Milestone 2:Data Collection & Preparation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report, confusion_matrix
import warnings
import pickle
from scipy import stats
warnings.filterwarnings('ignore')
plt.style.use('fivethirtyeight')
```

```
data=pd.read_excel("Data_Train.csv")
data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Du
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	
					DEL			

```
category=['Airline','Source','Destination','Additional_Info']
category
```

```
['Airline', 'Source', 'Destination', 'Additional_Info']
```

```
for i in category:
  print(i,data[i].unique())
```

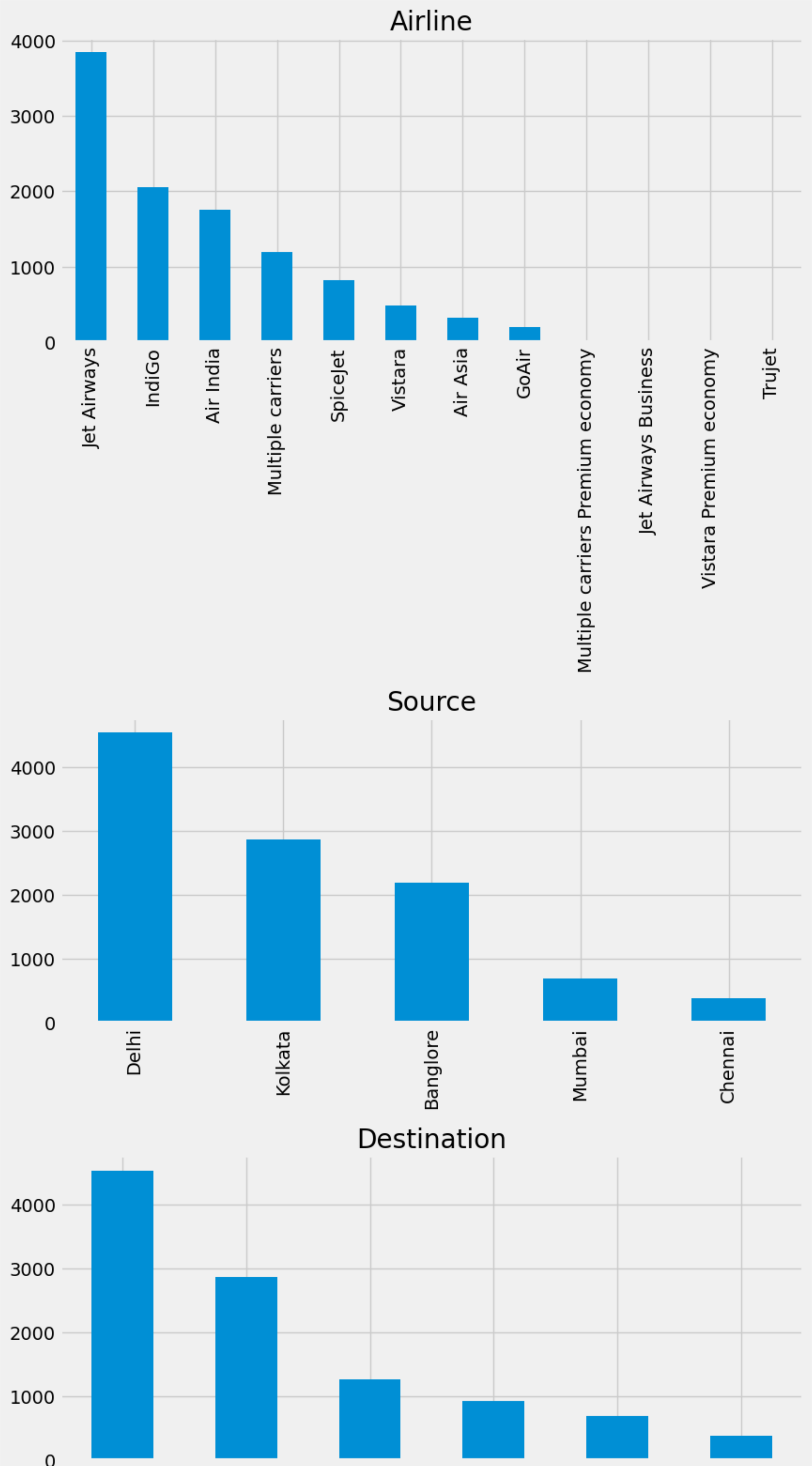
```
Airline ['IndiGo' 'Air India' 'Jet Airways' 'SpiceJet' 'Multiple carriers' 'GoAir'
 'Vistara' 'Air Asia' 'Vistara Premium economy' 'Jet Airways Business'
 'Multiple carriers Premium economy' 'Trujet']
Source ['Banglore' 'Kolkata' 'Delhi' 'Chennai' 'Mumbai']
Destination ['New Delhi' 'Banglore' 'Cochin' 'Kolkata' 'Delhi' 'Hyderabad']
Additional_Info ['No info' 'In-flight meal not included' 'No check-in baggage included'
 '1 Short layover' 'No Info' '1 Long layover' 'Change airports'
 'Business class' 'Red-eye flight' '2 Long layover']
```

```
category_cols=data.select_dtypes(include=['object']).columns
category_cols
```

```
Index(['Airline', 'Source', 'Destination', 'Route', 'Dep_Time', 'Arrival_Time',
       'Duration', 'Total_Stops', 'Additional_Info', 'City1'],
      dtype='object')
```

```
#plotting a barchart for each of the categorical value
for columns in category_cols:
  plt.figure(figsize=(20,4))
```

```
    plt.subplot(121)
    data[columns].value_counts().plot(kind='bar')
    plt.title(columns)
```


Airline
4000
3000
2000
1000
0
Jet Airways
IndiGo
Air India
Multiple carriers
SpiceJet
Vistara
Air Asia
GoAir
Multiple carriers Premium economy
Jet Airways Business
Vistara Premium economy
Trujet
Source
4000
3000
2000
1000
0
Delhi
Kolkata
Banglore
Mumbai
Chennai
Destination
4000
3000
2000
1000
0

Read the Dataset

```
data.Date_of_Journey=data.Date_of_Journey.str.split('/')
data.Date_of_Journey
```

```
0        [24, 03, 2019]
1         [1, 05, 2019]
2         [9, 06, 2019]
3        [12, 05, 2019]
4        [01, 03, 2019]
              ...
10678     [9, 04, 2019]
10679    [27, 04, 2019]
10680    [27, 04, 2019]
10681    [01, 03, 2019]
10682     [9, 05, 2019]
Name: Date_of_Journey, Length: 10683, dtype: object
```

```
data.Total_Stops.unique()
```

```
array(['non-stop', '2 stops', '1 stop', '3 stops', nan, '4 stops'],
      dtype=object)
```

```
data.Route=data.Route.str.split('->')
data.Route
```

```
0                    [BLR → DEL]
1        [CCU → IXR → BBI → BLR]
2        [DEL → LKO → BOM → COK]
3              [CCU → NAG → BLR]
4              [BLR → NAG → DEL]
                  ...
10678                [CCU → BLR]
10679                [CCU → BLR]
10680                [BLR → DEL]
10681                [BLR → DEL]
10682    [DEL → GOI → BOM → COK]
Name: Route, Length: 10683, dtype: object
```

```
data['City1']=data.Route.str[0]
data['City2']=data.Route.str[1]
data['City3']=data.Route.str[2]
data['City4']=data.Route.str[3]
data['City5']=data.Route.str[4]
data['City6']=data.Route.str[5]
```

```
data.Date_of_Journey=data.Date_of_Journey.str.split('/')
data.Date_of_Journey
```

```
0       NaN
1       NaN
2       NaN
3       NaN
4       NaN
         ..
10678   NaN
10679   NaN
10680   NaN
10681   NaN
10682   NaN
Name: Date_of_Journey, Length: 10683, dtype: float64
```

```
data.Dep_Time=data.Dep_Time.str.split(':')
```

```
data['Dep_Time_Hour']=data.Dep_Time.str[0]
data['Dep_Time_Mins']=data.Dep_Time.str[1]
```

```
data.Arrival_Time=data.Arrival_Time.str.split('')
```

```
data['Arrival_date']=data.Arrival_Time.str[1]
data['Time_of_Arrival']=data.Arrival_Time.str[0]
```

```
data['Time_of_Arrival']=data.Time_of_Arrival.str.split(':')
```

```
data['Travel_Hours']=data.Duration.str[0]
data['Travel_Hours']=data['Travel_Hours'].str.split('h')
data['Travel_Hours']=data['Travel_Hours'].str[0]
data.Travel_Hours=data.Travel_Hours
data['Travel_Mins']=data.Duration.str[1]
data.Travel_Mins=data.Travel_Mins.str.split('m')
data.Travel_Mins=data.Travel_Mins.str[0]
```

```
data.Total_Stops.replace('non_stop',0,inplace=True)
data.Total_Stops=data.Total_Stops.str.split('')
data.Total_Stops=data.Total_Stops.str[0]
```

```
data.Total_Stops.replace('non_stop',0,inplace=True)
data.Total_Stops=data.Total_Stops.str.split('')
data.Total_stops=data.Total_Stops.str[0]
```

```
data.Additional_Info.unique()
```

```
array(['No info', 'In-flight meal not included',
       'No check-in baggage included', '1 Short layover', 'No Info',
       '1 Long layover', 'Change airports', 'Business class',
       'Red-eye flight', '2 Long layover'], dtype=object)
```

```
data.Additional_Info.replace('No Info','No Info',inplace=True)
```

```
data.isnull().sum()
```

```
Airline                0
Date_of_Journey    10683
Source                 0
Destination            0
Route                  1
Dep_Time               0
Arrival_Time           0
Duration               0
Total_Stops            1
Additional_Info        0
Price                  0
City1                  1
City2              10683
City3              10683
City4              10683
City5              10683
City6              10683
Dep_Time_Hour          0
Dep_Time_Mins          0
Arrival_date           0
Time_of_Arrival        0
Travel_Hours           0
Travel_Mins            0
dtype: int64
```

```
data.drop(['City4','City5','City6'],axis=1,inplace=True)
```

```
data.drop(['Date_of_Journey','Route','Dep_Time','Arrival_Time','Duration'],axis=1,inplace=True)
data.drop(['Time_of_Arrival'],axis=1,inplace=True)
```

```
data.isnull().sum()
```

```
Airline            0
Source             0
Destination        0
Total_Stops        1
Additional_Info    0
Price              0
```

```
City1                1
City2            10683
City3            10683
Dep_Time_Hour        0
Dep_Time_Mins        0
Arrival_date         0
Travel_Hours         0
Travel_Mins          0
dtype: int64
```

Replacing Missing Values

```
data['City1'].fillna('None',inplace=True)
```

```
data['Arrival_date'].fillna(data['Dep_Time_Hour'],inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Airline          10683 non-null  object 
 1   Source           10683 non-null  object 
 2   Destination      10683 non-null  object 
 3   Total_Stops      10682 non-null  object 
 4   Additional_Info  10683 non-null  object 
 5   Price            10683 non-null  int64  
 6   City1            10683 non-null  object 
 7   City2            0 non-null      float64
 8   City3            0 non-null      float64
 9   Dep_Time_Hour    10683 non-null  object 
 10  Dep_Time_Mins    10683 non-null  object 
 11  Arrival_date     10683 non-null  object 
 12  Travel_Hours     10683 non-null  object 
 13  Travel_Mins      10683 non-null  object 
dtypes: float64(2), int64(1), object(11)
memory usage: 1.1+ MB
```

```
data['Travel_Mins'].fillna(0,inplace=True)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Source           10683 non-null  object
 2   Destination      10683 non-null  object
 3   Total_Stops      10682 non-null  object
 4   Additional_Info  10683 non-null  object
 5   Price            10683 non-null  int64 
 6   Date             10683 non-null  object
 7   Month            10683 non-null  object
 8   Year             10683 non-null  object
 9   City1            10682 non-null  object
 10  City2            10682 non-null  object
 11  City3            10683 non-null  object
 12  Dep_Time_Hour    10683 non-null  object
 13  Dep_Time_Mins    10683 non-null  object
 14  Arrival_date     10683 non-null  object
 15  Travel_Hours     10683 non-null  object
 16  Travel_Mins      10683 non-null  object
dtypes: int64(1), object(16)
memory usage: 1.4+ MB
```

```
data.Dep_Time_Hour=data.Dep_Time_Hour.astype('int64')
data.Dep_Time_Mins=data.Dep_Time_Mins.astype('int64')
data.Arrival_date=data.Arrival_date.astype('int64')
data.head()
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	City2
0	IndiGo	Banglore	New Delhi	[,]	No info	3897	BLR → DEL	NaN
1	Air India	Kolkata	Banglore	[,]	No info	7662	CCU → IXR → BBI → BLR	NaN
2	Jet Airways	Delhi	Cochin	[,]	No info	13882	DEL → LKO → BOM → COK	NaN
3	IndiGo	Kolkata	Banglore	[,]	No info	6218	CCU → NAG → BLR	NaN
4	IndiGo	Banglore	New Delhi	[,]	No info	13302	BLR → NAG → DEL	NaN

```
data[data['Airline']=='Air India']
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	Ci
1	Air India	Kolkata	Banglore	[,]	No info	7662	CCU → IXR → BBI → BLR	N
10	Air India	Delhi	Cochin	[,]	No info	8907	DEL → BLR → COK	N
12	Air India	Chennai	Kolkata	[,]	No info	4667	MAA → CCU	N

```
data.drop(index=6474,inplace=True,axis=0)
```

AMD

```
categorical=['Airline','Source','Destination','Additional_Info','City1']
numerical=['Total_Stops','date','Month','Year','Dep_Time_Hour','Dep_Time_Mins','Arrival_Time_Hou
           'Arrival_Time_Mins','Travel_Hours','Travel_Mins']
```

DEL

Label Encoding

18 Air India Delhi Cochin [,] No info 13381 → N

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

... ...

```
data.Airline=le.fit_transform(data.Airline)
data.Source=le.fit_transform(data.Source)
data.Destination=le.fit_transform(data.Destination)
```

```
data.head()
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	City2	City3	Dep_Time_Hour	Dep_Time_Mins	Arrival_dat
0	3	0	5	[,]	No info	3897	BLR → DEL	NaN	NaN	22	20	
1	1	3	0	[,]	No info	7662	CCU → IXR → BBI → BLR	NaN	NaN	5	50	
2	4	2	1	[,]	No info	13882	DEL → LKO → BOM → COK	NaN	NaN	9	25	
3	3	3	0	[,]	No info	6218	CCU → NAG → BLR	NaN	NaN	18	5	
4	3	0	5	[,]	No info	13302	BLR → NAG → DEL	NaN	NaN	16	50	

Output Columns

```
data.head()
```

	Airline	Source	Destination	Total_Stops	Additional_Info	Price	City1	City2	City3	Dep_Time_Hour	Dep_Time_Mins	Arrival_dat
0	3	0	5	[,]	No info	3897	BLR → DEL	NaN	NaN	22	20	
1	1	3	0	[,]	No info	7662	CCU → IXR → BBI → BLR	NaN	NaN	5	50	
2	4	2	1	[,]	No info	13882	DEL → LKO → BOM → COK	NaN	NaN	9	25	
3	3	3	0	[,]	No info	6218	CCU → NAG → BLR	NaN	NaN	18	5	
4	3	0	5	[,]	No info	13302	BLR → NAG → DEL	NaN	NaN	16	50	

```
data=data[['Airline','Source','Destination','Dep_Time_Hour','Dep_Time_Mins','Arrival_date','Pri
```

```
data.head()
```

	Airline	Source	Destination	Dep_Time_Hour	Dep_Time_Mins	Arrival_date	Price
0	3	0	5	22	20	0	3897
1	1	3	0	5	50	1	7662
2	4	2	1	9	25	0	13882
3	3	3	0	18	5	2	6218
4	3	0	5	16	50	2	13302

```
data.describe()
```

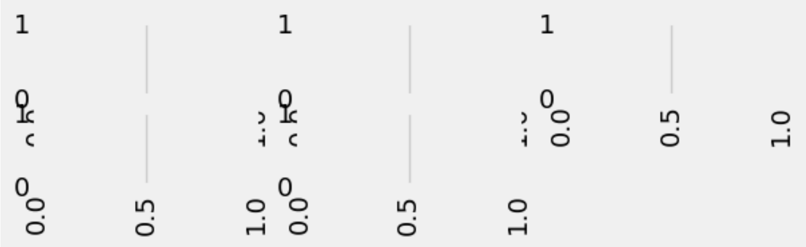
	Airline	Source	Destination	Dep_Time_Hour	Dep_Time_Mins	Arrival_date	Price
count	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000	10682.000000
mean	3.966205	1.952069	1.435967	12.490358	24.408819	0.896836	9086.292735
std	2.352090	1.177110	1.474773	5.748819	18.767225	0.711845	4610.885695
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1759.000000
25%	3.000000	2.000000	0.000000	8.000000	5.000000	0.000000	5277.000000
50%	4.000000	2.000000	1.000000	11.000000	25.000000	1.000000	8372.000000
75%	4.000000	3.000000	2.000000	18.000000	40.000000	1.000000	12373.000000
max	11.000000	4.000000	5.000000	23.000000	55.000000	2.000000	79512.000000

Milestone 3:Explortory Data Analysis

```
import seaborn as sns
c=1
plt.figure(figsize=(20,45))
```

```
<Figure size 2000x4500 with 0 Axes>
<Figure size 2000x4500 with 0 Axes>
```

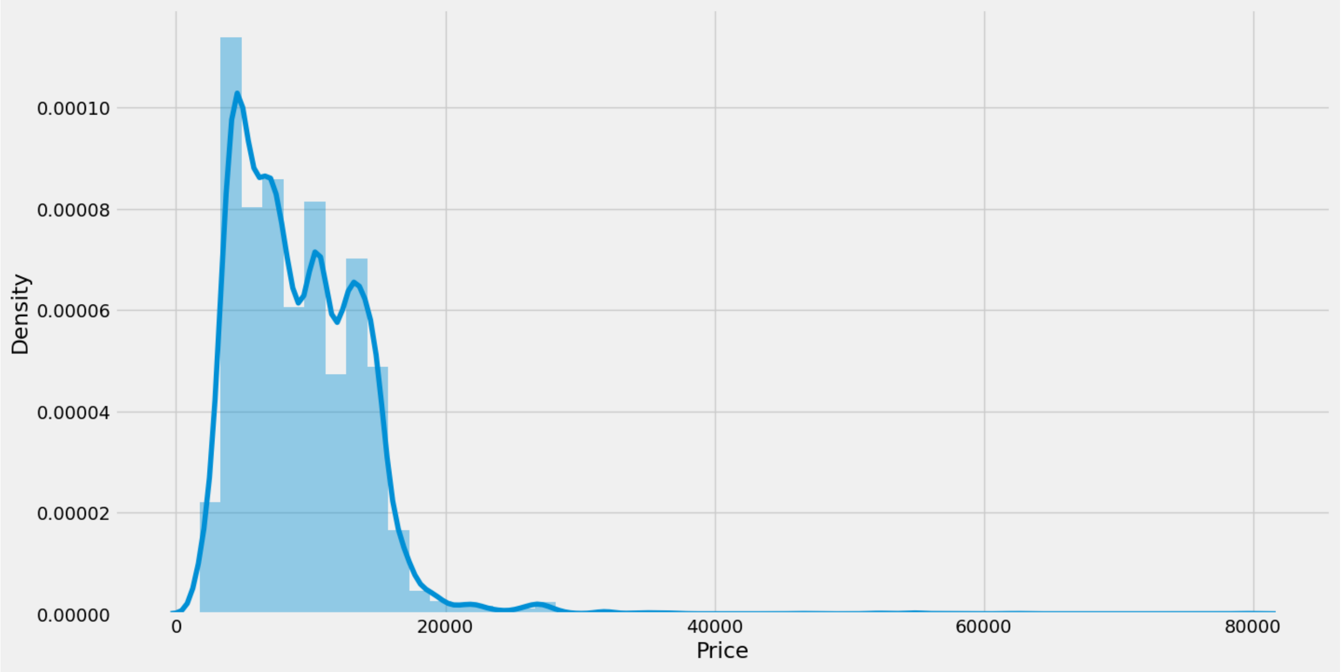
```
for i in categorical:
  plt.subplot(6,3,c)

  plt.xticks(rotation=90)
  plt.tight_layout(pad=3.0)
  c=c+1
```



```
plt.show()
```

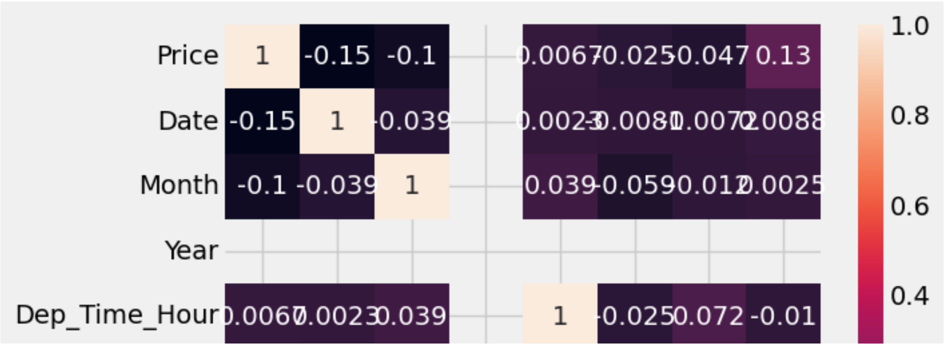
```
plt.figure(figsize=(15,8))
sns.distplot(data.Price)
```

```
<Axes: xlabel='Price', ylabel='Density'>
```



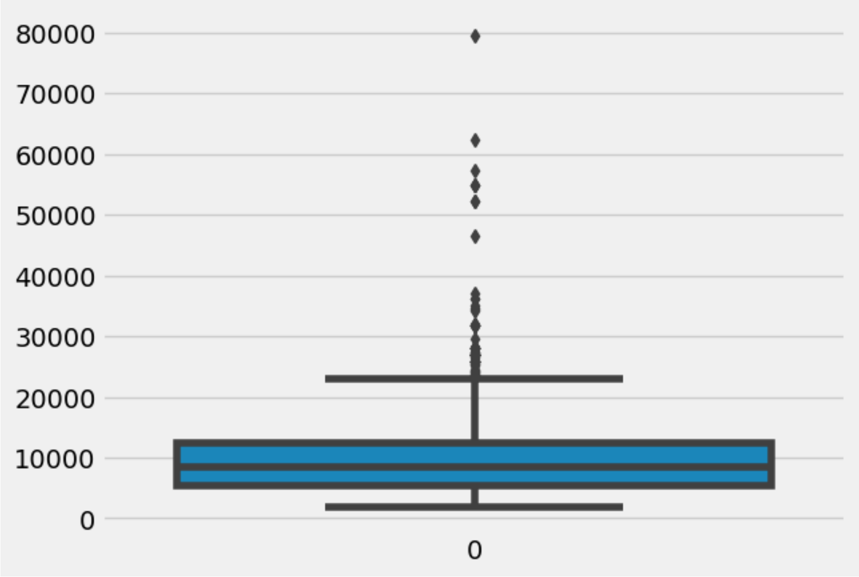
```
sns.heatmap(data.corr(),annot=True)
```

```
<Axes: >
```



```
# Detecting the Outliers
import seaborn as sns
sns.boxplot(data['Price'])
```

```
<Axes: >
```



```
y=data['Price']
x=data.drop(columns=['Price'],axis=1)
```

```
### Scaling the Data
```

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler
```

```
x_scaled=ss.fit_transform
```

```
data.head()
```

	Airline	Source	Destination	Dep_Time_Hour	Dep_Time_Mins	Arrival_date	Price
0	3	0	5	22	20	0	3897
1	1	3	0	5	50	1	7662
2	4	2	1	9	25	0	13882
3	3	3	0	18	5	2	6218
4	3	0	5	16	50	2	13302

```
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.head()
```

	Airline	Source	Destination	Dep_Time_Hour	Dep_Time_Mins	Arrival_date
10005	6	2	1	8	30	1
3684	4	2	1	11	30	1
1034	8	2	1	15	45	2
3909	6	2	1	12	50	0

```
x_train.shape
```

```
(8545, 6)
```

```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, AdaBoostRegresso
rfr=RandomForestRegressor()
gb=GradientBoostingRegressor()
ad=AdaBoostRegressor()
```

Milestone 4:Model Building

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

Regression Model

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
```

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

```
knn=KNeighborsRegressor()
svr=SVR()
dt=DecisionTreeRegressor()
```

```
for i in [knn,svr,dt]:
  i.fit(x_train,y_train)

  y_pred=i.predict(x_test)
  test_score=r2_score(y_test,y_pred)
  train_score=r2_score(y_train,i.predict(x_train))
  if abs(train_score-test_score)<=0.1:
    print(i)
    print('R2 Score is',r2_score(y_test,y_pred))
    print('R2 Score for train data',r2_score(y_train,i.predict(x_train)))
    print('Mean Squared Error is',mean_absolute_error(y_test,y_pred))
    print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
    print('Roott Mean Squared Error is',(mean_squared_error(y_test,y_pred,squared=False)))
```

```
KNeighborsRegressor()
R2 Score is 0.5723008556363665
R2 Score for train data 0.6651092826489714
Mean Squared Error is 1814.9600374356573
Mean Squared Error is 9041141.594010295
Roott Mean Squared Error is 3006.8491139414186
SVR()
R2 Score is -0.032519454381908 73
R2 Score for train data -0.02447186805496515
Mean Squared Error is 3627.1188577608436
Mean Squared Error is 21826451.393833652
Roott Mean Squared Error is 4671.878786295044
DecisionTreeRegressor()
R2 Score is 0.6380395836239183
R2 Score for train data 0.7321334660000298
Mean Squared Error is 1741.7595521859014
Mean Squared Error is 7651489.181144442
Roott Mean Squared Error is 2766.132531377418
```

```
from sklearn.model_selection import cross_val_score

for i in range(2,5):
  cv=cross_val_score(rfr,x,y,cv=i)
  print(rfr,cv.mean())
```

```
RandomForestRegressor() 0.6055338682850067
RandomForestRegressor() 0.6231738744144611
RandomForestRegressor() 0.6389578943109726
```

```
from sklearn.model_selection import RandomizedSearchCV

param_grid={'n_estimators':[10,30,50,70,100],'max_depth':[None,1,2,3],'max_features':['auto','s

rfr=RandomForestRegressor()


rfr.fit(x_train,y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
gb=GradientBoostingRegressor()
gb_res=RandomizedSearchCV(estimator=gb,param_distributions=param_grid,cv=3,verbose=2,n_jobs=-1)



gb_res.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
▸          RandomizedSearchCV
▸ estimator: GradientBoostingRegressor
      ▸ GradientBoostingRegressor
```

Accuracy

```
rfr=RandomForestRegressor(n_estimators=10,max_features='sqrt',max_depth=None)
rfr.fit(x_train,y_train)
y_train_pred=rfr.predict(x_train)
y_test_pred=rfr.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.622418136124503
test accuracy 0.4663663616316054
```

```
knn=KNeighborsRegressor(n_neighbors=2,algorithm='auto',metric_params=None,n_jobs=-1)
knn.fit(x_train,y_train)
y_train_pred=knn.predict(x_train)
y_test_pred=knn.predict(x_test)
print("train accuracy",r2_score(y_train_pred,y_train))
print("test accuracy",r2_score(y_test_pred,y_test))
```

```
train accuracy 0.5082560160438218
test accuracy 0.28385693588182304
```

```
predicted_values=pd.DataFrame({'Actual':y_test,'Predicted':y_pred})

predicted_values
```

	Actual	Predicted
6075	16655	19069.857143
3544	4959	5496.333333
9291	9187	8928.000000
5032	3858	3657.230769
2483	12898	12821.529412
...	...	...
9797	7408	12319.470588
9871	4622	4903.714286
10063	7452	7104.100000
8803	7060	6244.185185
8618	13731	11612.809524

2137 rows × 2 columns

```
prices=rfr.predict(x_test)
```

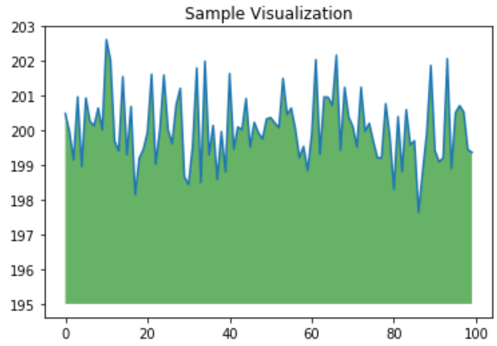
```
price_list=pd.DataFrame({'Price':prices})
```

```
price_list
```

	Price
0	19999.841747
1	5604.342171
2	8959.272976
3	3654.219529
4	13165.037332
...	...
2132	12589.262280
2133	4764.224621
2134	6873.395159
2135	6253.708491
2136	11819.839555

2137 rows × 1 columns

```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```



✓ 0s completed at 7:24 AM ● ✕