```
from google.colab import files
uploaded=files.upload()
```

Choose Files Data_Train.xlsx

- **Data_Train.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 530389 bytes, last modified: 1/1/2021 - 100% done

```
Saving Data_Train.xlsx to Data_Train (1).xlsx
```

```
from google.colab import files
uploaded1=files.upload()
```

Choose Files Test_set.xlsx

- **Test_set.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 120774 bytes, last modified: 1/1/2021 - 100% done

```
Saving Test_set.xlsx to Test_set (1).xlsx
```

```
import pandas as pd
import io
#reading a excel file and converting it into a dataframe object
train_df=pd.DataFrame(pd.read_excel(io.BytesIO(uploaded['Data_Train.xlsx'])))


#reading a excel file and converting into a dataframe object
test_df=pd.DataFrame(pd.read_excel(io.BytesIO(uploaded1['Test_set.xlsx'])))


train_df.head(5)
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Durat
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 5
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI	05:50	13:15	7h 2

```
test_df.head()
```

```
train_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
```

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
```

```
train_shape=train_df.shape
test_shape=test_df.shape
print("Train dataset shape: ", train_shape)
print("Test dataset shape: ",test_shape)
```

```
Train dataset shape:  (10683, 11)
Test dataset shape:  (2671, 10)
```

```
#basic statistical data
train_df.describe()
```

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

```
#null values
train_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
```

From above we can see that in train dataset we have only 1 null value in row column and 1 null value in total_stops column so we can drop these two null value rows

```
#dropping null values
train_df.dropna(inplace=True)
```

```
train_df.shape
```

```
(10682, 11)
```

```
test_df.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
```

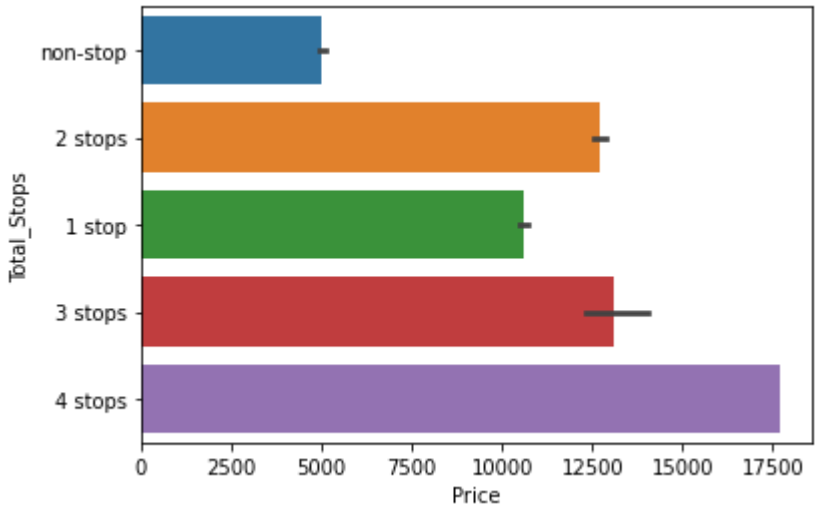
```
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
dtype: int64
```

In train dataset we do not have any null values

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

```
barplot=sns.barplot(train_df['Price'],train_df['Total_Stops'])
barplot.set_xlabel('Price')
barplot.set_ylabel('Total_Stops')
```

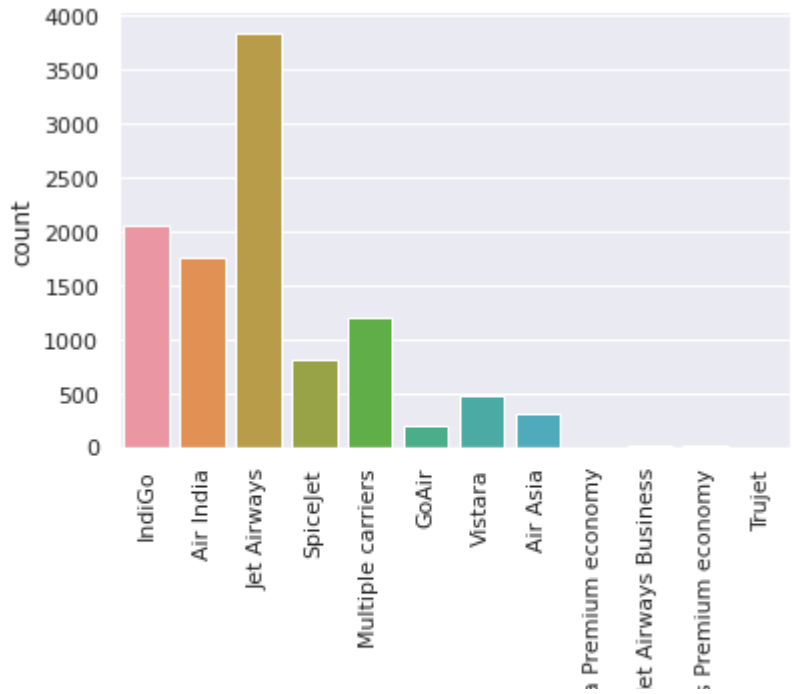
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
Text(0, 0.5, 'Total_Stops')
```



We can see that the price is highest for 4 stops and lowest for 0 stops,so the price is increasing as the number of stops increasing

```
sns.set(style='darkgrid')
countplot=sns.countplot(x='Airline',data=train_df)
countplot.set_xticklabels(countplot.get_xticklabels(),rotation=90)
```

```
[Text(0, 0, 'IndiGo'),
 Text(0, 0, 'Air India'),
 Text(0, 0, 'Jet Airways'),
 Text(0, 0, 'SpiceJet'),
 Text(0, 0, 'Multiple carriers'),
 Text(0, 0, 'GoAir'),
 Text(0, 0, 'Vistara'),
 Text(0, 0, 'Air Asia'),
 Text(0, 0, 'Vistara Premium economy'),
 Text(0, 0, 'Jet Airways Business'),
 Text(0, 0, 'Multiple carriers Premium economy'),
 Text(0, 0, 'Trujet')]
```



So maximum flights are in "Jet Airways" and minimum are in "Vistara Premium economy" , "Jet Airways Buisness" , "Multiple carriers Premium economy" and "Trujet"

HANDLING CATEGORICAL DATA

```
#number of distinct airlines present in train data
train_df['Airline'].nunique()
```

```
12
```

```
#number of distinct airlines present in test data
test_df['Airline'].nunique()
```

```
11
```

```
#value count for each distinct airline in train data
train_df['Airline'].value_counts()
```

```
Jet Airways                          3849
IndiGo                               2053
Air India                            1751
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
```

```
#value count for each distinct airline in test data
test_df['Airline'].value_counts()
```

```
Jet Airways                          897
IndiGo                               511
Air India                            440
Multiple carriers                    347
SpiceJet                             208
Vistara                              129
Air Asia                              86
GoAir                                 46
Multiple carriers Premium economy      3
Jet Airways Business                   2
Vistara Premium economy                2
Name: Airline, dtype: int64
```

```
#number of distinct destination present in train dataset
train_df['Destination'].nunique()
```

```
6
```

```
#number of distinct destination present in test dataset
test_df['Destination'].nunique()
```

```
6
```

```
#value count for each distinct destination in train dataset
train_df['Destination'].value_counts()
```

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

```
#value count for each distinct destination in test dataset
```

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

```
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: Destination, dtype: int64
```

```
#value count for each distinct source in train data
train_df['Source'].value_counts()
```

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

```
#value count for each distinct source in test data
test_df['Source'].value_counts()
```

```
Delhi       1145
Kolkata      710
Banglore     555
Mumbai       186
Chennai       75
Name: Source, dtype: int64
```

```
#value count for each distict Additional_Info in train data
train_df['Additional_Info'].value_counts()
```

```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
Red-eye flight                     1
2 Long layover                     1
1 Short layover                    1
Name: Additional_Info, dtype: int64
```

```
#value count for each distict Additional_Info in test data
test_df['Additional_Info'].value_counts()
```

```
No info                         2148
In-flight meal not included      444
No check-in baggage included      76
Change airports                    1
```

```
1 Long layover                    1
Business class                    1
Name: Additional_Info, dtype: int64
```

```
#from above we can that majority of part contain no info
#so we can drop the Additional_Info column
train_df.drop(['Additional_Info'],axis=1,inplace=True)
test_df.drop(['Additional_Info'],axis=1,inplace=True)
```

```
#doing OneHotEncoding on 'Airline' , 'estination' and 'Source' column
train_Airline=pd.get_dummies(train_df[['Airline']],drop_first=True)
train_Destination=pd.get_dummies(train_df[['Destination']],drop_first=True)
train_Source=pd.get_dummies(train_df[['Source']],drop_first=True)

test_Airline=pd.get_dummies(test_df[['Airline']],drop_first=True)
test_Destination=pd.get_dummies(test_df[['Destination']],drop_first=True)
test_Source=pd.get_dummies(test_df[['Source']],drop_first=True)
```

```
train_Airline.head()
```

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multip carrie
0	0	0	1	0	0	
1	1	0	0	0	0	
2	0	0	0	1	0	
3	0	0	1	0	0	
4	0	0	1	0	0	

```
test_Airline.head()
```

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multip carrie
0	0	0	0	1	0	
1	0	0	1	0	0	
2	0	0	0	1	0	
3	0	0	0	0	0	
4	0	0	0	0	0	

```
train_Destination.head()
```

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	D
0	0	0	0	0	
1	0	0	0	0	
2	1	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

```
test_Destination.head()
```

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	D
0	1	0	0	0	
1	0	0	0	0	
2	1	0	0	0	
3	1	0	0	0	
4	0	1	0	0	

```
train_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

```
test_Source.head()
```

Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai

```
#value count for no of stops for train data
train_df['Total_Stops'].value_counts()
```

```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

```
#value count for no of stops for test data
test_df['Total_Stops'].value_counts()
```

```
1 stop      1431
non-stop     849
2 stops      379
3 stops       11
4 stops        1
Name: Total_Stops, dtype: int64
```

```
#converting object values into integer in Total_Stops
train_df['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1,
test_df['Total_Stops'].replace(['1 stop', 'non-stop', '2 stops', '3 stops', '4 stops'], [1, 0
```

```
#Route column as Route and Total Stops column are related to each other
train_df.drop(['Route'],axis=1,inplace=True)
test_df.drop(['Route'],axis=1,inplace=True)
```

```
#drop Airline,Source,Destination columns
train_df.drop(['Airline','Source','Destination'],axis=1,inplace=True)
test_df.drop(['Airline','Source','Destination'],axis=1,inplace=True)
```

EDA

```
#convert duration times in minutes
train_df['Duration_min']=train_df['Duration'].str.replace("h", '*60').str.replace(' ','+').st
test_df['Duration_min']=test_df['Duration'].str.replace("h", '*60').str.replace(' ','+').str.
```

```
#drop "Duration" column
train_df.drop(['Duration'],axis=1,inplace=True)
test_df.drop(['Duration'],axis=1,inplace=True)
```

```
#Extracting departure hours and departure minutes from "Dep_Time " column
train_df['Dep_hour']=pd.to_datetime(train_df.Dep_Time).dt.hour
```

```
train_df['Dep_min']=pd.to_datetime(train_df.Dep_Time).dt.minute

test_df['Dep_hour']=pd.to_datetime(test_df.Dep_Time).dt.hour
test_df['Dep_min']=pd.to_datetime(test_df.Dep_Time).dt.minute


#Now we can don't need "Dep_Time" column,so we can drop this column
#drop Dep_Time column
train_df.drop(["Dep_Time"],axis=1,inplace=True)
test_df.drop(["Dep_Time"],axis=1,inplace=True)


#Similarly , Extracting arrival hour,arrival minutes from "Arrival_Time" column
train_df['Arrival_hour']=pd.to_datetime(train_df.Arrival_Time).dt.hour
train_df['Arrival_min']=pd.to_datetime(train_df.Arrival_Time).dt.minute

test_df['Arrival_hour']=pd.to_datetime(test_df.Arrival_Time).dt.hour
test_df['Arrival_min']=pd.to_datetime(test_df.Arrival_Time).dt.minute


#drop "Arrival_Time" column
train_df.drop(['Arrival_Time'],axis=1,inplace=True)
test_df.drop(['Arrival_Time'],axis=1,inplace=True)


#Extracting journey day, journey month, journey year,journey weekday from date of journey
train_df['Journey_Day'] = pd.to_datetime(train_df.Date_of_Journey, format='%d/%m/%Y').dt.day
train_df['Journey_Month'] = pd.to_datetime(train_df.Date_of_Journey, format='%d/%m/%Y').dt.mo
train_df['Journey_Year']= pd.to_datetime(train_df.Date_of_Journey, format='%d/%m/%Y').dt.year
train_df['Journey_Weekday']= pd.to_datetime(train_df.Date_of_Journey, format='%d/%m/%Y').dt.w

test_df['Journey_Day'] = pd.to_datetime(test_df.Date_of_Journey, format='%d/%m/%Y').dt.day
test_df['Journey_Month'] = pd.to_datetime(test_df.Date_of_Journey, format='%d/%m/%Y').dt.mont
test_df['Journey_Year']= pd.to_datetime(test_df.Date_of_Journey, format='%d/%m/%Y').dt.year
test_df['Journey_Weekday']= pd.to_datetime(test_df.Date_of_Journey, format='%d/%m/%Y').dt.wee


sns.set(style='darkgrid')
countplot=sns.countplot(x='Journey_Weekday',data=train_df)
countplot.set_xticklabels(countplot.get_xticklabels(),rotation=90)
```

```
[Text(0, 0, '0'),
 Text(0, 0, '1'),
 Text(0, 0, '2'),
 Text(0, 0, '3'),
 Text(0, 0, '4'),
 Text(0, 0, '5'),
 Text(0, 0, '6')]
```


```
train_df['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
```

```
test_df['Journey_Year']
```

```
0       2019
1       2019
2       2019
3       2019
4       2019
        ...
2666    2019
2667    2019
2668    2019
2669    2019
2670    2019
Name: Journey_Year, Length: 2671, dtype: int64
```

```
#Journey_Year column contain only 2019 year,so we can drop this column
#dropping "Journey_Year" column
train_df.drop(['Journey_Year'],axis=1,inplace=True)
test_df.drop(['Journey_Year'],axis=1,inplace=True)
#dropping "Date_of_Journey" column
train_df.drop(['Date_of_Journey'],axis=1,inplace=True)
test_df.drop(['Date_of_Journey'],axis=1,inplace=True)
```

```
#concate dataframes
train_data=pd.concat([train_df,train_Airline,train_Source,train_Destination],axis=1)
test_data=pd.concat([test_df,test_Airline,test_Source,test_Destination],axis=1)
```

```
train_data.head()
```

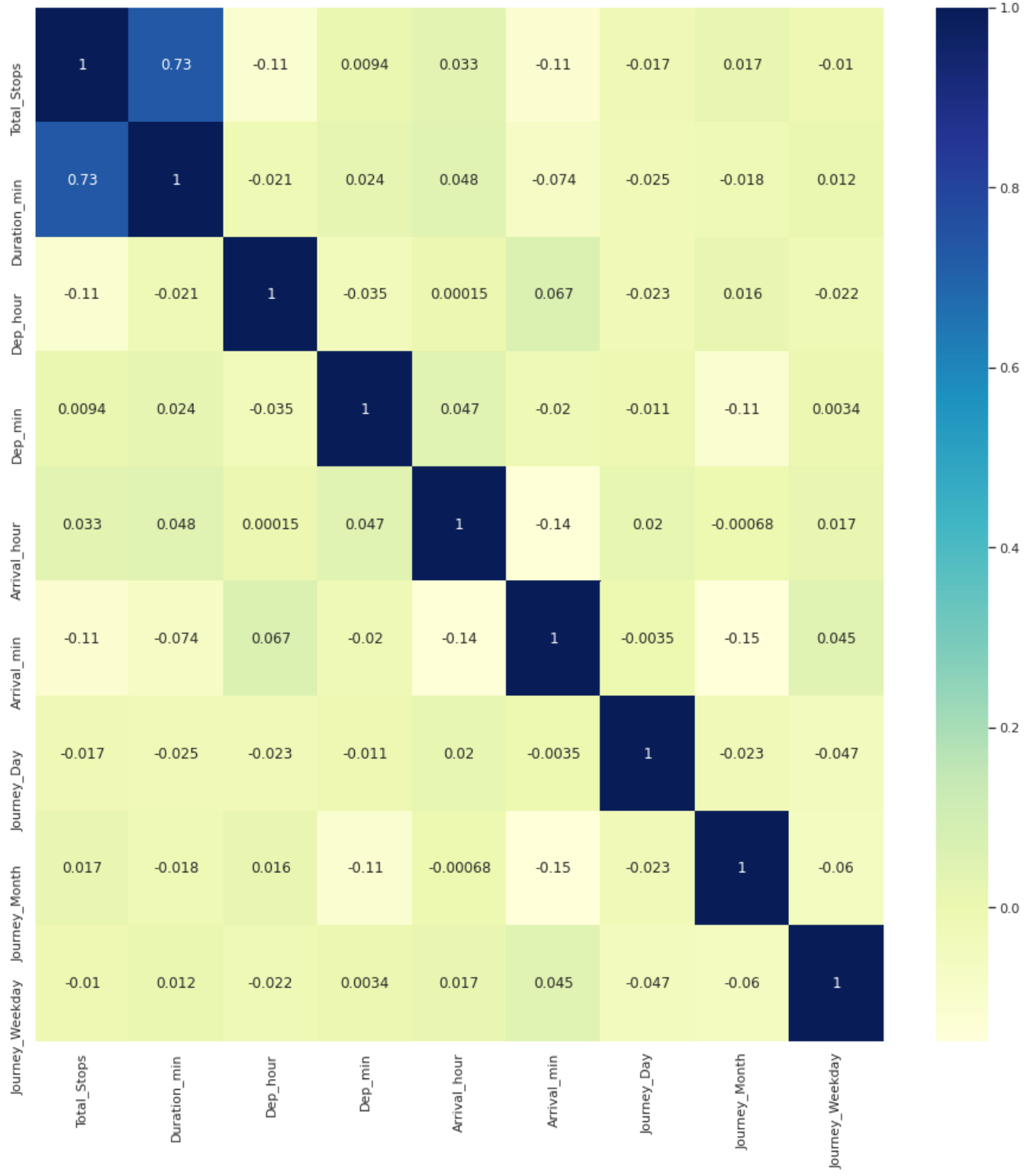
	Total_Stops	Price	Duration_min	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Journ
0	0	3897	170	22	20	1	10	
1	2	7662	445	5	50	13	15	
2	2	13882	1140	9	25	4	25	
3	1	6218	325	18	5	23	30	
4	1	13302	285	16	50	21	35	

```
test_data.head()
```

	Total_Stops	Duration_min	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Journey_Day
0	1	655	17	30	4	25	6
1	1	240	6	20	10	20	12
2	1	1425	19	15	19	0	21
3	1	780	8	0	21	0	21
4	0	170	23	55	2	45	24

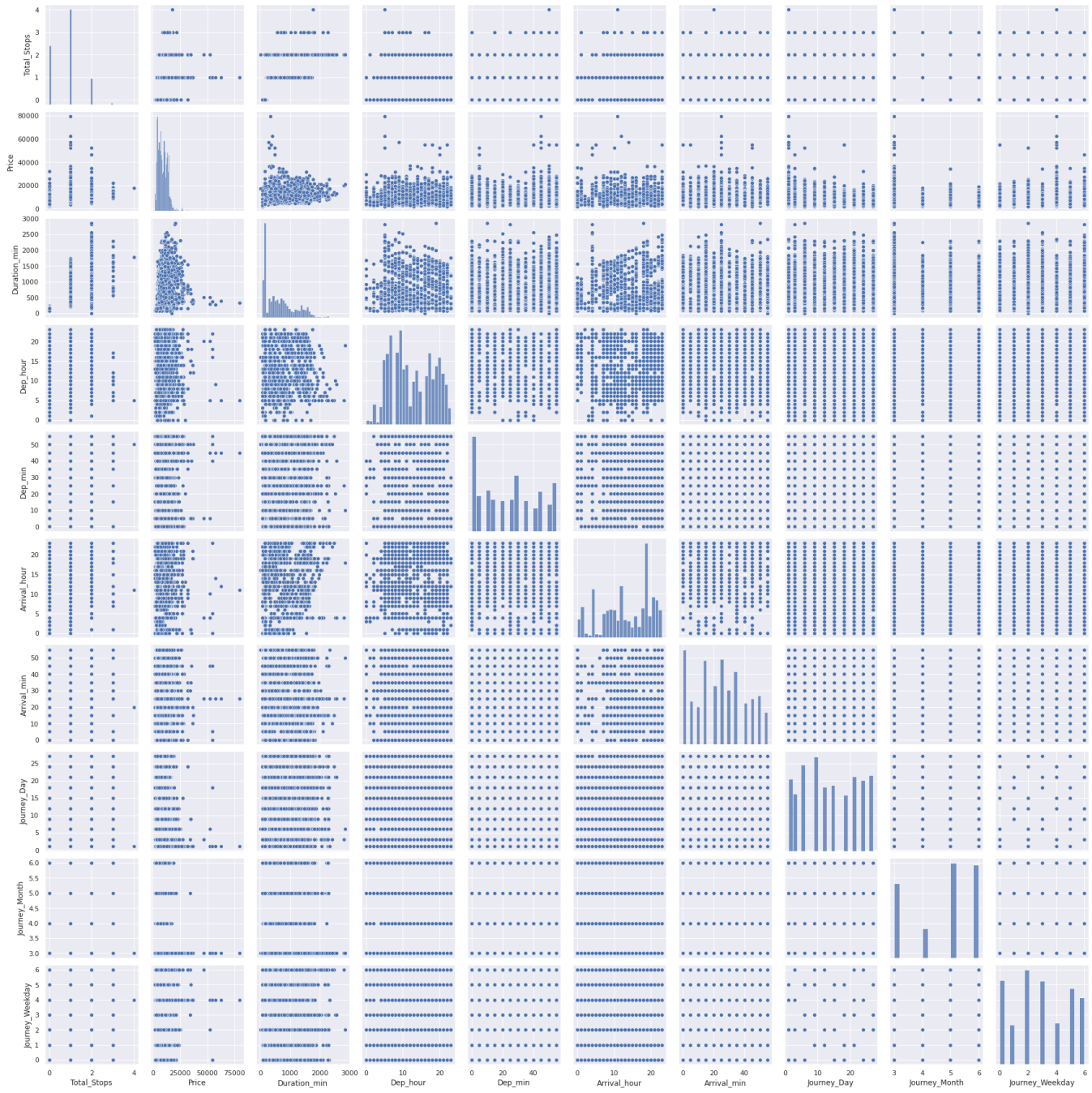
```
plt.figure(figsize=(16,16))
sns.heatmap(train_df.corr(),annot=True,cmap="YlGnBu")
plt.show()
```


```
plt.figure(figsize=(16,16))
sns.heatmap(test_df.corr(),annot=True,cmap="YlGnBu")
plt.show()
```




```
sns.pairplot(train_df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f6c254cdac8>
```



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


#splitting the dataset as training and testing dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25, random_state=111)


from sklearn.model_selection import RandomizedSearchCV


import numpy as np


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)



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

     RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                           max_depth=None, max_features='auto', max_leaf_nodes=None,
                           max_samples=None, min_impurity_decrease=0.0,
                           min_impurity_split=None, min_samples_leaf=1,
                           min_samples_split=2, min_weight_fraction_leaf=0.0,
                           n_estimators=100, n_jobs=None, oob_score=False,
                           random_state=None, verbose=0, warm_start=False)


y_pred = reg_rf.predict(X_test)


reg_rf.score(X_train, y_train)

     0.952236945465693


reg_rf.score(X_test, y_test)

     0.8215857643099175


from sklearn.model_selection import RandomizedSearchCV
```

```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]


# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}


# Random search of parameters, using 5 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid,scoring=


rf_random.fit(X_train,y_train)
```

```
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, ma
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, m
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, ma
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, m
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, ma
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, m
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, ma
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, m
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, m
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto,
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, m
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto,
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, m
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto,
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, m
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto,
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, m
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto,
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt,
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt,
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt,
```

```
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, 

[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, m
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, 
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, m
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, 
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, m
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, 
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, m
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, 
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, m
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, 
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, m
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, 
```

```
rf_random.best_params_
```

```
{'max_depth': 20,
 'max_features': 'auto',
 'min_samples_leaf': 1,
 'min_samples_split': 15,
 'n_estimators': 700}
```

```
prediction = rf_random.predict(X_test)
```

```
import pickle
# open a file, where you ant to store the data
file = open('flight_rf.pkl', 'wb')

# dump information to that file
```

```
pickle.dump(reg_rf, file)
```

```
!pip install flask-ngrok
%mkdir templates -p
%mkdir css -p
```

```
Requirement already satisfied: flask-ngrok in /usr/local/lib/python3.6/dist-packages (0
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: Flask>=0.8 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-package
```

```
%%writefile css/styles.css
body {
    background-color: #e1f4f3;
    text-align: center;
}

.navbar {
    background-color: #333333;
}

a {
    color: #f1f9f9;
}

a:hover {
    color: #f0f0f0;
  }
```

```
Overwriting css/styles.css
```

```
%%writefile templates/home.html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Flight Price Prediction</title>


    <!-- BootStrap -->
```

```
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/boots
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" c


    <!-- css -->
    <link rel="stylesheet" href="/content/css/styles.css">


</head>

<body>

    <!-- As a heading -->
    <nav class="navbar navbar-inverse navbar-fixed-top">
        <div class="container-fluid">
            <div class="navbar-header">
                <a class="navbar-brand" href="/">FLIGHT PRICE</a>
            </div>
        </div>
    </nav>

    <br><br><br>


    <div class="container">



        <form action="\predict" method="post">


            <div class="row">
                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title">Departure Date</h5>
                            <!-- Departure -->
                            <input type="datetime-local" name="Dep_Time" id="Dep_Time" requir
                        </div>
                    </div>
                </div>
                <br>
                <br>
                <br>
                <div class="col-sm-6">
                    <div class="card">
                        <div class="card-body">
                            <h5 class="card-title">Arrival Date</h5>
                            <!-- Arrival -->
                            <input type="datetime-local" name="Arrival_Time" id="Arrival_Time
                        </div>
                    </div>
```

```
            </div>
        </div>
    </div>

    <br>
    <br>
    <br>

    <div class="row">
        <div class="col-sm-6">
            <div class="card">
                <div class="card-body">
                    <!-- Source -->
                    <h5 class="card-title">Source</h5>
                    <select name="Source" id="Source" required="required">
                        <option value="Delhi">Delhi</option>
                        <option value="Kolkata">Kolkata</option>
                        <option value="Mumbai">Mumbai</option>
                        <option value="Chennai">Chennai</option>
                    </select>
                </div>
            </div>
        </div>
        <div class="col-sm-6">
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">Destination</h5>
                    <!-- Destination -->
                    <select name="Destination" id="Destination" required="required">
                        <option value="Cochin">Cochin</option>
                        <option value="Delhi">Delhi</option>
                        <option value="New Delhi">New Delhi</option>
                        <option value="Hyderabad">Hyderabad</option>
                        <option value="Kolkata">Kolkata</option>
                    </select>
                </div>
            </div>
        </div>
    </div>



    <br>
    <br>
    <br>

    <div class="row">
        <div class="col-sm-6">
            <div class="card">
                <div class="card-body">
                    <h5 class="card-title">Stopage</h5>
                    <!-- Total Stops -->
```

```
                        <select name="stops" required="required">
                            <option value="0">Non-Stop</option>
                            <option value="1">1</option>
                            <option value="2">2</option>
                            <option value="3">3</option>
                            <option value="4">4</option>
                        </select>
                    </div>
                </div>
            </div>
            <div class="col-sm-6">
                <div class="card">
                    <div class="card-body">
                        <h5 class="card-title">Which Airline you want to travel?</h5>
                        <!-- Airline -->
                        <select name="airline" id="airline" required="required">
                            <option value="Jet Airways">Jet Airways</option>
                            <option value="IndiGo">IndiGo</option>
                            <option value="Air India">Air India</option>
                            <option value="Multiple carriers">Multiple carriers</option>
                            <option value="SpiceJet">SpiceJet</option>
                            <option value="Vistara">Vistara</option>
                            <option value="Air Asia">Air Asia</option>
                            <option value="GoAir">GoAir</option>
                            <option value="Multiple carriers Premium economy">Multiple ca
                            </option>
                            <option value="Jet Airways Business">Jet Airways Business</op
                            <option value="Vistara Premium economy">Vistara Premium econo
                            <option value="Trujet">Trujet</option>
                        </select>
                    </div>
                </div>
            </div>
        </div>



        <br>
        <br>
        <br>
        <!-- Submit -->
        <input type="submit" value="Submit" class="btn btn-secondary">
    </form>

    <br>
    <br>
    <h3>{{ prediction_text }}</h3>

    <br>
    <br>
```

```
    </div>



    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>

</body>

</html>

     Overwriting templates/home.html


from flask_ngrok import run_with_ngrok
from flask import Flask


%%writefile Procfile
web: gunicorn app:app



from flask import Flask, request, render_template
from flask_cors import cross_origin
import sklearn
import pickle
import pandas as pd


app = Flask(__name__)
run_with_ngrok(app)
model = pickle.load(open("flight_rf.pkl", "rb"))



@app.route("/")
@cross_origin()
def home():
    return render_template("home.html")
```

```
@app.route("/predict", methods = ["GET", "POST"])
@cross_origin()
def predict():
    if request.method == "POST":

        # Date_of_Journey
        date_dep = request.form["Dep_Time"]
        Journey_day = int(pd.to_datetime(date_dep, format="%Y-%m-%dT%H:%M").day)
        Journey_month = int(pd.to_datetime(date_dep, format ="%Y-%m-%dT%H:%M").month)
        # print("Journey Date : ",Journey_day, Journey_month)

        # Departure
        Dep_hour = int(pd.to_datetime(date_dep, format ="%Y-%m-%dT%H:%M").hour)
        Dep_min = int(pd.to_datetime(date_dep, format ="%Y-%m-%dT%H:%M").minute)
        # print("Departure : ",Dep_hour, Dep_min)

        # Arrival
        date_arr = request.form["Arrival_Time"]
        Arrival_hour = int(pd.to_datetime(date_arr, format ="%Y-%m-%dT%H:%M").hour)
        Arrival_min = int(pd.to_datetime(date_arr, format ="%Y-%m-%dT%H:%M").minute)
        # print("Arrival : ", Arrival_hour, Arrival_min)

        # Duration
        dur_hour = abs(Arrival_hour - Dep_hour)
        dur_min = abs(Arrival_min - Dep_min)
        # print("Duration : ", dur_hour, dur_min)

        # Total Stops
        Total_stops = int(request.form["stops"])
        # print(Total_stops)

        # Airline
        # AIR ASIA = 0 (not in column)
        airline=request.form['airline']
        if(airline=='Jet Airways'):
            Jet_Airways = 1
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='IndiGo'):
            Jet_Airways = 0
            IndiGo = 1
            Air_India = 0
```

```
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Air India'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 1
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Multiple carriers'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 1
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='SpiceJet'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 1
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Vistara'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
```

```
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 1
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='GoAir'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 1
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Multiple carriers Premium economy'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 1
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Jet Airways Business'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 1
        Vistara_Premium_economy = 0
        Trujet = 0

    elif (airline=='Vistara Premium economy'):
        Jet_Airways = 0
        IndiGo = 0
```

```
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 1
        Trujet = 0

    elif (airline=='Trujet'):
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 1

    else:
        Jet_Airways = 0
        IndiGo = 0
        Air_India = 0
        Multiple_carriers = 0
        SpiceJet = 0
        Vistara = 0
        GoAir = 0
        Multiple_carriers_Premium_economy = 0
        Jet_Airways_Business = 0
        Vistara_Premium_economy = 0
        Trujet = 0

    # print(Jet_Airways,
    #     IndiGo,
    #     Air_India,
    #     Multiple_carriers,
    #     SpiceJet,
    #     Vistara,
    #     GoAir,
    #     Multiple_carriers_Premium_economy,
    #     Jet_Airways_Business,
    #     Vistara_Premium_economy,
    #     Trujet)

    # Source
    # Banglore = 0 (not in column)
    Source = request.form["Source"]
    if (Source == 'Delhi'):
```

```
if (Source == 'Delhi'):
    s_Delhi = 1
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 0

elif (Source == 'Kolkata'):
    s_Delhi = 0
    s_Kolkata = 1
    s_Mumbai = 0
    s_Chennai = 0

elif (Source == 'Mumbai'):
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 1
    s_Chennai = 0

elif (Source == 'Chennai'):
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 1

else:
    s_Delhi = 0
    s_Kolkata = 0
    s_Mumbai = 0
    s_Chennai = 0

# print(s_Delhi,
#     s_Kolkata,
#     s_Mumbai,
#     s_Chennai)

# Destination
# Banglore = 0 (not in column)
Source = request.form["Destination"]
if (Source == 'Cochin'):
    d_Cochin = 1
    d_Delhi = 0
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0

elif (Source == 'Delhi'):
    d_Cochin = 0
    d_Delhi = 1
    d_New_Delhi = 0
    d_Hyderabad = 0
    d_Kolkata = 0
```

```
    elif (Source == 'New_Delhi'):
        d_Cochin = 0
        d_Delhi = 0
        d_New_Delhi = 1
        d_Hyderabad = 0
        d_Kolkata = 0

    elif (Source == 'Hyderabad'):
        d_Cochin = 0
        d_Delhi = 0
        d_New_Delhi = 0
        d_Hyderabad = 1
        d_Kolkata = 0

    elif (Source == 'Kolkata'):
        d_Cochin = 0
        d_Delhi = 0
        d_New_Delhi = 0
        d_Hyderabad = 0
        d_Kolkata = 1

    else:
        d_Cochin = 0
        d_Delhi = 0
        d_New_Delhi = 0
        d_Hyderabad = 0
        d_Kolkata = 0

    # print(
    #     d_Cochin,
    #     d_Delhi,
    #     d_New_Delhi,
    #     d_Hyderabad,
    #     d_Kolkata
    # )


#     ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
#    'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
#    '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']

    prediction=model.predict([[
        Total_stops,
        Journey_day,
        Journey_month,
```

```
                Dep_hour,
                Dep_min,
                Arrival_hour,
                Arrival_min,
                dur_hour,
                dur_min,
                Air_India,
                GoAir,
                IndiGo,
                Jet_Airways,
                Jet_Airways_Business,
                Multiple_carriers,
                Multiple_carriers_Premium_economy,
                SpiceJet,
                Trujet,
                Vistara,
                Vistara_Premium_economy,
                s_Chennai,
                s_Delhi,
                s_Kolkata,
                s_Mumbai,
                d_Cochin,
                d_Delhi,
                d_Hyderabad,
                d_Kolkata,
                d_New_Delhi
            ]])

            output=round(prediction[0],2)

            return render_template('home.html',prediction_text="Your Flight price is Rs. {}".form


        return render_template("home.html")



! pip install flask_cors
```

```
Requirement already satisfied: flask_cors in /usr/local/lib/python3.6/dist-packages (3.0
Requirement already satisfied: Six in /usr/local/lib/python3.6/dist-packages (from flask
Requirement already satisfied: Flask>=0.9 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: click>=5.1 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: itsdangerous>=0.24 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: Werkzeug>=0.15 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: Jinja2>=2.10.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/dist-package
```

```
app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
```

```
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Running on http://aea25e561732.ngrok.io
 * Traffic stats available on http://127.0.0.1:4040
```

```
!pip freeze requirements.txt
```