# PROBLEM STATEMENT: Which model is suitable for flight price prediction dataset

### Importing packages

In [1]:

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

### Data Collection

In [2]:

```
train_df=pd.read_csv(r"C:\Users\Teju\Downloads\Data_Train-Flight.csv")
train_df
```

Out[2]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Dura |
|---|---|---|---|---|---|---|---|---|
| **0** | IndiGo | 24/03/2019 | Banglore | New Delhi | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h |
| **1** | Air India | 1/05/2019 | Kolkata | Banglore | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h |
| **2** | Jet Airways | 9/06/2019 | Delhi | Cochin | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | |
| **3** | IndiGo | 12/05/2019 | Kolkata | Banglore | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h |
| **4** | IndiGo | 01/03/2019 | Banglore | New Delhi | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **10678** | Air Asia | 9/04/2019 | Kolkata | Banglore | CCU ? BLR | 19:55 | 22:25 | 2h |
| **10679** | Air India | 27/04/2019 | Kolkata | Banglore | CCU ? BLR | 20:45 | 23:20 | 2h |
| **10680** | Jet Airways | 27/04/2019 | Banglore | Delhi | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | Vistara | 01/03/2019 | Banglore | New Delhi | BLR ? DEL | 11:30 | 14:10 | 2h |
| **10682** | Air India | 9/05/2019 | Delhi | Cochin | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h |

10683 rows × 11 columns

In [3]:

```
test_df=pd.read_csv(r"C:\Users\Teju\Downloads\Test_set-Flight.csv")
test_df
```

Out[3]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Durat |
|---|---|---|---|---|---|---|---|---|
| **0** | Jet Airways | 6/06/2019 | Delhi | Cochin | DEL ? BOM ? COK | 17:30 | 04:25 07 Jun | 10h 5 |
| **1** | IndiGo | 12/05/2019 | Kolkata | Banglore | CCU ? MAA ? BLR | 06:20 | 10:20 | |
| **2** | Jet Airways | 21/05/2019 | Delhi | Cochin | DEL ? BOM ? COK | 19:15 | 19:00 22 May | 23h 4 |
| **3** | Multiple carriers | 21/05/2019 | Delhi | Cochin | DEL ? BOM ? COK | 08:00 | 21:00 | |
| **4** | Air Asia | 24/06/2019 | Banglore | Delhi | BLR ? DEL | 23:55 | 02:45 25 Jun | 2h 5 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **2666** | Air India | 6/06/2019 | Kolkata | Banglore | CCU ? DEL ? BLR | 20:30 | 20:25 07 Jun | 23h 5 |
| **2667** | IndiGo | 27/03/2019 | Kolkata | Banglore | CCU ? BLR | 14:20 | 16:55 | 2h 3 |
| **2668** | Jet Airways | 6/03/2019 | Delhi | Cochin | DEL ? BOM ? COK | 21:50 | 04:25 07 Mar | 6h 3 |
| **2669** | Air India | 6/03/2019 | Delhi | Cochin | DEL ? BOM ? COK | 04:00 | 19:15 | 15h 1 |
| **2670** | Multiple carriers | 15/06/2019 | Delhi | Cochin | DEL ? BOM ? COK | 04:55 | 19:15 | 14h 2 |

2671 rows × 10 columns

### Data cleaning and preprocessing

In [4]:

```
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
```

In [5]:

```
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
```

In [6]:

```
train_df.describe()
```

Out[6]:

| | 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 |

In [7]:

```
test_df.describe()
```

Out[7]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Dura |
|---|---|---|---|---|---|---|---|---|
| **count** | 2671 | 2671 | 2671 | 2671 | 2671 | 2671 | 2671 | 2 |
| **unique** | 11 | 44 | 5 | 6 | 100 | 199 | 704 | |
| **top** | Jet Airways | 9/05/2019 | Delhi | Cochin | DEL ? BOM ? COK | 10:00 | 19:00 | 2h |
| **freq** | 897 | 144 | 1145 | 1145 | 624 | 62 | 113 | |

In [8]:

```
train_df.head()
```

Out[8]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Duration |
|---|---|---|---|---|---|---|---|---|
| **0** | IndiGo | 24/03/2019 | Banglore | New Delhi | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h 50m |
| **1** | Air India | 1/05/2019 | Kolkata | Banglore | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h 25m |
| **2** | Jet Airways | 9/06/2019 | Delhi | Cochin | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | 19h |
| **3** | IndiGo | 12/05/2019 | Kolkata | Banglore | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h 25m |
| **4** | IndiGo | 01/03/2019 | Banglore | New Delhi | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h 45m |

In [9]:

```
test_df.head()
```

Out[9]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Duration |
|---|---|---|---|---|---|---|---|---|
| **0** | Jet Airways | 6/06/2019 | Delhi | Cochin | DEL ? BOM ? COK | 17:30 | 04:25 07 Jun | 10h 55m |
| **1** | IndiGo | 12/05/2019 | Kolkata | Banglore | CCU ? MAA ? BLR | 06:20 | 10:20 | 4h |
| **2** | Jet Airways | 21/05/2019 | Delhi | Cochin | DEL ? BOM ? COK | 19:15 | 19:00 22 May | 23h 45m |
| **3** | Multiple carriers | 21/05/2019 | Delhi | Cochin | DEL ? BOM ? COK | 08:00 | 21:00 | 13h |
| **4** | Air Asia | 24/06/2019 | Banglore | Delhi | BLR ? DEL | 23:55 | 02:45 25 Jun | 2h 50m |

In [10]:

```
train_df.tail()
```

Out[10]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Dura |
|---|---|---|---|---|---|---|---|---|
| **10678** | Air Asia | 9/04/2019 | Kolkata | Banglore | CCU ? BLR | 19:55 | 22:25 | 2h |
| **10679** | Air India | 27/04/2019 | Kolkata | Banglore | CCU ? BLR | 20:45 | 23:20 | 2h |
| **10680** | Jet Airways | 27/04/2019 | Banglore | Delhi | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | Vistara | 01/03/2019 | Banglore | New Delhi | BLR ? DEL | 11:30 | 14:10 | 2h |
| **10682** | Air India | 9/05/2019 | Delhi | Cochin | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h |

In [11]:

```
test_df.tail()
```

Out[11]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Duratic |
|---|---|---|---|---|---|---|---|---|
| **2666** | Air India | 6/06/2019 | Kolkata | Banglore | CCU ? DEL ? BLR | 20:30 | 20:25 07 Jun | 23h 55 |
| **2667** | IndiGo | 27/03/2019 | Kolkata | Banglore | CCU ? BLR | 14:20 | 16:55 | 2h 35 |
| **2668** | Jet Airways | 6/03/2019 | Delhi | Cochin | DEL ? BOM ? COK | 21:50 | 04:25 07 Mar | 6h 35 |
| **2669** | Air India | 6/03/2019 | Delhi | Cochin | DEL ? BOM ? COK | 04:00 | 19:15 | 15h 15 |
| **2670** | Multiple carriers | 15/06/2019 | Delhi | Cochin | DEL ? BOM ? COK | 04:55 | 19:15 | 14h 20 |

In [12]:

```
train_df.shape
```

Out[12]:

```
(10683, 11)
```

In [13]:

```
test_df.shape
```

Out[13]:

```
(2671, 10)
```

In [14]:

```
train_df.columns
```

Out[14]:

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

In [15]:

```
test_df.columns
```

Out[15]:

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

**Find the Null values**

In [16]:

```
train_df.isnull().sum()
```

Out[16]:

```
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
```

In [17]:

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

Out[17]:

```
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 [18]:

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

In [19]:

```
train_df.isnull().sum()
```

Out[19]:

```
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
```

In [20]:

```
train_df["Airline"].value_counts()
```

Out[20]:

```
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
```

In [21]:

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

Out[21]:

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

In [22]:

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

Out[22]:

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

In [23]:

```
test_df["Airline"].value_counts()
```

Out[23]:

```
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
```

In [24]:

```
AL={"Airline":{"Jet Airways":0,"IndiGo":1,"Air India":2,"Multiple carriers":3,"SpiceJet"
train_df=train_df.replace(AL)
train_df
```

Out[24]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Durat |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 24/03/2019 | Banglore | New Delhi | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h 5 |
| **1** | 2 | 1/05/2019 | Kolkata | Banglore | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h 2 |
| **2** | 0 | 9/06/2019 | Delhi | Cochin | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | |
| **3** | 1 | 12/05/2019 | Kolkata | Banglore | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h 2 |
| **4** | 1 | 01/03/2019 | Banglore | New Delhi | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h 4 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **10678** | 6 | 9/04/2019 | Kolkata | Banglore | CCU ? BLR | 19:55 | 22:25 | 2h 3 |
| **10679** | 2 | 27/04/2019 | Kolkata | Banglore | CCU ? BLR | 20:45 | 23:20 | 2h 3 |
| **10680** | 0 | 27/04/2019 | Banglore | Delhi | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | 5 | 01/03/2019 | Banglore | New Delhi | BLR ? DEL | 11:30 | 14:10 | 2h 4 |
| **10682** | 2 | 9/05/2019 | Delhi | Cochin | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h 2 |

10682 rows × 11 columns

In [25]:

```
source={"Source":{"Delhi":0,"Kolkata":1,"Banglore":2,"Mumbai":3,"Chennai":4}}
train_df=train_df.replace(source)
train_df
```

Out[25]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Durati |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 24/03/2019 | 2 | New Delhi | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h 50 |
| **1** | 2 | 1/05/2019 | 1 | Banglore | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h 25 |
| **2** | 0 | 9/06/2019 | 0 | Cochin | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | 19 |
| **3** | 1 | 12/05/2019 | 1 | Banglore | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h 25 |
| **4** | 1 | 01/03/2019 | 2 | New Delhi | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h 45 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **10678** | 6 | 9/04/2019 | 1 | Banglore | CCU ? BLR | 19:55 | 22:25 | 2h 30 |
| **10679** | 2 | 27/04/2019 | 1 | Banglore | CCU ? BLR | 20:45 | 23:20 | 2h 35 |
| **10680** | 0 | 27/04/2019 | 2 | Delhi | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | 5 | 01/03/2019 | 2 | New Delhi | BLR ? DEL | 11:30 | 14:10 | 2h 40 |
| **10682** | 2 | 9/05/2019 | 0 | Cochin | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h 20 |

10682 rows × 11 columns

In [26]:

```
Dest={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,"New Delhi":3,"Hyderabad":4,"Kolk
train_df=train_df.replace(Dest)
train_df
```

Out[26]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Duratic |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 24/03/2019 | 2 | 3 | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h 50 |
| **1** | 2 | 1/05/2019 | 1 | 1 | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h 25 |
| **2** | 0 | 9/06/2019 | 0 | 0 | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | 1 |
| **3** | 1 | 12/05/2019 | 1 | 1 | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h 25 |
| **4** | 1 | 01/03/2019 | 2 | 3 | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h 45 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **10678** | 6 | 9/04/2019 | 1 | 1 | CCU ? BLR | 19:55 | 22:25 | 2h 30 |
| **10679** | 2 | 27/04/2019 | 1 | 1 | CCU ? BLR | 20:45 | 23:20 | 2h 35 |
| **10680** | 0 | 27/04/2019 | 2 | 2 | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | 5 | 01/03/2019 | 2 | 3 | BLR ? DEL | 11:30 | 14:10 | 2h 40 |
| **10682** | 2 | 9/05/2019 | 0 | 0 | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h 20 |

10682 rows × 11 columns

In [27]:

```
Stop={"Total_Stops":{"non-stop":0,"1 stop":1,"2 stops":2,"3 stops":3,"4 stops":4}}
train_df=train_df.replace(Stop)
train_df
```

Out[27]:

| | Airline | Date_of_Journey | Source | Destination | Route | Dep_Time | Arrival_Time | Durati |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 24/03/2019 | 2 | 3 | BLR ? DEL | 22:20 | 01:10 22 Mar | 2h 50 |
| **1** | 2 | 1/05/2019 | 1 | 1 | CCU ? IXR ? BBI ? BLR | 05:50 | 13:15 | 7h 25 |
| **2** | 0 | 9/06/2019 | 0 | 0 | DEL ? LKO ? BOM ? COK | 09:25 | 04:25 10 Jun | 19 |
| **3** | 1 | 12/05/2019 | 1 | 1 | CCU ? NAG ? BLR | 18:05 | 23:30 | 5h 25 |
| **4** | 1 | 01/03/2019 | 2 | 3 | BLR ? NAG ? DEL | 16:50 | 21:35 | 4h 45 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **10678** | 6 | 9/04/2019 | 1 | 1 | CCU ? BLR | 19:55 | 22:25 | 2h 30 |
| **10679** | 2 | 27/04/2019 | 1 | 1 | CCU ? BLR | 20:45 | 23:20 | 2h 35 |
| **10680** | 0 | 27/04/2019 | 2 | 2 | BLR ? DEL | 08:20 | 11:20 | |
| **10681** | 5 | 01/03/2019 | 2 | 3 | BLR ? DEL | 11:30 | 14:10 | 2h 40 |
| **10682** | 2 | 9/05/2019 | 0 | 0 | DEL ? GOI ? BOM ? COK | 10:55 | 19:15 | 8h 20 |

10682 rows × 11 columns

**Exploratory Data Analysis**

In [28]:

```
df=train_df[['Airline','Source','Destination','Total_Stops','Price']]
sns.heatmap(df.corr(),annot=True)
```

Out[28]:

<Axes: >


In [29]:

```
x=train_df[['Airline','Source','Destination','Total_Stops']]
y=train_df['Price']
```

## Linear Regression

In [30]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=100)
```

In [31]:

```
from sklearn.linear_model import LinearRegression
ln=LinearRegression()
ln.fit(x_train,y_train)
print(ln.intercept_)
```

```
7211.098088897474
```

In [32]:

```
coeff_df=pd.DataFrame(ln.coef_,x.columns,columns=['coefficient'])
coeff_df
```

Out[32]:

| | coefficient |
|---|---|
| **Airline** | -418.483922 |
| **Source** | -3275.073380 |
| **Destination** | 2505.480291 |
| **Total_Stops** | 3541.798053 |

In [33]:

```
score=ln.score(x_test,y_test)
score
```

Out[33]:

```
0.4108304890928345
```

In [34]:

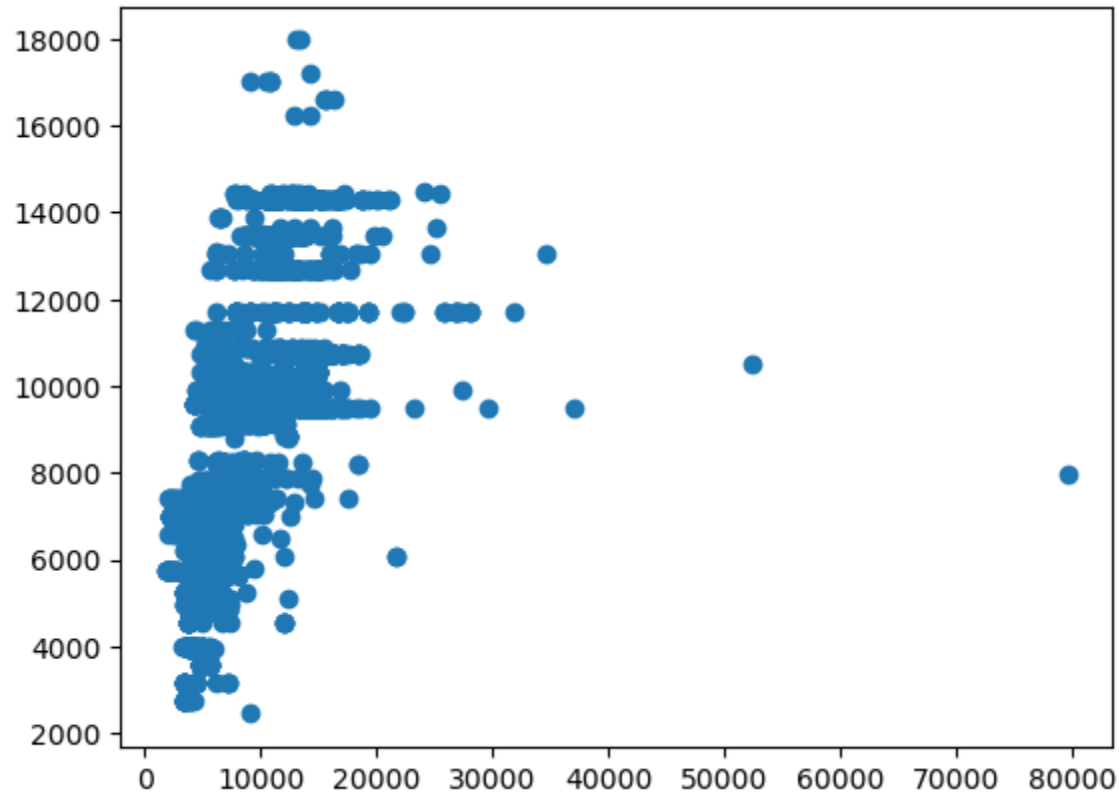
```
predictions=ln.predict(x_test)
```

In [35]:

```
plt.scatter(y_test,predictions)
```

Out[35]:

```
<matplotlib.collections.PathCollection at 0x17b67cf3d60>
```




In [36]:

```
x=np.array(train_df['Price']).reshape(-1,1)
y=np.array(train_df['Total_Stops']).reshape(-1,1)
train_df.dropna(inplace=True)
```

In [37]:

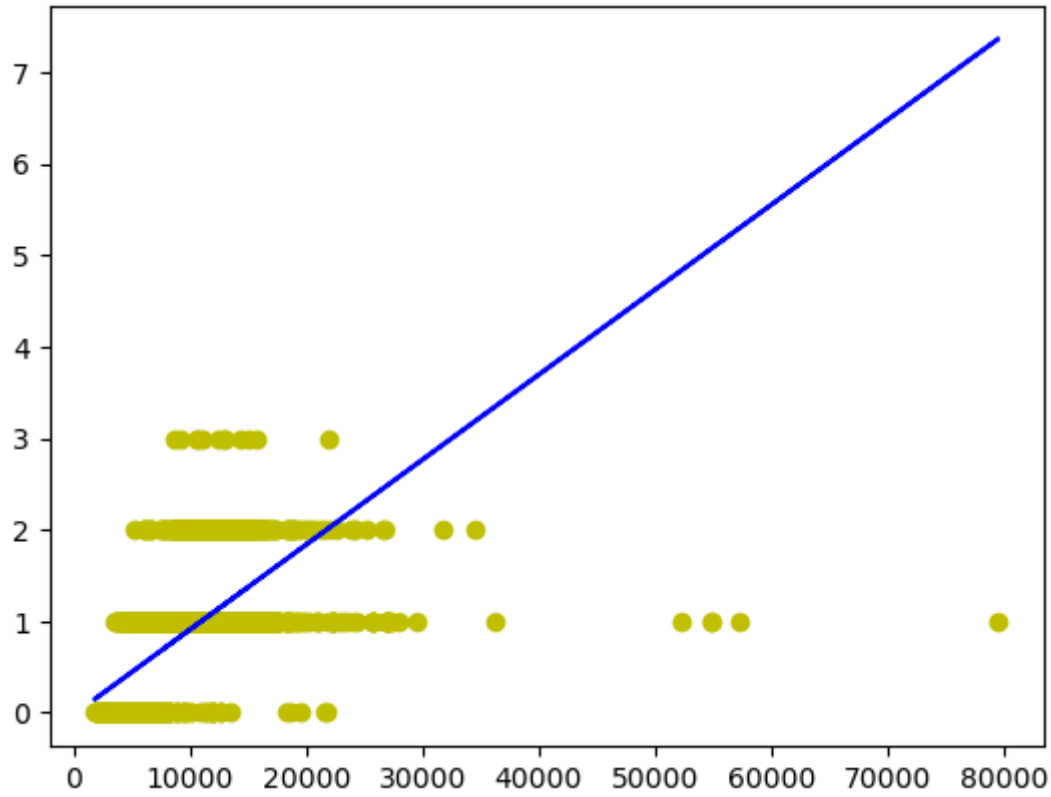
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
ln.fit(x_train,y_train)
ln.fit(x_train,y_train)
```

Out[37]:

```
LinearRegression
LinearRegression()
```

In [38]:

```
y_pred=ln.predict(x_test)
plt.scatter(x_test,y_test,color='y')
plt.plot(x_test,y_pred,color='b')
plt.show()
```




## By using Linear Regression we didn't get the accuracy for this model. So we will use Logistic Regression

## Logistic Regression

In [39]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [40]:

```
x=np.array(df['Price']).reshape(-1,1)
y=np.array(df['Total_Stops']).reshape(-1,1)
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(max_iter=10000)
import warnings
warnings.simplefilter(action='ignore')
```

```
C:\Users\Teju\AppData\Local\Temp\ipykernel_15532\1264944960.py:3: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  df.dropna(inplace=True)
```

In [41]:

```
lr.fit(x_train,y_train)
```

Out[41]:

```
LogisticRegression
LogisticRegression(max_iter=10000)
```

In [42]:

```
score=lr.score(x_test,y_test)
score
```

Out[42]:

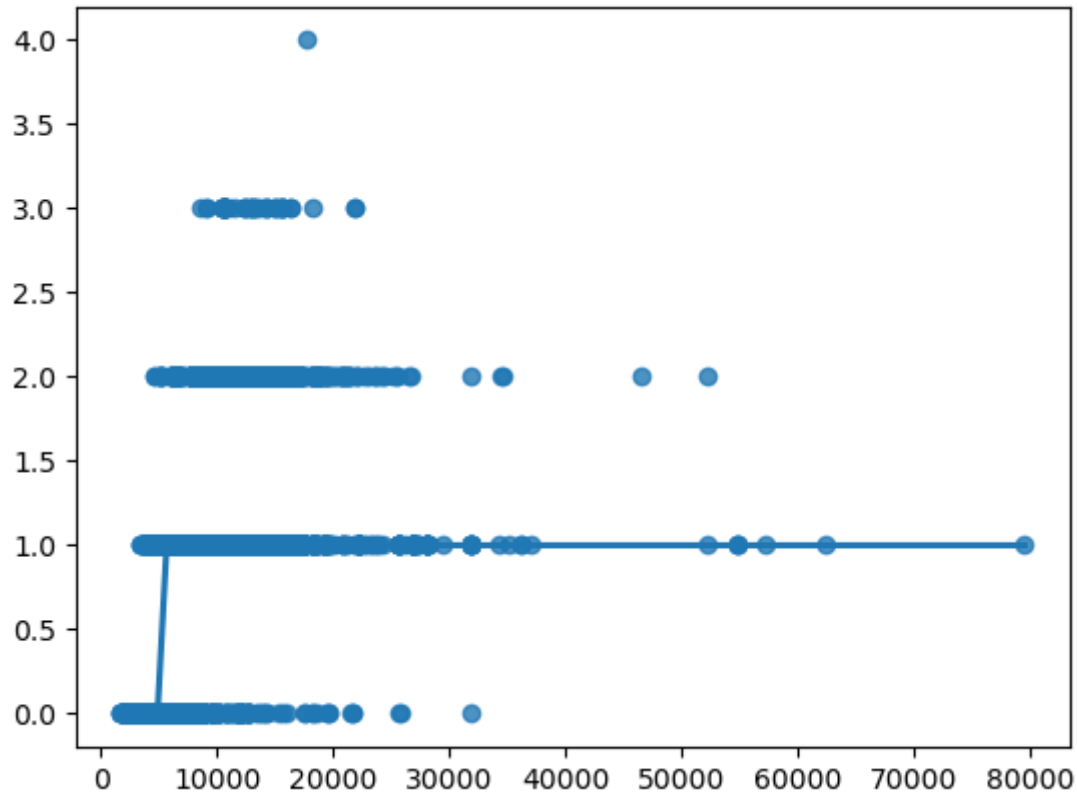
```
0.7160686427457098
```

In [43]:

```
sns.regplot(x=x,y=y,data=df,logistic=True,ci=None)
```

Out[43]:

<Axes: >




## Decision Tree

In [44]:

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(random_state=0)
dtc.fit(x_train,y_train)
```

Out[44]:

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=0)
```

In [45]:

```
score=dtc.score(x_test,y_test)
score
```

Out[45]:

0.9369734789391576

## Random Forest

In [55]:

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[55]:

RandomForestClassifier

RandomForestClassifier()

In [73]:

```
params={'max_depth':[2,3,5,10,20],
'min_samples_leaf':[5,10,20,50,100,200],
'n_estimators':[10,25,30,50,100,200]}
```

In [74]:

```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rfc,param_grid=params,cv=2,scoring="accuracy")
```

In [75]:

```
grid_search.fit(x_train,y_train)
```

Out[75]:

GridSearchCV

estimator: RandomForestClassifier

RandomForestClassifier

In [76]:

```
grid_search.best_score_
```

Out[76]:

```
0.8733442961132574
```

In [77]:

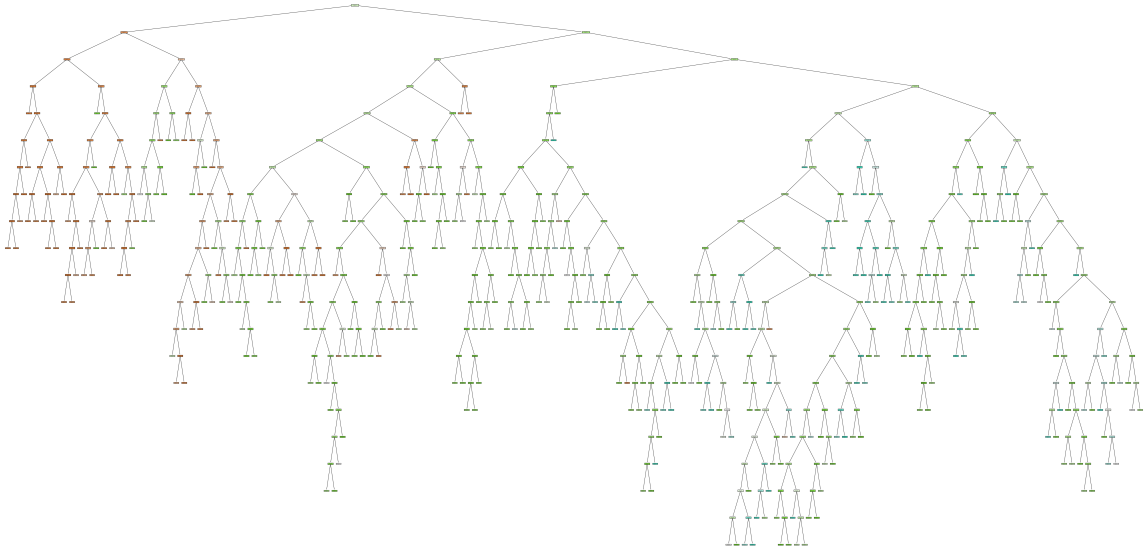
```
rf_best=grid_search.best_estimator_
rf_best
```

Out[77]:

```
RandomForestClassifier
RandomForestClassifier(max_depth=20, min_samples_leaf=5)
```

In [78]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[4],class_names=['0','1','2','3','4'],filled=True);
```



In [79]:

```
score=rfc.score(x_test,y_test)
print(score)
```

```
0.9369734789391576
```

## Conclusion:

Based on accuracy scores of all models that were implemented we can conclude that "Decision Tree" is the best model for the given dataset

In [ ]:

```

```