,697,381]
diff=('Cochin','Banglore','Delhi','New Delhi','Hyderbad','Kolkata')
ax.pie(data, labels =diff,autopct='%1.2f%%')
plt.title('percentage of airline comapnies')
```

```
Text(0.5, 1.0, 'percentage of airline comapnies')
```

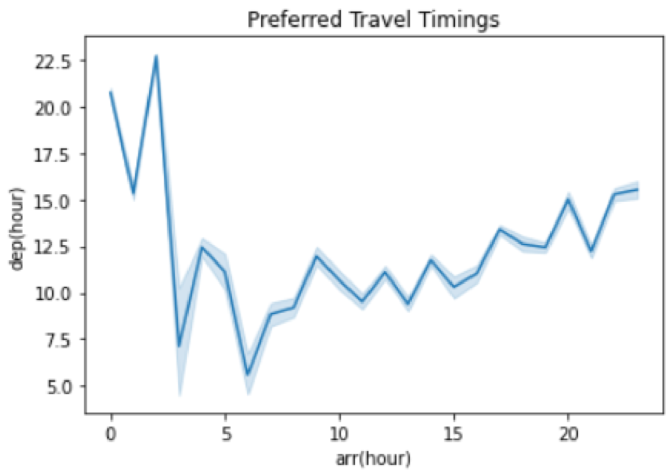
percentage of airline comapnies



The pie plot is as before is used to find the percentage of flights that are traveling to various destinations, this pie plot can also be used to find the tourist locations

```
g=sns.lineplot(x=df['arr(hour)'],y=df['dep(hour)'])
g.set_title('Preferred Travel Timings')
```

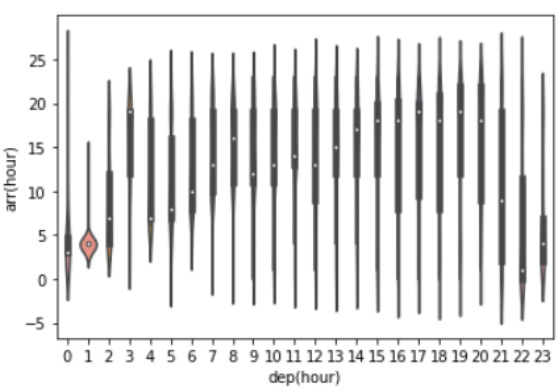
```
Text(0.5, 1.0, 'Preferred Travel Timings')
```



The **inference** from the following plot is to check the most active flight timings , basically the timings the passengers prefer the most to travel

+ Code + Text

```
sns.violinplot(x=df['dep(hour)'],y=df['arr(hour)'],data=df)
plt.show()
```



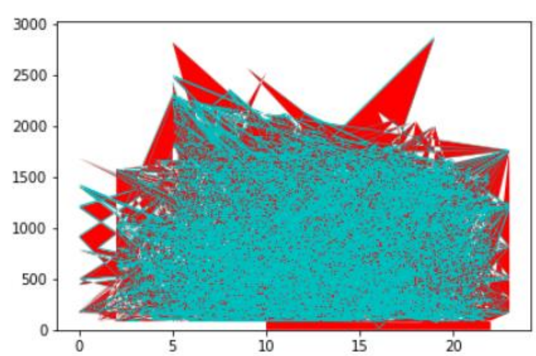
In this plot we basically try to find for the range of the arrival times for a particular departure time E.g.: for 12 is in the range from 3 to 24 , but most occurring arrival time range is from 10 to 20

```
plt.figure(figsize=(15,10))
sns.scatterplot(df['Price'],df['duration(mins)'],hue=df['Number_of_stops'])
plt.legend(bbox_to_anchor=(1, 1), loc=2)
plt.show()
```

```
usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the followir
  FutureWarning
```

This graph basically plots we can see many relations for example the relation between duration and the stops whether it increases the time or not, also the most preferred price ranges

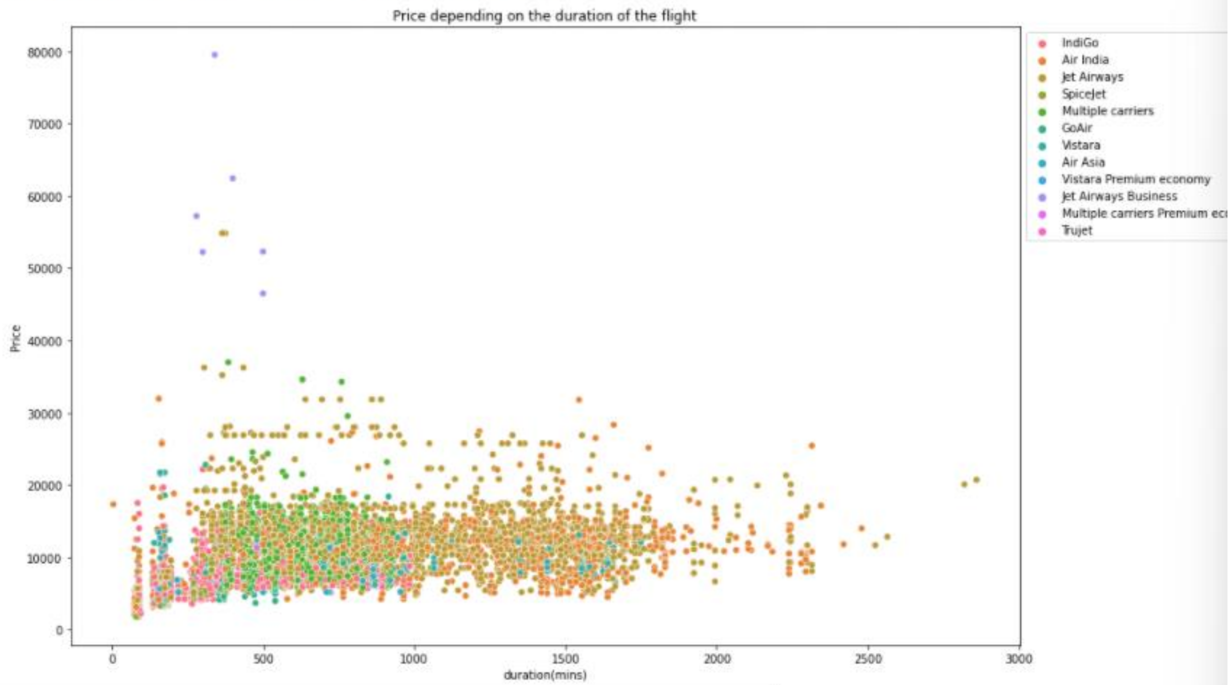
```
plt.stackplot(df['dep(hour)'],df['duration(mins)'],df['arr(hour)'],
              colors =['r', 'c'])
plt.show()
```



This graph performing stack plot for the departure hour and arrival hour based on the duration

```
plt.figure(figsize=(15,10))
g=sns.scatterplot(x=df['duration(mins)'],y=df['Price'],hue=df['Airline'])
g.set_title('Price depending on the duration of the flight')
plt.legend(bbox_to_anchor=(1, 1), loc=2)
```

```
<matplotlib.legend.Legend at 0x7ff67d5fa950>
```



In this plot We can see the airlines that provide the best price for the given price and duration

ENCODING

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()

df['Airline'] = label_encoder.fit_transform(df['Airline'])
df['Source'] = label_encoder.fit_transform(df['Source'])
df['Destination'] = label_encoder.fit_transform(df['Destination'])
```

```
df.head()
```

	Airline	Source	Destination	Route	Price	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
0	3	0	5	BLR → DEL	3897	170	0.0	24	3	2019	22	20	1	10
1	1	3	0	CCU → IXR → BBI → BLR	7662	445	2.0	5	1	2019	5	50	13	15
2	4	2	1	DEL → LKO → BOM → COK	13882	1140	2.0	6	9	2019	9	25	4	25
3	3	3	0	CCU → NAG → BLR	6218	325	1.0	5	12	2019	18	5	23	30
4	3	0	5	BLR → NAG → DEL	13302	285	1.0	3	1	2019	16	50	21	35

SCALING

MIn-Max Scaling

```
[62] df_min_max_scaled = df.copy()
     df_notneeded = pd.DataFrame(df,columns=['Route','Price'])
     df_min_max_scaled.drop(['Route','Price'],axis=1,inplace=True)
     display(df_min_max_scaled.head())
     display(df_notneeded.head())
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	year	dep(hour)	dep(minute)	arr(hour)	arr(min)
0	3	0	5	170	0.0	24	3	2019	22	20	1	10
1	1	3	0	445	2.0	5	1	2019	5	50	13	15
2	4	2	1	1140	2.0	6	9	2019	9	25	4	25
3	3	3	0	325	1.0	5	12	2019	18	5	23	30
4	3	0	5	285	1.0	3	1	2019	16	50	21	35

	Route	Price
0	BLR → DEL	3897
1	CCU → IXR → BBI → BLR	7662
2	DEL → LKO → BOM → COK	13882
3	CCU → NAG → BLR	6218
4	BLR → NAG → DEL	13302

```
] #MIN-MAX scaling
  for column in df_min_max_scaled.columns:
    df_min_max_scaled[column] = (df_min_max_scaled[column] - df_min_max_scaled[column].min()) / (df_min_max_scaled[column].max() - df_min_max_scaled[column].min())
```

```
result = pd.concat([df_min_max_scaled, df_notneeded], axis=1)
result.drop(columns=['year'],axis=1,inplace=True)
result.sample(5)
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price
4139	0.363636	0.75	0.0	0.255692	0.25	0.083333	0.454545	0.869565	0.000000	0.347826	0.272727	CCU → BOM → BLR	14388
9022	0.363636	0.75	0.0	0.257443	0.25	0.875000	0.363636	0.347826	0.454545	0.869565	0.818182	CCU → BOM → BLR	14571
8226	0.545455	0.50	0.2	0.294221	0.25	0.750000	0.181818	0.173913	0.818182	0.782609	0.909091	DEL → BOM → COK	11098
4586	0.363636	0.00	1.0	0.150613	0.25	0.625000	0.181818	0.608696	0.090909	0.913043	0.363636	BLR → BOM → DEL	16946
6901	0.363636	0.00	1.0	0.380035	0.25	0.000000	0.000000	0.608696	0.090909	0.347826	0.272727	BLR → BOM → DEL	22270

```
] temp=result
```

VECTORIZATION

```
import nltk
import re
import string
from wordcloud import WordCloud,STOPWORDS
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
stop_words = ['→']
```

```
result.head()
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price
0	0.272727	0.00	1.0	0.057793	0.00	0.875000	0.181818	0.956522	0.363636	0.043478	0.181818	BLR → DEL	3897
1	0.090909	0.75	0.0	0.154116	0.50	0.083333	0.000000	0.217391	0.909091	0.565217	0.272727	CCU → IXR → BBI → BLR	7662
2	0.363636	0.50	0.2	0.397548	0.50	0.125000	0.727273	0.391304	0.454545	0.173913	0.454545	DEL → LKO → BOM → COK	13882
3	0.272727	0.75	0.0	0.112084	0.25	0.083333	1.000000	0.782609	0.090909	1.000000	0.545455	CCU → NAG → BLR	6218
4	0.272727	0.00	1.0	0.098074	0.25	0.000000	0.000000	0.695652	0.909091	0.913043	0.636364	BLR → NAG → DEL	13302

```
df=temp
df
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price
0	0.272727	0.00	1.0	0.057793	0.00	0.875000	0.181818	0.956522	0.363636	0.043478	0.181818	BLR → DEL	3897
1	0.090909	0.75	0.0	0.154116	0.50	0.083333	0.000000	0.217391	0.909091	0.565217	0.272727	CCU → IXR → BBI → BLR	7662
2	0.363636	0.50	0.2	0.397548	0.50	0.125000	0.727273	0.391304	0.454545	0.173913	0.454545	DEL → LKO → BOM → COK	13882
3	0.272727	0.75	0.0	0.112084	0.25	0.083333	1.000000	0.782609	0.090909	1.000000	0.545455	CCU → NAG → BLR	6218
4	0.272727	0.00	1.0	0.098074	0.25	0.000000	0.000000	0.695652	0.909091	0.913043	0.636364	BLR → NAG → DEL	13302

```
categorical=df
df
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price
0	0.272727	0.00	1.0	0.057793	0.00	0.875000	0.181818	0.956522	0.363636	0.043478	0.181818	BLR → DEL	3897
1	0.090909	0.75	0.0	0.154116	0.50	0.083333	0.000000	0.217391	0.909091	0.565217	0.272727	CCU → IXR → BBI → BLR	7662
2	0.363636	0.50	0.2	0.397548	0.50	0.125000	0.727273	0.391304	0.454545	0.173913	0.454545	DEL → LKO → BOM → COK	13882
3	0.272727	0.75	0.0	0.112084	0.25	0.083333	1.000000	0.782609	0.090909	1.000000	0.545455	CCU → NAG → BLR	6218
4	0.272727	0.00	1.0	0.098074	0.25	0.000000	0.000000	0.695652	0.909091	0.913043	0.636364	BLR → NAG → DEL	13302
...	...	...	...	...	...	...	...	...	...	...	...	...	...

```
categorical['Route1']=categorical['Route'].str.split('→').str[0]
categorical['Route2']=categorical['Route'].str.split('→').str[1]
categorical['Route3']=categorical['Route'].str.split('→').str[2]
categorical['Route4']=categorical['Route'].str.split('→').str[3]
categorical['Route5']=categorical['Route'].str.split('→').str[4]
categorical
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price	Route1	Route2	Route3	Route4	Route5
0	0.272727	0.00	1.0	0.057793	0.00	0.875000	0.181818	0.956522	0.363636	0.043478	0.181818	BLR → DEL	3897	BLR	DEL	NaN	NaN	NaN
1	0.090909	0.75	0.0	0.154116	0.50	0.083333	0.000000	0.217391	0.909091	0.565217	0.272727	CCU → IXR → BBI → BLR	7662	CCU	IXR	BBI	BLR	NaN
2	0.363636	0.50	0.2	0.397548	0.50	0.125000	0.727273	0.391304	0.454545	0.173913	0.454545	DEL → LKO → BOM → COK	13882	DEL	LKO	BOM	COK	NaN
3	0.272727	0.75	0.0	0.112084	0.25	0.083333	1.000000	0.782609	0.090909	1.000000	0.545455	CCU → NAG → BLR	6218	CCU	NAG	BLR	NaN	NaN
4	0.272727	0.00	1.0	0.098074	0.25	0.000000	0.000000	0.695652	0.909091	0.913043	0.636364	BLR → NAG → DEL	13302	BLR	NAG	DEL	NaN	NaN

```
# Applying label encoder
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
```

```
for i in ['Route1', 'Route2', 'Route3', 'Route4', 'Route5']:
    categorical[i]=encoder.fit_transform(categorical[i])
```

```
from sklearn.preprocessing import MinMaxScaler
df=categorical
min_max_scaler = MinMaxScaler()
df[['Route1', 'Route2', 'Route3', 'Route4', 'Route5']] = min_max_scaler.fit_transform(df[['Route1', 'Route2', 'Route3', 'Route4', 'Route5']])
df
```

	Airline	Source	Destination	duration(mins)	Number_of_stops	date	month	dep(hour)	dep(minute)	arr(hour)	arr(min)	Route	Price	Route1	Route2	Route3	Route4	Route5
0	0.272727	0.00	1.0	0.057793	0.00	0.875000	0.181818	0.956522	0.363636	0.043478	0.181818	BLR → DEL	3897	0.00	0.295455	1.000000	1.000000	1.0
1	0.090909	0.75	0.0	0.154116	0.50	0.083333	0.000000	0.217391	0.909091	0.565217	0.272727	CCU → IXR → BBI → BLR	7662	0.50	0.568182	0.034483	0.230769	1.0
2	0.363636	0.50	0.2	0.397548	0.50	0.125000	0.727273	0.391304	0.454545	0.173913	0.454545	DEL → LKO → BOM → COK	13882	0.75	0.727273	0.137931	0.384615	1.0
3	0.272727	0.75	0.0	0.112084	0.25	0.083333	1.000000	0.782609	0.090909	1.000000	0.545455	CCU → NAG → BLR	6218	0.50	0.772727	0.103448	1.000000	1.0
4	0.272727	0.00	1.0	0.098074	0.25	0.000000	0.000000	0.695652	0.909091	0.913043	0.636364	BLR → NAG → DEL	13302	0.00	0.772727	0.275862	1.000000	1.0

HYPOTHESIS TESTING

Z test

Considering the price of flights we can tell that the average price is 9087.2 rs.A sample of 32 flights has an average greater than 9087.2 rs. The standard deviation of the population is 4611.5 rs. At α = 0.05, is there enough evidence to reject the claim.

```
M=df['Price'].mean() #population mean
```

```
[115] df1=df['Price'].sample(32)
```

```
[116] m=df1.mean() #sample mean
```

```
[117] s=df['Price'].std() #population standard deviation
```

```
[118] M
```

```
9087.21456656057
```

```
[119] m
```

```
9794.625
```

```
[120] s
```

```
4611.548810094335
```

```
import numpy as np
import scipy.stats as st
```

```
[125] #H0 : μ = 9087.2   Ha : μ > 9087.2
n = 32
xbar =  m
mu = M
sigma =  s
alpha = 0.05
```

```
[126] z = (xbar-mu)/(sigma/np.sqrt(n))
z
```

```
0.8677600262579092
```

```
[127] p_val=(1-st.norm.cdf(z))*2
p_val
```

```
0.385525717427301
```

```
[128] if(p_val>alpha):
  print("Accept Null Hypothesis")
else:
  print("Reject Null Hypothesis")
```

```
Accept Null Hypothesis
```

T test

Considering the duration of movies we can tell that the average duration is 99.2 minutes.A random sample of 10 movies has an average duration of 106.8 minutes. The standard deviation of the population is 17 minutes. At α = 0.10, is there enough evidence to reject the claim?

```
[129] M=df['Price'].mean() #population mean
```

```
[130] df1=df['Price'].sample(50)
```

```
[131] m=df1.mean() #sample mean
```

```
[136] s=df1.std() #population standard deviation
```

```
[133] M
```

```
9087.21456656057
```

```
[134] m
```

```
9280.3
```

```
[137] s
```

```
4587.423438720526
```

```
[138] import numpy as np
      import scipy.stats as st
```

```
#H0 : μ = 9087.2   Ha : μ < 9087.2
n = 50
degrees_of_freedom = n-1
xbar =  m
mu = M
sigma =  s
alpha = 0.05
```

```
[140] t = (xbar-mu)/(sigma/np.sqrt(n))
      t
```

```
0.2976224478886227
```

```
[143] t_critical=abs(st.t.ppf(0.05/2,degrees_of_freedom))
      t_critical
```

```
2.0095752344892093
```

```
if(t<t_critical):
  print("Accept Null Hypothesis")
else:
  print("Reject Null Hypothesis")
```

```
Accept Null Hypothesis
```

MACHINE LEARNING MODELS

RANDOM FOREST

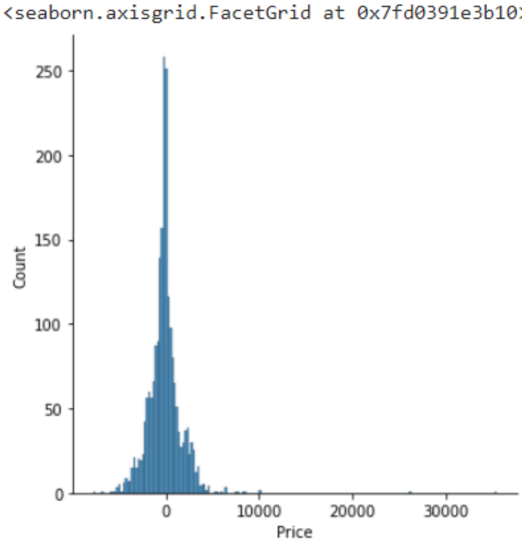
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import GradientBoostingRegressor,RandomForestRegressor
```

```
] rf=RandomForestRegressor()
rf_random=RandomizedSearchCV(estimator=rf,param_distributions=random_grid,cv=3,verbose=2,n_jobs=-1,)

rf_random.fit(X_train,y_train)

# best parameter
rf_random.best_params_
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
{'max_depth': 15, 'max_features': 'sqrt', 'n_estimators': 120}
```

```
#predicting the values
prediction = rf_random.predict(X_test)

#distribution plot between actual value and predicted value
sns.displot(y_test-prediction)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fd0391e3b10>
```




```
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
```

```
r2_score(y_test,prediction)
```

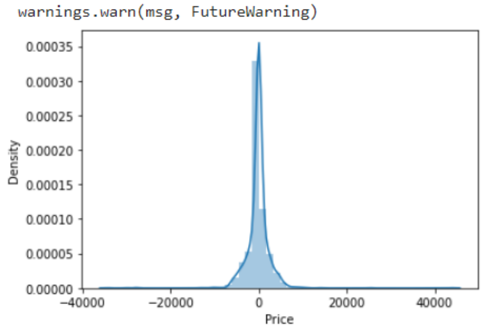
```
0.8313916509762904
```

DECISION TREE

```python
from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error
def predict(ml_model):
    print('Model is: {}'.format(ml_model))
    model= ml_model.fit(X_train,y_train)
    print("Training score: {}".format(model.score(X_train,y_train)))
    predictions = model.predict(X_test)
    print("Predictions are: {}".format(predictions))
    print('\n')
    r2score=r2_score(y_test,predictions)
    print("r2 score is: {}".format(r2score))

    print('MAE:{}'.format(mean_absolute_error(y_test,predictions)))
    print('MSE:{}'.format(mean_squared_error(y_test,predictions)))
    print('RMSE:{}'.format(np.sqrt(mean_squared_error(y_test,predictions))))

    sns.distplot(y_test-predictions)
```

```python
predict(DecisionTreeRegressor())
```

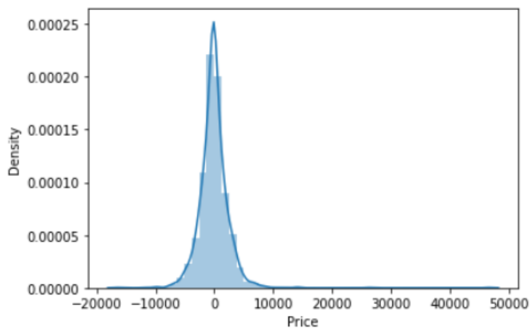
```
Model is: DecisionTreeRegressor()
Training score: 0.9693071159054724
Predictions are: [16840.  5752.  9397. ...  8327. 13339. 14151.]


r2 score is: 0.6438624775100347
MAE:1390.3250194977381
MSE:7679057.319076067
RMSE:2771.1112065516363
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be
  warnings.warn(msg, FutureWarning)
```



KNN REGRESSOR

```
[119] predict(KNeighborsRegressor())
```

```
Model is: KNeighborsRegressor()
Training score: 0.7965252230814815
Predictions are: [16315.   5903.4  8620.   ...  6471.8 11858.4 13167.6]


r2 score is: 0.6934951937961836
MAE:1635.1854000935891
MSE:6608873.894992981
RMSE:2570.773015066282
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated functic
  warnings.warn(msg, FutureWarning)
```




```
120]
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model, metrics
reg = linear_model.LinearRegression()

# train the model using the training sets
reg.fit(X_train, y_train)
```

```
LinearRegression()
```

```
predictions = reg.predict(X_test)
print("Predictions are: {}".format(predictions))
print('\n')
r2score=r2_score(y_test,predictions)
print("r2 score is: {}".format(r2score))
```

```
Predictions are: [12334.26582315 10717.32189507 11323.49598833 ...  9354.24249801
  9979.94578211 10359.5401708 ]


r2 score is: 0.5006668608006783
```