```
!pip install scikit-learn==1.3.2
```

```
Requirement already satisfied: scikit-learn==1.3.2 in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.3.2) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.3.2) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.3.2) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn==1.3.2) (3.2.0)
```

```
pip show scikit-learn
```

```
Name: scikit-learn
Version: 1.3.2
Summary: A set of python modules for machine learning and data mining
Home-page: http://scikit-learn.org
Author:
Author-email:
License: new BSD
Location: /usr/local/lib/python3.10/dist-packages
Requires: joblib, numpy, scipy, threadpoolctl
Required-by: bigframes, fastai, imbalanced-learn, librosa, mlxtend, qudida, sklearn-pandas, yellowbrick
```

```
pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.23.5)
Requirement already satisfied: scipy>=1.5.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.2.0)
```

```
pip install --upgrade joblib
```

```
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (1.3.2)
```

```
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.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
df_train=pd.read_excel('/content/Data_Train.xlsx')
```

```
df_train.head()
```

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

```
df_train.tail()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	CCU → BLR	20:45	23:20	2h 35m	non-stop	No info	4145
	Jet				BLR						

```
df_train.describe
```

```
<bound method NDFrame.describe of            Airline Date_of_Journey    Source Destination  \
0           IndiGo      24/03/2019  Banglore   New Delhi
1        Air India       1/05/2019   Kolkata    Banglore
2      Jet Airways       9/06/2019     Delhi      Cochin
3           IndiGo      12/05/2019   Kolkata    Banglore
4           IndiGo      01/03/2019  Banglore   New Delhi
...            ...             ...       ...         ...
10678     Air Asia       9/04/2019   Kolkata    Banglore
10679    Air India      27/04/2019   Kolkata    Banglore
10680  Jet Airways      27/04/2019  Banglore       Delhi
10681      Vistara      01/03/2019  Banglore   New Delhi
10682    Air India       9/05/2019     Delhi      Cochin

                    Route Dep_Time  Arrival_Time Duration Total_Stops  \
0                 BLR → DEL    22:20  01:10 22 Mar   2h 50m    non-stop
1      CCU → IXR → BBI → BLR    05:50         13:15   7h 25m     2 stops
2      DEL → LKO → BOM → COK    09:25  04:25 10 Jun      19h     2 stops
3            CCU → NAG → BLR    18:05         23:30   5h 25m      1 stop
4            BLR → NAG → DEL    16:50         21:35   4h 45m      1 stop
...                      ...      ...           ...      ...         ...
10678            CCU → BLR    19:55         22:25   2h 30m    non-stop
10679            CCU → BLR    20:45         23:20   2h 35m    non-stop
10680            BLR → DEL    08:20         11:20       3h    non-stop
10681            BLR → DEL    11:30         14:10   2h 40m    non-stop
10682  DEL → GOI → BOM → COK    10:55         19:15   8h 20m     2 stops

      Additional_Info  Price
0             No info   3897
1             No info   7662
2             No info  13882
3             No info   6218
4             No info  13302
...               ...    ...
10678         No info   4107
10679         No info   4145
10680         No info   7229
10681         No info  12648
10682         No info  11753

[10683 rows x 11 columns]>
```

```
df_train.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
```

```
df_train['Duration'].value_counts()
```

```
2h 50m     550
1h 30m     386
2h 45m     337
2h 55m     337
2h 35m     329
          ...
31h 30m      1
30h 25m      1
42h 5m       1
4h 10m       1
```

```
47h 40m      1
Name: Duration, Length: 368, dtype: int64
```

```
df_train.shape
```

```
(10683, 11)
```

```
df_train.dropna(inplace = True)
```

```
df_train.shape
```

```
(10682, 11)
```

```
df_train.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
Price             0
dtype: int64
```

```
#creating new features
df_train["Journey_day"]=pd.to_datetime(df_train.Date_of_Journey, format='%d/%m/%Y').dt.day
df_train["Journey_month"]=pd.to_datetime(df_train.Date_of_Journey, format='%d/%m/%Y').dt.month
df_train.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

```
df_train.head()
```

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

```
df_train["Dep_hour"]=pd.to_datetime(df_train.Dep_Time).dt.hour
df_train["Dep_minute"]=pd.to_datetime(df_train.Dep_Time).dt.minute
df_train.drop(["Dep_Time"],axis=1,inplace=True)
df_train.head()
```

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	De
0	IndiGo	Banglore	New Delhi	BLR → DEL	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	13:15	7h 25m	2 stops	No info	7662	1	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	04:25 10 Jun	19h	2 stops	No info	13882	9	6	

```
df_train["Arrival_hour"]=pd.to_datetime(df_train.Arrival_Time).dt.hour
df_train["Arrival_minute"]=pd.to_datetime(df_train.Arrival_Time).dt.minute
df_train.drop(["Arrival_Time"], axis = 1, inplace = True)
df_train.head()
```

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_mi
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16	

```
duration = list(df_train["Duration"])


duration_hours = []
duration_minutes = []

# Loop through the 'Duration' column
for duration in df_train['Duration']:
    parts = duration.replace('h', '').replace('m', '').split()  # Remove 'h' and 'm', then split
    if len(parts) == 2:
        hours, minutes = map(int, parts)
    elif 'h' in duration:
        hours = int(parts[0])
        minutes = 0
    elif 'm' in duration:
        hours = 0
        minutes = int(parts[0])
    else:
        hours = 0
        minutes = 0

    duration_hours.append(hours)
    duration_minutes.append(minutes)
```

```
# Print hours and minutes
print("Hours:", hours)
print("Minutes:", minutes)
```

```
Hours: 8
Minutes: 20
```

```
df_train["Duration_hours"] = duration_hours
```

```
df_train["Duration_mins"] = duration_minutes
```

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

```
df_train.head()
```

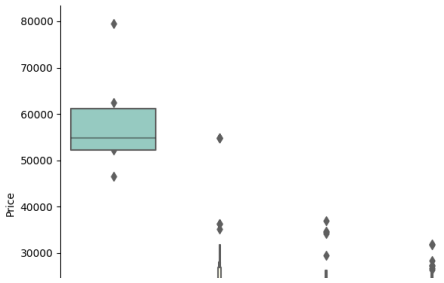
	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_minute	Arri
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	50	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	9	6	9	25	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	12	5	18	5	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	1	3	16	50	

HANDLING CATEGORICAL DATA

```
df_train["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
```

```
sns.catplot(y = "Price", x = "Airline", data = df_train.sort_values("Price", ascending = False), kind="boxen",palette='Set3' ,height = 6,
plt.show()
```



Here we can infer that jet airways business is highest based on its fare

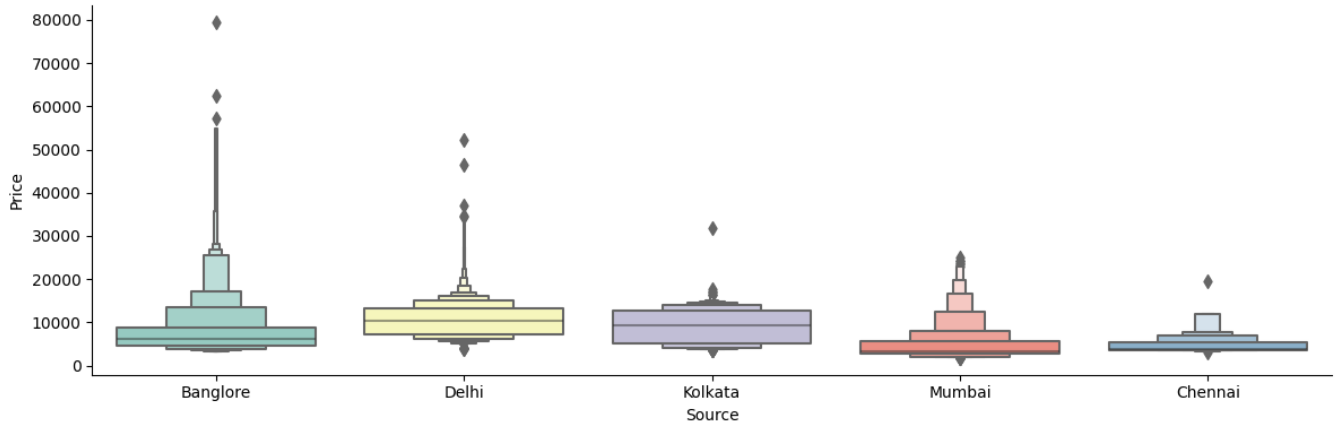

```
Airline = df_train[["Airline"]]

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

Airline.head()
```

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airli
0	0	0	1	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	
2	0	0	0	1	0	0	0	0	
3	0	0	1	0	0	0	0	0	
4	0	0	1	0	0	0	0	0	

```
df_train["Source"].value_counts()
```

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

```
sns.catplot(y = "Price", x = "Source", data = df_train.sort_values("Price", ascending = False), kind="boxen",palette='Set3' , height = 4,
plt.show()
```



From the graph we can infer that banglore is bussiest airport with high prices.

```
Source = df_train[["Source"]]

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

Source.head()
```

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

```
df_train["Destination"].value_counts()
```

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

```
Destination = df_train[["Destination"]]
Destination = pd.get_dummies(Destination, drop_first = True)
```

```
Destination.head()
```

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

```
df_train["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: 10682, dtype: object
```

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

```
df_train.isna().sum()
```

```
Airline           0
Source            0
Destination       0
Total_Stops       0
Price             0
Journey_day       0
Journey_month     0
Dep_hour          0
Dep_minute        0
Arrival_hour      0
Arrival_minute    0
Duration_hours    0
Duration_mins     0
dtype: int64
```

```
df_train["Total_Stops"].value_counts()
```

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

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

```
df_train.head()
```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_minute	Arrival_hour	Arrival_minute
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35

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

```
df_train.head()
```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_minute	Arrival_hour	...	Airline_V Premium e
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	...	
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	...	
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	...	
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	...	
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	...	

5 rows × 33 columns

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

```
df_train.head()
```

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_minute	Arrival_hour	Arrival_minute	Duration_hours	Duration_mins
0	0	3897	24	3	22	20	1	10	2	50
1	2	7662	1	5	5	50	13	15	7	25
2	2	13882	9	6	9	25	4	25	19	0
3	1	6218	12	5	18	5	23	30	5	25
4	1	13302	1	3	16	50	21	35	4	45

5 rows × 30 columns

```
df_train.shape
```

```
(10682, 30)
```

TEST DATA

```
test_data = pd.read_excel('/content/Test_set.xlsx')
```

```
test_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info

```
test_data.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
```

```
test_data.dropna(inplace = True)
```

```
print(test_data.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
```

```
#creating new features
test_data["Journey_day"]=pd.to_datetime(test_data.Date_of_Journey, format='%d/%m/%Y').dt.day
test_data["Journey_month"]=pd.to_datetime(test_data.Date_of_Journey, format='%d/%m/%Y').dt.month
```

```
test_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Journey_da
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info	
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info	1
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included	2
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info	2
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info	2

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

```
test_data.head()
```

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Journey_day	Journey_month
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info	6	6
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info	12	5
2	Jet Airways	Delhi	Cochin	DEL → BOM	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included	21	5

```
test_data["Dep_hour"]=pd.to_datetime(test_data.Dep_Time).dt.hour
test_data["Dep_minute"]=pd.to_datetime(test_data.Dep_Time).dt.minute
test_data.drop(["Dep_Time"],axis=1,inplace=True)
test_data.head()
```

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Journey_day	Journey_month	Dep_hour
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	04:25 07 Jun	10h 55m	1 stop	No info	6	6	17
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	10:20	4h	1 stop	No info	12	5	6
2	Jet Airways	Delhi	Cochin	DEL → BOM → COK	19:00 22 May	23h 45m	1 stop	In-flight meal not included	21	5	19
3	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	21:00	13h	1 stop	No info	21	5	8
4	Air Asia	Banglore	Delhi	BLR → DEL	02:45 25 Jun	2h 50m	non-stop	No info	24	6	23

```
test_data["Arrival_hour"]=pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_minute"]=pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)
test_data.head()
```

```
Airline  Source  Destination  Route  Duration  Total Stops  Additional Info  Journey day  Journey month  Dep hour  Dep minute  A
duration = list(test_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))
```

```
test_data["Duration_hours"] = duration_hours
```

```
test_data["Duration_mins"] = duration_mins
```

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

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Journey_day	Journey_month	Dep_hour	Dep_minute	Arrival_hou
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	1 stop	No info	6	6	17	30	
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	1 stop	No info	12	5	6	20	1
2	Jet Airways	Delhi	Cochin	DEL → BOM → COK	1 stop	In-flight meal not included	21	5	19	15	1
3	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	1 stop	No info	21	5	8	0	2
4	Air Asia	Banglore	Delhi	BLR → DEL	non-stop	No info	24	6	23	55	

```
test_data["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
Vistara Premium economy                2
Jet Airways Business                   2
Name: Airline, dtype: int64
```

```
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
Airline.head()
```

Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	Multiple carriers Premium economy	SpiceJet	Vistara	Vistara Premium economy

```
test_data["Source"].value_counts()
```

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

```
Source = pd.get_dummies(test_data["Source"], drop_first= True)
```

```
Source
```

	Chennai	Delhi	Kolkata	Mumbai
0	0	1	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	0	0
4	0	0	0	0
...	...	...	...	...
2666	0	0	1	0
2667	0	0	1	0
2668	0	1	0	0
2669	0	1	0	0
2670	0	1	0	0

2671 rows × 4 columns

```
test_data["Destination"].value_counts()
```

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

```
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)
```

```
Destination.head()
```

	Cochin	Delhi	Hyderabad	Kolkata	New Delhi
0	1	0	0	0	0
1	0	0	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	0	1	0	0	0

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

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

```
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)
```

```
data_test.head()
```

	Airline	Source	Destination	Total_Stops	Journey_day	Journey_month	Dep_hour
0	Jet Airways	Delhi	Cochin	1	6	6	17
1	IndiGo	Kolkata	Banglore	1	12	5	6
2	Jet Airways	Delhi	Cochin	1	21	5	19
3	Multiple	Delhi	Cochin	1	21	5	8

```
data_test.shape
```

```
(2671, 31)
```

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

```
data_test.head()
```

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_minute	Arrival_hour	Arriv
0	1	6	6	17	30	4	
1	1	12	5	6	20	10	
2	1	21	5	19	15	19	
3	1	21	5	8	0	21	
4	0	24	6	23	55	2	

5 rows × 28 columns

FEATURE SELECTION

```
df_train.shape
```

```
(10682, 30)
```

```
df_train.columns
```

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_minute', 'Arrival_hour', 'Arrival_minute', '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'],
      dtype='object')
```

```
X_train = df_train[['Total_Stops', 'Journey_day', 'Journey_month',
            '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']]
```

```
X_train.columns
```

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

```
X_train.head()
```

	Total_Stops	Journey_day	Journey_month	Airline_Air India	Airline_GoAir	Airline_Indi(
0	0	24	3	0	0	
1	2	1	5	1	0	
2	2	9	6	0	0	
3	1	12	5	0	0	
4	1	1	3	0	0	

5 rows × 23 columns

```
y_train = df_train['Price']
```

```
y_train.head()
```

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

```
print(X_train.shape)
print(y_train.shape)
```

```
(10682, 23)
(10682,)
```

```
data_test.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour', 'Dep_minute',
       'Arrival_hour', 'Arrival_minute', 'Duration_hours', 'Duration_mins',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi'],
      dtype='object')
```

```
X_test = data_test[['Total_Stops', 'Journey_day', 'Journey_month','Duration_hours','Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Ai
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi']]
```

```
X_train,X_test,y_train,y_test=train_test_split(X_train[:8011],y_train[:8011],test_size=0.25,random_state=42)
```

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(6008, 23)
(2003, 23)
(6008,)
(2003,)
```

```
X_train.dtypes
```

```
Total_Stops                                int64
Journey_day                                int64
Journey_month                              int64
Airline_Air India                          uint8
Airline_GoAir                              uint8
Airline_IndiGo                             uint8
Airline_Jet Airways                        uint8
Airline_Jet Airways Business               uint8
Airline_Multiple carriers                  uint8
Airline_Multiple carriers Premium economy  uint8
Airline_SpiceJet                           uint8
Airline_Trujet                             uint8
Airline_Vistara                            uint8
Airline_Vistara Premium economy            uint8
Source_Chennai                             uint8
```

```
Source_Delhi                              uint8
Source_Kolkata                            uint8
Source_Mumbai                             uint8
Destination_Cochin                        uint8
Destination_Delhi                         uint8
Destination_Hyderabad                     uint8
Destination_Kolkata                       uint8
Destination_New Delhi                     uint8
dtype: object
```

```
y_train.dtypes
```

```
dtype('int64')
```

X_train to numpy array

```
columns_to_convert = ['Total_Stops', 'Journey_day', 'Journey_month','Duration_hours', '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_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']

# Convert specified columns to NumPy array
X_train_ = df_train[columns_to_convert].values

# 'data_array' is now a NumPy array containing the specified columns
print((X_train_))
```

```
[[ 0 24  3 ...  0  0  1]
 [ 2  1  5 ...  0  0  0]
 [ 2  9  6 ...  0  0  0]
 ...
 [ 0 27  4 ...  0  0  0]
 [ 0  1  3 ...  0  0  1]
 [ 2  9  5 ...  0  0  0]]
```

```
X_train_.dtype
```

```
dtype('int64')
```

```
X_train_.shape
```

```
(10682, 23)
```

```
y_train
```

```
1872     4823
5306     3943
2664    11299
4674     6600
57       7414
        ...
5226    10676
5390     7670
860      6144
7603     2754
7270    10262
Name: Price, Length: 6008, dtype: int64
```

y_train to numpy array

```
y_train_array = y_train.values
```

```
print(y_train_array)
```

```
[ 4823  3943 11299 ...  6144  2754 10262]
```

```
y_train_ = y_train_array[:6008]
```

```
print(y_train_.shape)
```

```
(6008,)
```

```
y_train_.dtype
```

```
dtype('int64')
```

X_test to numpy array

```
data_test.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour', 'Dep_minute',
       'Arrival_hour', 'Arrival_minute', 'Duration_hours', 'Duration_mins',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi'],
      dtype='object')
```

```
X_test = data_test[['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi']]
```

```
X_test.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Delhi',
       'Kolkata', 'Kolkata', 'Mumbai', 'Cochin', 'Delhi', 'Delhi', 'Hyderabad',
       'Kolkata', 'Kolkata', 'New Delhi'],
      dtype='object')
```

```
df = pd.DataFrame(X_test)

# Extract specific columns and convert to NumPy array
columns_to_extract = ['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi',
       'Kolkata',  'Mumbai', 'Cochin', 'Delhi', 'Hyderabad',
       'Kolkata',  'New Delhi']
X_test_ = df[columns_to_extract].values

# 'data_array' is now a NumPy array containing the specified columns' values
print(X_test_)
```

```
[[ 1  6  6 ...  0  0  0]
 [ 1 12  5 ...  1  0  0]
 [ 1 21  5 ...  0  0  0]
 ...
 [ 1  6  3 ...  0  0  0]
 [ 1  6  3 ...  0  0  0]
 [ 1 15  6 ...  0  0  0]]
```

```
X_test_.shape
```

```
(2671, 35)
```

```
y_test.shape
```

```
(2003,)
```

y_test to numpy array

```
y_test_array = y_test.values
```

```
print(y_test_array)
```

```
[ 7229 15136  8728 ...  8586 15554  3332]
```

```
y_test_ = y_test_array[:6008]
```

```
print(y_test_.shape)
```

```
(2003,)
```

```
print(X_train_.shape)
print(X_test_.shape)
print(y_train_.shape)
print(y_test_.shape)
```

```
(10682, 23)
(2671, 35)
(6008,)
(2003,)
```

```
X_train_,X_test_,y_train_,y_test_=train_test_split(X_train_[:3379],y_train_[:3379],test_size=0.25,random_state=42)
```

```
print(X_train_.shape)
print(X_test_.shape)
print(y_train_.shape)
print(y_test_.shape)
```

```
(2534, 23)
(845, 23)
(2534,)
(845,)
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import r2_score
```

```
knn_model = KNeighborsRegressor(n_neighbors=3)
knn_model.fit(X_train_, y_train_)

predictions = knn_model.predict(X_test_)
r2_knn = r2_score(y_test_, predictions)
print(f'R-squared Score for KNN Regression: {r2_knn}')
```

```
R-squared Score for KNN Regression: -0.23895727128083522
```

ACCURACY USING KNN

```
knn_classifier = KNeighborsClassifier(n_neighbors=3)
knn_classifier.fit(X_train_, y_train_)
predictions = knn_classifier.predict(X_test_)
error_rate = 1 - accuracy_score(y_test_, predictions)
print("Error Rate (Accuracy): {:.2f}%".format(error_rate * 100))
```

```
Error Rate (Accuracy): 99.29%
```

```
import pickle
from sklearn.neighbors import KNeighborsRegressor
import joblib
```

```
knn_model = KNeighborsRegressor(n_neighbors=3)

# Train the model
knn_model.fit(X_train_, y_train_)

# Save the trained model to a file using pickle
joblib.dump(knn_model, 'knn_model.pkl')

print("KNN model has been trained and saved as 'knn_model.pkl'")
```

```
KNN model has been trained and saved as 'knn_model.pkl'
```

APPLYING LINEAR REGRESSION

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import r2_score
```

```
df_train.shape
```

```
(10682, 30)
```

```
df_train.columns
```

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_minute', 'Arrival_hour', 'Arrival_minute', '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'],
      dtype='object')
```

```
X_train_reg = df_train[[ 'Total_Stops', 'Journey_day',
       'Journey_month',
       'Duration_hours', '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_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']]
```

```
X_train_reg.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       '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_Vistara', 'Airline_Vistara Premium economy', 'Source_Chennai',
       'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
       'Destination_New Delhi'],
      dtype='object')
```

```
X_train_reg.head()
```

	Total_Stops	Journey_day	Journey_month	Duration_hours	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Ai
0	0	24	3	2	0	0	1	0	0	
1	2	1	5	7	1	0	0	0	0	
2	2	9	6	19	0	0	0	1	0	
3	1	12	5	5	0	0	1	0	0	
4	1	1	3	4	0	0	1	0	0	

5 rows × 23 columns

```
data_test.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour', 'Dep_minute',
       'Arrival_hour', 'Arrival_minute', 'Duration_hours', 'Duration_mins',
       'Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet Airways Business',
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi'],
      dtype='object')
```

```
X_test_reg=data_test[['Total_Stops', 'Journey_day', 'Journey_month','Duration_hours','Air India', 'GoAir', 'IndiGo', 'Jet Airways', 'Jet
       'Multiple carriers', 'Multiple carriers Premium economy', 'SpiceJet',
       'Vistara', 'Vistara Premium economy', 'Chennai', 'Delhi', 'Kolkata',
       'Mumbai', 'Cochin', 'Delhi', 'Hyderabad', 'Kolkata', 'New Delhi']]
```

```
X_test_reg
```

	Total_Stops	Journey_day	Journey_month	Duration_hours	Air India	GoAir	IndiGo	Jet Airways	Jet Airways Business	Multiple carriers	...	Kolkata	Kolk
0	1	6	6	10	0	0	0	1	0	0	...	0	
1	1	12	5	4	0	0	1	0	0	0	...	1	
2	1	21	5	23	0	0	0	1	0	0	...	0	
3	1	21	5	13	0	0	0	0	0	1	...	0	
4	0	24	6	2	0	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
2666	1	6	6	23	1	0	0	0	0	0	...	1	
2667	0	27	3	2	0	0	1	0	0	0	...	1	

```
y_train_reg = df_train['Price']
```

```
y_train_reg.head()
```

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

```
y_test
```

```
554      7229
3872    15136
3103     8728
7049     7934
7861     4174
        ...
3989     5694
6805     7608
3044     8586
5314    15554
705      3332
Name: Price, Length: 2003, dtype: int64
```

```
y_test_reg=df_train['Price']
```

```
print(X_train_reg.shape)
print(X_test_reg.shape)
print(y_train_reg.shape)
print(y_test_reg.shape)
```

```
(10682, 23)
(2671, 27)
(10682,)
(10682,)
```

```
X_train_reg,X_test_reg,y_train_reg,y_test_reg=train_test_split(X_train_reg[:2671],y_train_reg[:2671],test_size=0.25,random_state=42)
```

```
X_train_reg.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       '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_Vistara', 'Airline_Vistara Premium economy', 'Source_Chennai',
       'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
       'Destination_New Delhi'],
      dtype='object')
```

```
y_train_reg
```

```
2565    14120
2662    11071
1452     3384
278      8991
1927    17024
        ...
```

```
1638     9663
1095    10877
1130     6216
1294    10413
860      6144
Name: Price, Length: 2003, dtype: int64
```

```
X_test_reg.columns
```

```
Index(['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       '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_Vistara', 'Airline_Vistara Premium economy', 'Source_Chennai',
       'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
       'Destination_New Delhi'],
      dtype='object')
```

```
model = LinearRegression()
model.fit(X_train_reg, y_train_reg)
```

▾ LinearRegression

LinearRegression()

```
y_pred_reg = model.predict(X_test_reg)
```

```
r2_reg = r2_score(y_test_reg, y_pred_reg)
print("The RSquared test for the Flight Fare Prediction model is: ", r2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-150-94fcd38247ee> in <cell line: 2>()
      1 r2_reg = r2_score(y_test_reg, y_pred_reg)
----> 2 print("The RSquared test for the Flight Fare Prediction model is: ", r2)

NameError: name 'r2' is not defined
```

SEARCH STACK OVERFLOW

R-squared value of 0.5204 suggests that our model captures a moderate amount of the variance in the flight fares.

MSE-MEAN SQUARED ERROR

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

```
MSE=mean_squared_error(y_test_reg,y_pred_reg)
print("The Mean squared error for the Flight Fare Prediction model is: ", MSE)
```

```
The Mean squared error for the Flight Fare Prediction model is:  10486936.738824513
```

AT 10486936.738824513 mse value indicate that the model is better at predicting the flight fares, as it means that the predicted values are closer to the actual values on average.

MEAN ABSOLUTE ERROR

```
MAE=mean_absolute_error(y_test_reg,y_pred_reg)
print("The MEAN ABSOLUTE ERROR for the Flight Fare Prediction model is: ", MAE)
```

```
The MEAN ABSOLUTE ERROR for the Flight Fare Prediction model is:  2100.8020156663115
```

The absolute difference between the predicted flight fares and the actual flight fares is 2100.80201566631155.

MEAN ABSOLUTE PERCENTAGE ERROR

```
MAPE = np.mean(np.abs((y_test_reg - y_pred_reg) / y_test)) * 100
print("The MEAN ABSOLUTE PERCENTAGE ERROR for the Flight Fare Prediction model is: ", MAPE)
```

```
The MEAN ABSOLUTE PERCENTAGE ERROR for the Flight Fare Prediction model is:  26.966696030260877
```

A MAPE of 26.966696030260877% suggests that, on average, the model's predictions for flight fares are off by about one-third of the actual fare values. A lower MAPE indicates a more accurate model.

RMSE-ROOT MEAN SQUARED ERROR

```
import numpy as np
```

```
RMSE = np.sqrt(y_test_reg,y_pred_reg)
print("The ROOT MEAN SQUARED ERROR for the Flight Fare Prediction model is: ",  RMSE)
```

```
The ROOT MEAN SQUARED ERROR for the Flight Fare Prediction model is:  2120    121.301278
1211     75.670338
1461    127.628367
1511    104.618354
1237     89.861004
          ...
2586     82.401456
654      52.478567
1713    111.332834
1891    101.823376
1588     95.885348
Name: Price, Length: 668, dtype: float64
```

Lower RMSE values indicate that the model's predictions are closer to the actual values, while higher RMSE values suggest larger prediction errors.

THEIL'S U1 AND U2

By calculating both U1 and U2, WE can gain a understanding of the overall inequality in a dataset.

```
def theil_u1(y_test_reg, y_pred_reg):
    numerator = np.sqrt(np.mean((y_test_reg - y_pred_reg) ** 2))
    denominator = np.sqrt(np.mean(y_test_reg ** 2))
    return numerator / denominator
```

```
U1 = theil_u1(y_test_reg, y_pred_reg)
print(f"Theil's U1: {U1:.2f}%")
```

```
Theil's U1: 0.99%
```

Theil's U1 statistic measures the proportion of total inequality in the data. 0.99% of the overall inequality is due to disparities between groups, indicating a relatively low level of group-based inequality in the dataset.

```
def theil_u2(y_test_reg, y_pred_reg):
    numerator = np.sqrt(np.mean((y_test_reg - y_pred_reg) ** 2))
    denominator = np.sqrt(np.mean(y_test_reg ** 2))
    return numerator / denominator * 100
```

```
U2 = theil_u2(y_test_reg, y_pred_reg)
print(f"Theil's U2: {U2:.2f}%")
```

```
Theil's U2: 99.09%
```

A Theil's U2 statistic of 99.09% suggests that 99.09% of the total inequality in the data is due to disparities within groups, rather than between groups.

IOA-INDEX OF AGREEMENT

```
def index_of_agreement(y_test_reg, y_pred_reg):
    y_mean = np.mean(y_test_reg)
    numerator = np.sum((y_test_reg - y_pred_reg) ** 2)
    denominator = np.sum((np.abs(y_pred_reg - y_mean) + np.abs(y_test_reg - y_mean)) ** 2)
    return 1 - (numerator / denominator)
```

```
ioa = index_of_agreement(y_test_reg, y_pred_reg)
print(f'Index of Agreement (IOA): {ioa:.2f}')
```

```
Index of Agreement (IOA): 0.39
```

An Index of Agreement (IOA) value of 0.39 suggests a relatively low level of agreement between observed data and model predictions.

TEST FOR HETEROSCADASTICITY(BREUSCH PAGAN TEST)

```
import statsmodels.api as sm
from statsmodels.stats.diagnostic import het_breuschpagan



X_train_reg = pd.DataFrame(X_train_reg, columns=['Total_Stops', 'Journey_day', 'Journey_month', 'Duration_hours',
       '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_Vistara', 'Airline_Vistara Premium economy', 'Source_Chennai',
       'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
       'Destination_New Delhi'])

X_train_reg = sm.add_constant(X_train_reg)


model = sm.OLS(y_train_reg, X_train_reg).fit()


_, p_value, _, _ = het_breuschpagan(model.resid, X_train_reg)
print(f"P-value from Breusch-Pagan test: {p_value}")


if p_value > 0.05:
    print("The data does not exhibit heteroscedasticity.")
else:
    print("The data exhibits heteroscedasticity.")
```

```
P-value from Breusch-Pagan test: 1.1863477299362613e-12
The data exhibits heteroscedasticity.
```

p-value of 1.1863477299362613e-12 is extremely small, indicating strong evidence against the null hypothesis of homoskedasticity (constant variance of residuals). residuals basically represent errors made by model in predictions. [**IMP**:indicating that the model's predictions are equally accurate for different ranges of the independent variables.]

DURBIN WATSON TEST

```
import statsmodels.api as sm
from statsmodels.stats.stattools import durbin_watson



X_train_const = sm.add_constant(X_train_reg)


model = sm.OLS(y_train_reg, X_train_const).fit()


durbin_watson_stat = durbin_watson(model.resid)

print(f"Durbin-Watson Statistic: {durbin_watson_stat}")
```

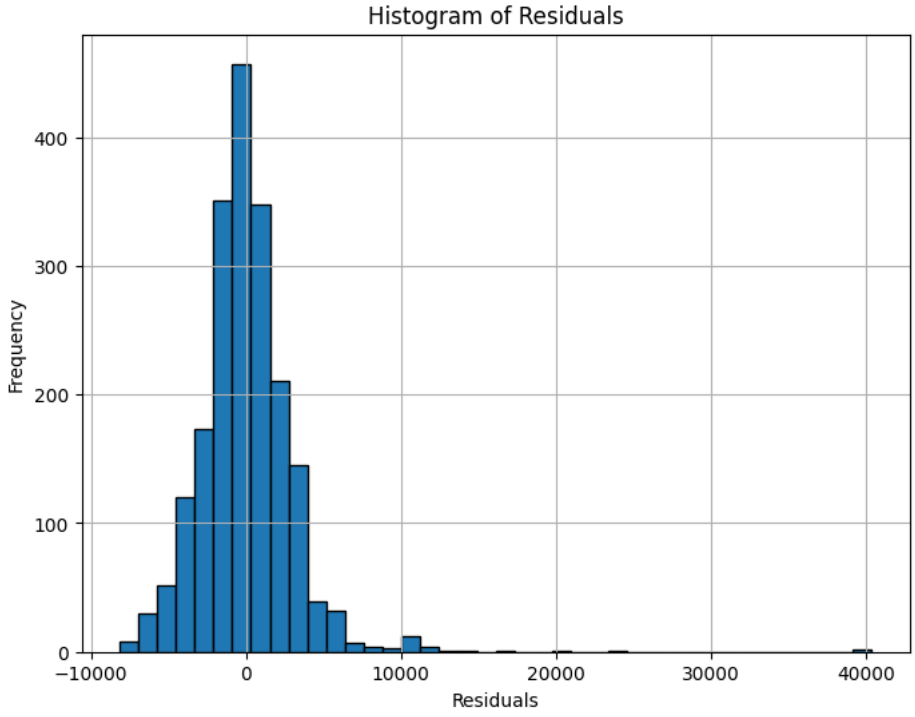
```
Durbin-Watson Statistic: 2.012276384363311
```

HERE Durbin-Watson Statistic: 2.012276384363311 which means that there is no significant autocorrelation in the residuals. This is a good result because it indicates that the residuals (errors) in your regression model are independent of each other.

HISTOGRAM FOR RESIDUALS

```
import matplotlib.pyplot as plt


y_pred_train = model.predict(X_train_reg)


residuals = (y_train_reg - y_pred_train)
```

```
plt.figure(figsize=(8, 6))
plt.hist(residuals, bins=40, edgecolor='black')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.grid(True)
plt.show()
```



1.The majority of the residuals fall within the range of -10,000 to 10,000, it suggests that most of your model's predictions have errors within this range.

2.The bulk of your model's predictions have errors centered around zero, indicating that, on average, your model tends to predict values close to the actual flight fares.

Random Forest Regressor*

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
df_train.columns
```

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
       'Dep_minute', 'Arrival_hour', 'Arrival_minute', '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'],
      dtype='object')
```

```
X_train_rf = df_train[['Total_Stops','Journey_day', 'Journey_month',
        'Duration_hours',
        '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_Vistara', 'Airline_Vistara Premium economy',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']].values
```

```
y_train_rf=df_train['Price']
```

```
X_train_rf,X_test_rf,y_train_rf,y_test_rf=train_test_split(X_train_rf,y_train_rf,test_size=0.25,random_state=42)
```

```
print(X_train_rf.shape)
print(X_test_rf.shape)
print(y_train_rf.shape)
print(y_test_rf.shape)
```

```
(8011, 23)
(2671, 23)
(8011,)
(2671,)
```

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
model.fit(X_train_rf, y_train_rf)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
predictions = model.predict(X_test_rf)
```

```
mse = mean_squared_error(y_test_rf, predictions)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 5531979.774784286
```

```
y_pred_rf = model.predict(X_test_rf)
```

```
r2_rf= r2_score(y_test_rf, y_pred_rf)
print("The RSquared test for the Flight Fare Prediction model is: ", r2_rf)
```

```
The RSquared test for the Flight Fare Prediction model is:  0.7316269982655875
```

COMPARISIONS

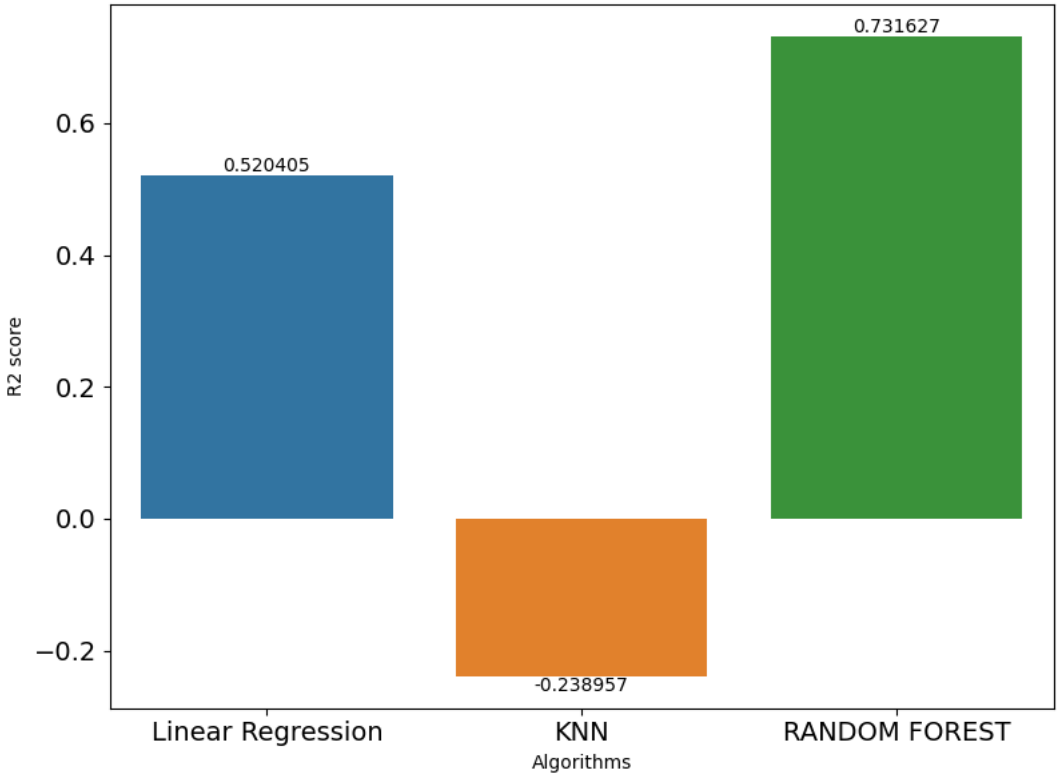
```
scores=[r2_reg,r2_knn,r2_rf]
algorithms=["Linear Regression","KNN","RANDOM FOREST"]
```

```
for i in range (len(algorithms)):
  print("The R2 score achieved using "+algorithms[i]+ "is: "+str(scores[i])+"%")
```

```
The R2 score achieved using Linear Regressionis: 0.5204047804851053%
The R2 score achieved using KNNis: -0.2389572712808352 2%
The R2 score achieved using RANDOM FORESTis: 0.7316269982655875%
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(8,6))
plt.xlabel("Algorithms")
plt.ylabel("R2 score")
ax=sns.barplot(x=algorithms,y=scores)
for label in ax.containers:
  ax.bar_label(label)
plt.tight_layout()
plt.tick_params(labelsize=14)
```


0.731627
0.6
0.520405
0.4
0.2
R2 score
0.0
−0.2
-0.238957
Linear Regression
KNN
RANDOM FOREST
Algorithms