MINI PROJECT-2

1.Problem Statement:Which model is suitable best for flight price prediction Dataset

In [1]:

```
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.linear_model import LinearRegression
from sklearn import preprocessing,svm
```

Data Collection

Read the Data

In [2]:

```
train_df=pd.read_csv(r"C:\Users\samit\OneDrive\Documents\Copy of Data_Train.csv")
train_df
```

Out[2]:

	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	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\samit\OneDrive\Documents\Copy of Data_Train.csv")
test_df
```

Out[3]:

	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	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

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: 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 [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]:

	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 [8]:

```
train_df.columns
```

Out[8]:

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

In [9]:

```
test_df.columns
```

Out[9]:

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

In [10]:

```
train_df.head()
```

Out[10]:

	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 [11]:

```
test_df.head()
```

Out[11]:

	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 [12]:

```
train_df.shape
```

Out[12]:

```
(10683, 11)
```

In [13]:

```
test_df.shape
```

Out[13]:

```
(10683, 11)
```

To find duplicate values

In [14]:

```
train_df.duplicated().sum()
```

Out[14]:

```
220
```

In [15]:

```
test_df.duplicated().sum()
```

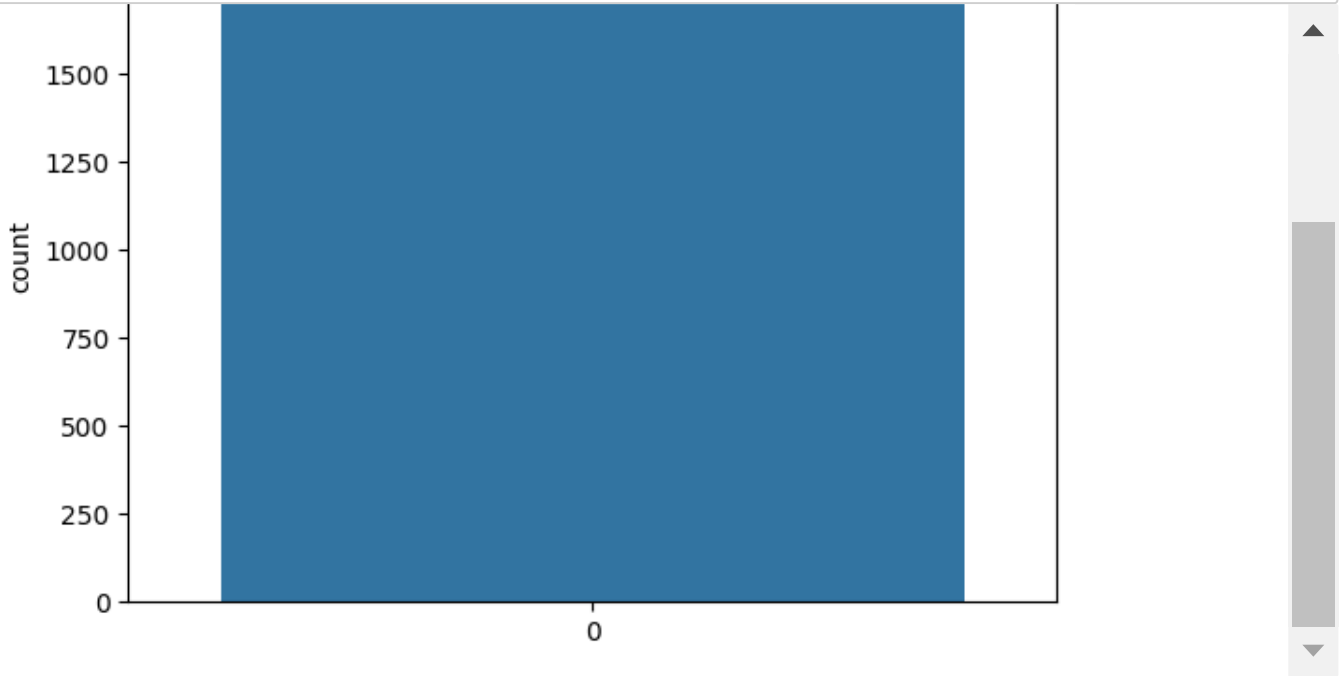
Out[15]:

```
220
```

Data Visualization:visualize the unique counts

In [16]:

```
sns.countplot(train_df['Price'].unique())
```



In [17]:

```
additionalinfo={"Additional_Info":{"No info":0,"In-flight meal not included":1,"No check
test_df=test_df.replace(additionalinfo)
test_df
```

Out[17]:

	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	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

Find null values

In [18]:

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

Type *Markdown* and LaTeX: α^2

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]:

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

Out[20]:

```
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
```

Type *Markdown* and LaTeX: α^2

In [21]:

```
airline={"Airline":{"Jet Airways":0,"IndiGo":1,"Air India":2,"Multiple carriers":3,
"SpiceJet":4,"Vistara":5,"Air Asia":6,"GoAir":7,
"Multiple carriers Premium economy":8,
"Jet Airways Business":9,"Vistara Premium economy":10,"Trujet":11}}
train_df=train_df.replace(airline)
train_df
```

Out[21]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Dura
0	1	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h
1	2	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h
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
4	1	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	Kolkata	Banglore	CCU ? BLR	19:55	22:25	2h
10679	2	27/04/2019	Kolkata	Banglore	CCU ? BLR	20:45	23:20	2h
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
10682	2	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h

10682 rows × 11 columns

In [22]:

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

Out[22]:

	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 5
1	2	1/05/2019	1	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
2	0	9/06/2019	0	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	1
3	1	12/05/2019	1	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 2
4	1	01/03/2019	2	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	Banglore	CCU ? BLR	19:55	22:25	2h 3
10679	2	27/04/2019	1	Banglore	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	2	9/05/2019	0	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [23]:

```
destination={"Destination":{"Cochin":0,"Banglore":1,"Delhi":2,
"New Delhi":3,"Hyderabad":4,"Kolkata":5}}
train_df=train_df.replace(destination)
train_df
```

Out[23]:

	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 5
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
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 2
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 3
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [24]:

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

Out[24]:

	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 5
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
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 2
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 3
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [25]:

```
train_df
```

Out[25]:

	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 5
1	2	1/05/2019	1	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 2
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 2
4	1	01/03/2019	2	3	BLR ? NAG ? DEL	16:50	21:35	4h 4
...	...	...	...	...	...	...	...	
10678	6	9/04/2019	1	1	CCU ? BLR	19:55	22:25	2h 3
10679	2	27/04/2019	1	1	CCU ? BLR	20:45	23:20	2h 3
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 4
10682	2	9/05/2019	0	0	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 2

10682 rows × 11 columns

In [26]:

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

Out[26]:

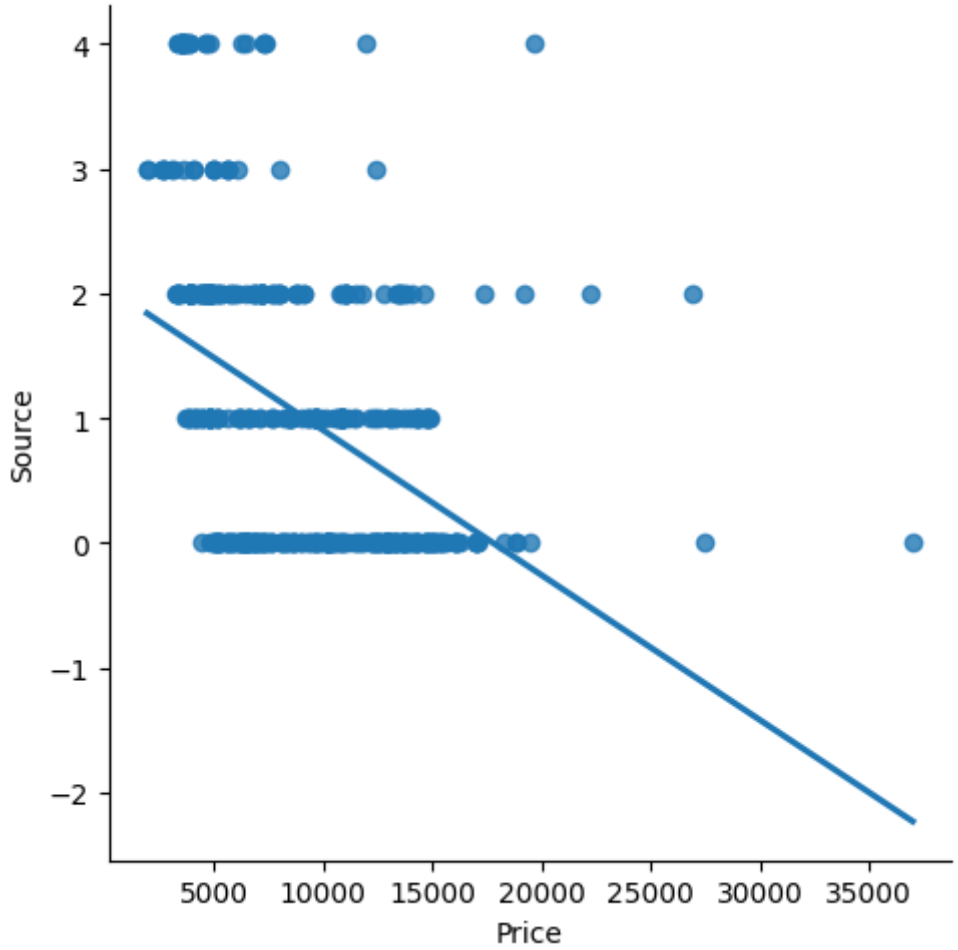
```
<Axes: >
```

In [27]:

```
train_df500=train_df[:][:500]
sns.lmplot(x="Price",y="Source",data=train_df500,order=1,ci=None)
```

Out[27]:

```
<seaborn.axisgrid.FacetGrid at 0x2d27f896e90>
```

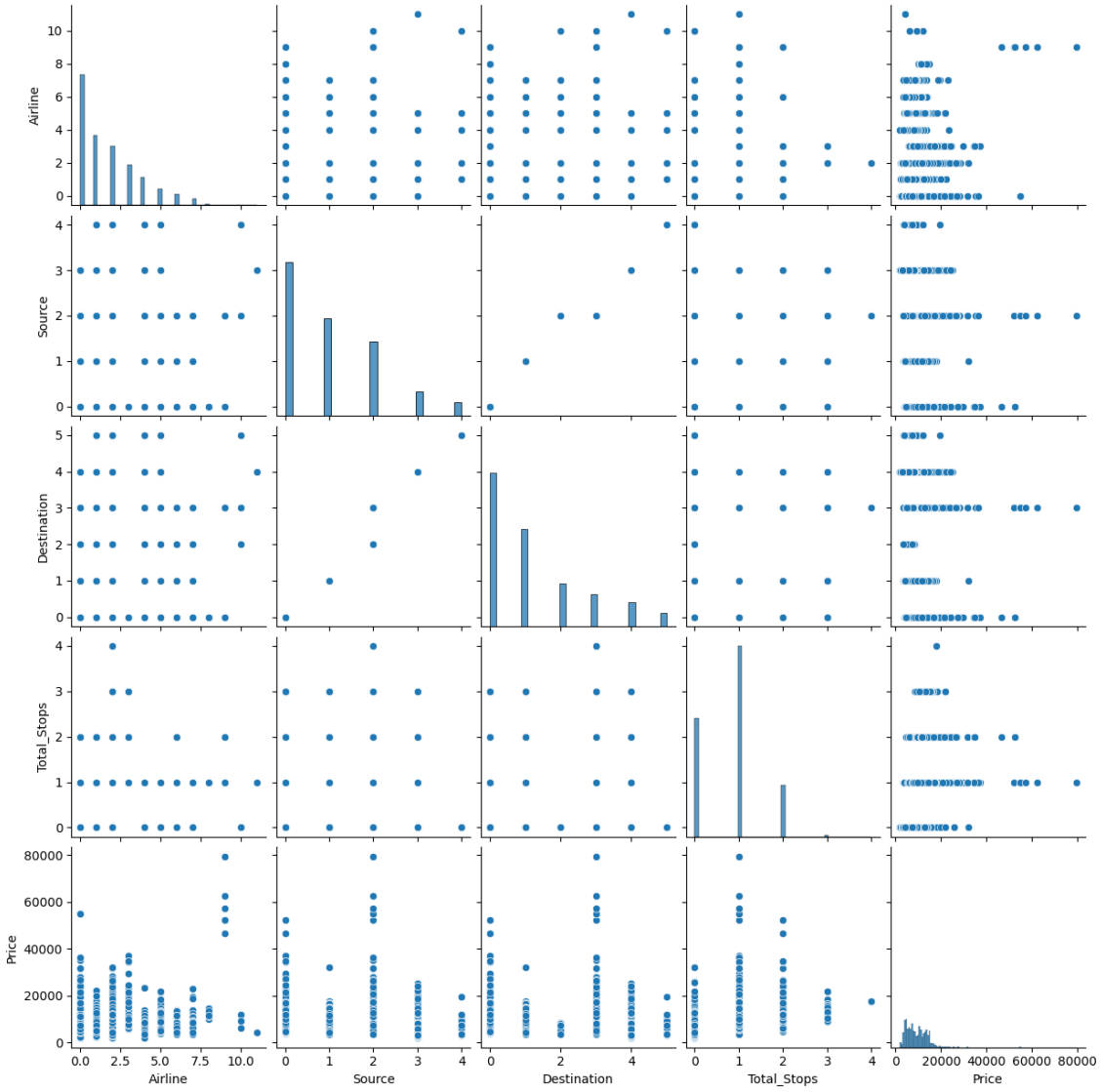


In [28]:

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

Out[28]:

```
<seaborn.axisgrid.PairGrid at 0x2d2134d2dd0>
```



In [29]:

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

Data Modeling:Using LinearRegression

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
regr=LinearRegression()
regr.fit(x_train,y_train)
```

Out[31]:

```
▾ LinearRegression

LinearRegression()
```

In [32]:

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

```
0.41083048909283415
```

In [33]:

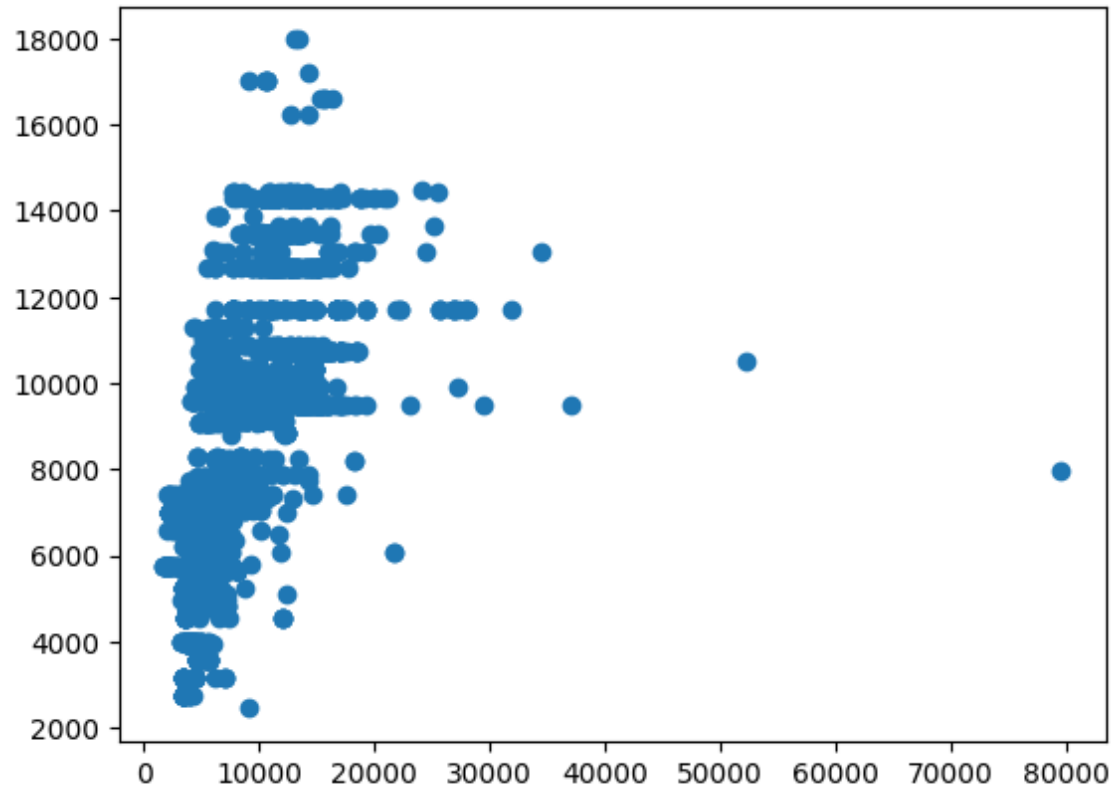
```
predictions=regr.predict(x_test)
```

In [34]:

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

Out[34]:

```
<matplotlib.collections.PathCollection at 0x2d21685fa10>
```



In [35]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_23828\1789970553.py:3: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  train_df.dropna(inplace=True)
```

In [36]:

```
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
regr.fit(X_train,y_train)
regr.fit(X_train,y_train)
```

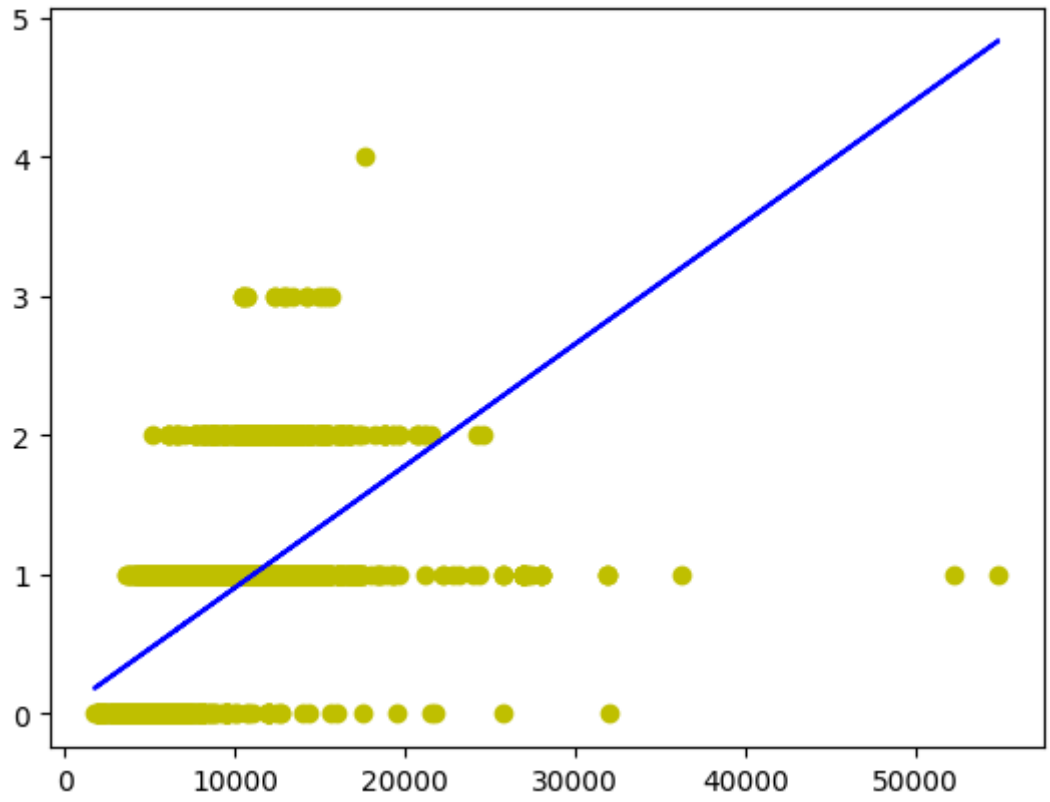
Out[36]:

LinearRegression

LinearRegression()

In [37]:

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



Logistic Regression

In [38]:

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

```
C:\Users\samit\AppData\Local\Temp\ipykernel_23828\3477783556.py:3: Settin
gWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  train_df.dropna(inplace=True)
```

In [39]:

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

```
C:\Users\samit\AppData\Local\Programs\Python\Python311\Lib\site-packages
\sklearn\utils\validation.py:1143: DataConversionWarning: A column-vector
y was passed when a 1d array was expected. Please change the shape of y t
o (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[39]:

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

In [40]:

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

```
0.7160686427457098
```

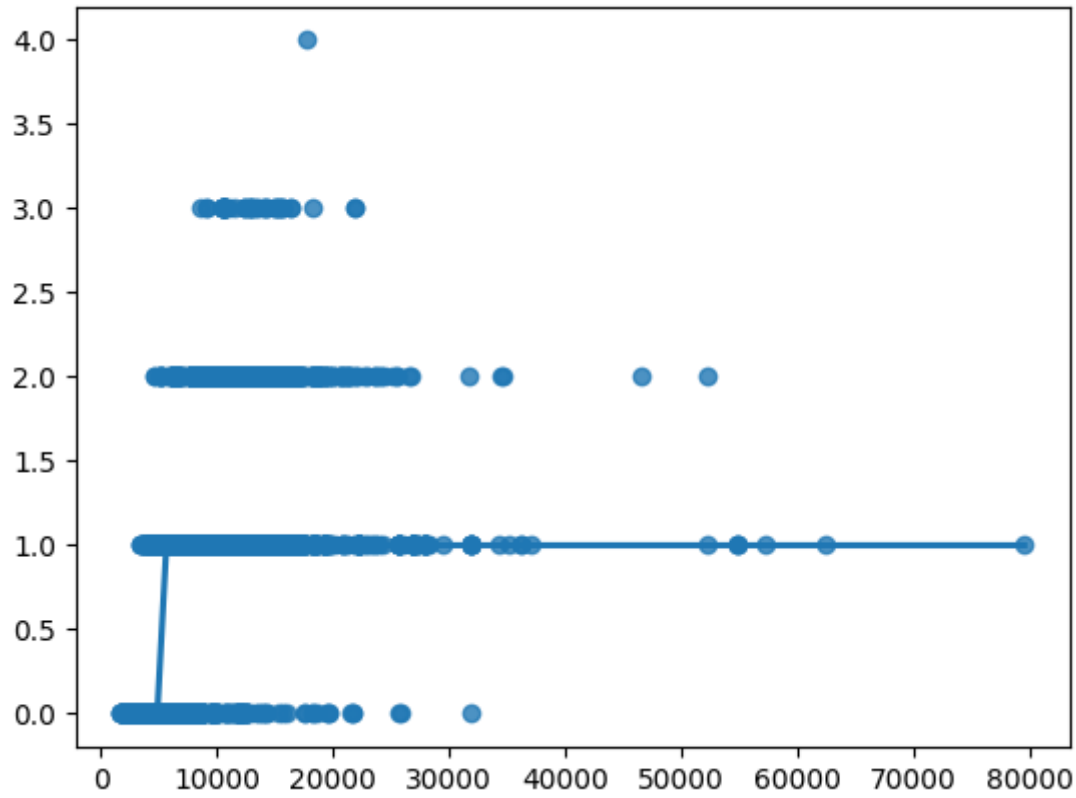
In [41]:

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

```
C:\Users\samit\AppData\Local\Programs\Python\Python311\Lib\site-packages
\statsmodels\genmod\families\links.py:198: RuntimeWarning: overflow encou
ntered in exp
  t = np.exp(-z)
```

Out[41]:

```
<Axes: >
```



Decision Tree

In [42]:

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

Out[42]:

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

In [43]:

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

```
0.9369734789391576
```

Random Forest

In [ ]:

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

In [ ]:

```
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 [ ]:

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

In [ ]:

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

In [ ]:

```
grid_search.best_score_
```

In [ ]:

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

In [ ]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(50,40))
plot_tree(rf_best.estimators_[5],class_names=['0','1','2'],filled=True);
```

Prediction and Evaluation

In [ ]:

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

In [ ]:

```
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2_score:",r2)
```

Conclusion

```
I dot the different types of accuracys,I conclude that Randomforest is the best fit
model for the flight price.
```

In [ ]:

```

```