Problem Statement:

To find the chances of number of stops of an flight based on the Airline,Source, Destination,and the Price

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:
```
df=pd.read_csv(r"C:\Users\my pc\desktop\Data_Train.csv")  #Reading the data
df
```

Out[2]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218

Data cleaning and Preprocessing

In [3]:
```
df.head()   #displaying first 5 rows
```

Out[3]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302

In [4]:
```
df.tail()   #displaying last 5 rows
```

Out[4]:

	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
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

In [5]:
```
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]:
```
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]:
```
df.isna().any()    #checking for null values
```

Out[7]:
```
Airline           False
Date_of_Journey   False
Source            False
Destination       False
Route              True
Dep_Time          False
Arrival_Time      False
Duration          False
Total_Stops        True
Additional_Info   False
Price             False
dtype: bool
```

In [8]:
```
df=df.drop("Route",axis=1)    #dropping the route column
df
```

Out[8]:

	Airline	Date_of_Journey	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	9/04/2019	Kolkata	Banglore	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	20:45	23:20	2h 35m	non-stop	No info	4145
10680	Jet Airways	27/04/2019	Banglore	Delhi	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 10 columns

In [9]:
```
#Dropping the columns which are not neccessary to find the target
df=df.drop(["Additional_Info","Duration","Date_of_Journey","Dep_Time","Arrival_Time"],axis=1)
```

In [10]:
```
df    #final dataframe
```

Out[10]:

	Airline	Source	Destination	Total_Stops	Price
0	IndiGo	Banglore	New Delhi	non-stop	3897
1	Air India	Kolkata	Banglore	2 stops	7662
2	Jet Airways	Delhi	Cochin	2 stops	13882
3	IndiGo	Kolkata	Banglore	1 stop	6218
4	IndiGo	Banglore	New Delhi	1 stop	13302
...	...	...	...	...	...
10678	Air Asia	Kolkata	Banglore	non-stop	4107
10679	Air India	Kolkata	Banglore	non-stop	4145
10680	Jet Airways	Banglore	Delhi	non-stop	7229
10681	Vistara	Banglore	New Delhi	non-stop	12648
10682	Air India	Delhi	Cochin	2 stops	11753

10683 rows × 5 columns

In [11]:
```
df.isna().any()   #checking for null values
```

Out[11]:
```
Airline        False
Source         False
Destination    False
Total_Stops     True
Price          False
dtype: bool
```

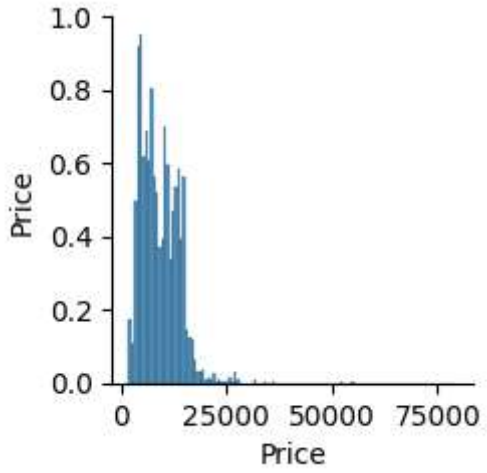
In [12]:
```
df.fillna(method="ffill",inplace=True)  #filling the null values using forward fill method
```

In [13]:
```
df.isna().any()    #again checking for null values
```

Out[13]:
```
Airline        False
Source         False
Destination    False
Total_Stops    False
Price          False
dtype: bool
```

In [14]:
```
sns.pairplot(df)
```

Out[14]:
```
<seaborn.axisgrid.PairGrid at 0x27d3b1dee90>
```



In [15]:
```
df.shape    #dispalying number of rows and columns
```

Out[15]:
```
(10683, 5)
```

Replacing the string oriented columns with integer values

In [16]:
```
df["Total_Stops"].value_counts()
```

Out[16]:
```
Total_Stops
1 stop      5625
non-stop    3492
2 stops     1520
3 stops       45
4 stops        1
Name: count, dtype: int64
```

In [17]:
```
t={"Total_Stops":{"1 stop":1,"non-stop":2,"2 stops":3,"3 stops":4,"4 stops":5}}
df=df.replace(t)
df
```

Out[17]:

	Airline	Source	Destination	Total_Stops	Price
0	IndiGo	Banglore	New Delhi	2	3897
1	Air India	Kolkata	Banglore	3	7662
2	Jet Airways	Delhi	Cochin	3	13882
3	IndiGo	Kolkata	Banglore	1	6218
4	IndiGo	Banglore	New Delhi	1	13302
...	...	...	...	...	...
10678	Air Asia	Kolkata	Banglore	2	4107
10679	Air India	Kolkata	Banglore	2	4145
10680	Jet Airways	Banglore	Delhi	2	7229
10681	Vistara	Banglore	New Delhi	2	12648
10682	Air India	Delhi	Cochin	3	11753

10683 rows × 5 columns

In [18]:
```
df["Airline"].value_counts()
```

Out[18]:
```
Airline
Jet Airways                          3849
IndiGo                               2053
Air India                            1752
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: count, dtype: int64
```

In [19]:
```
t={"Airline":{"Jet Airways":1,"IndiGo":2,"Air India":3,"Multiple carriers":4,"SpiceJet":5,
"Vistara":6,"Air Asia":7,"GoAir":8,"Multiple carriers Premium economy":9,
"Jet Airways Business":10,"Vistara Premium economy":11,"Trujet":12}}
df=df.replace(t)
```

In [20]:
```
df
```

Out[20]:

	Airline	Source	Destination	Total_Stops	Price
0	2	Banglore	New Delhi	2	3897
1	3	Kolkata	Banglore	3	7662
2	1	Delhi	Cochin	3	13882
3	2	Kolkata	Banglore	1	6218
4	2	Banglore	New Delhi	1	13302
...	...	...	...	...	...
10678	7	Kolkata	Banglore	2	4107
10679	3	Kolkata	Banglore	2	4145
10680	1	Banglore	Delhi	2	7229
10681	6	Banglore	New Delhi	2	12648
10682	3	Delhi	Cochin	3	11753

10683 rows × 5 columns

In [21]:
```
df["Source"].value_counts()
```

Out[21]:
```
Source
Delhi       4537
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: count, dtype: int64
```

In [22]:
```
t={"Source":{"Delhi":1,"Kolkata":2,"Banglore":3,"Mumbai":4,"Chennai":5}}
df=df.replace(t)
df
```

Out[22]:

	Airline	Source	Destination	Total_Stops	Price
0	2	3	New Delhi	2	3897
1	3	2	Banglore	3	7662
2	1	1	Cochin	3	13882
3	2	2	Banglore	1	6218
4	2	3	New Delhi	1	13302
...	...	...	...	...	...
10678	7	2	Banglore	2	4107
10679	3	2	Banglore	2	4145
10680	1	3	Delhi	2	7229
10681	6	3	New Delhi	2	12648
10682	3	1	Cochin	3	11753

10683 rows × 5 columns

In [23]:
```
df["Destination"].value_counts()
```

Out[23]:
```
Destination
Cochin       4537
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: count, dtype: int64
```

In [24]:
```
t={"Destination":{"Delhi":1,"Kolkata":2,"Banglore":3,"Cochin":4,"New Delhi":5,"Hyderabad":6}}
df=df.replace(t)
df
```

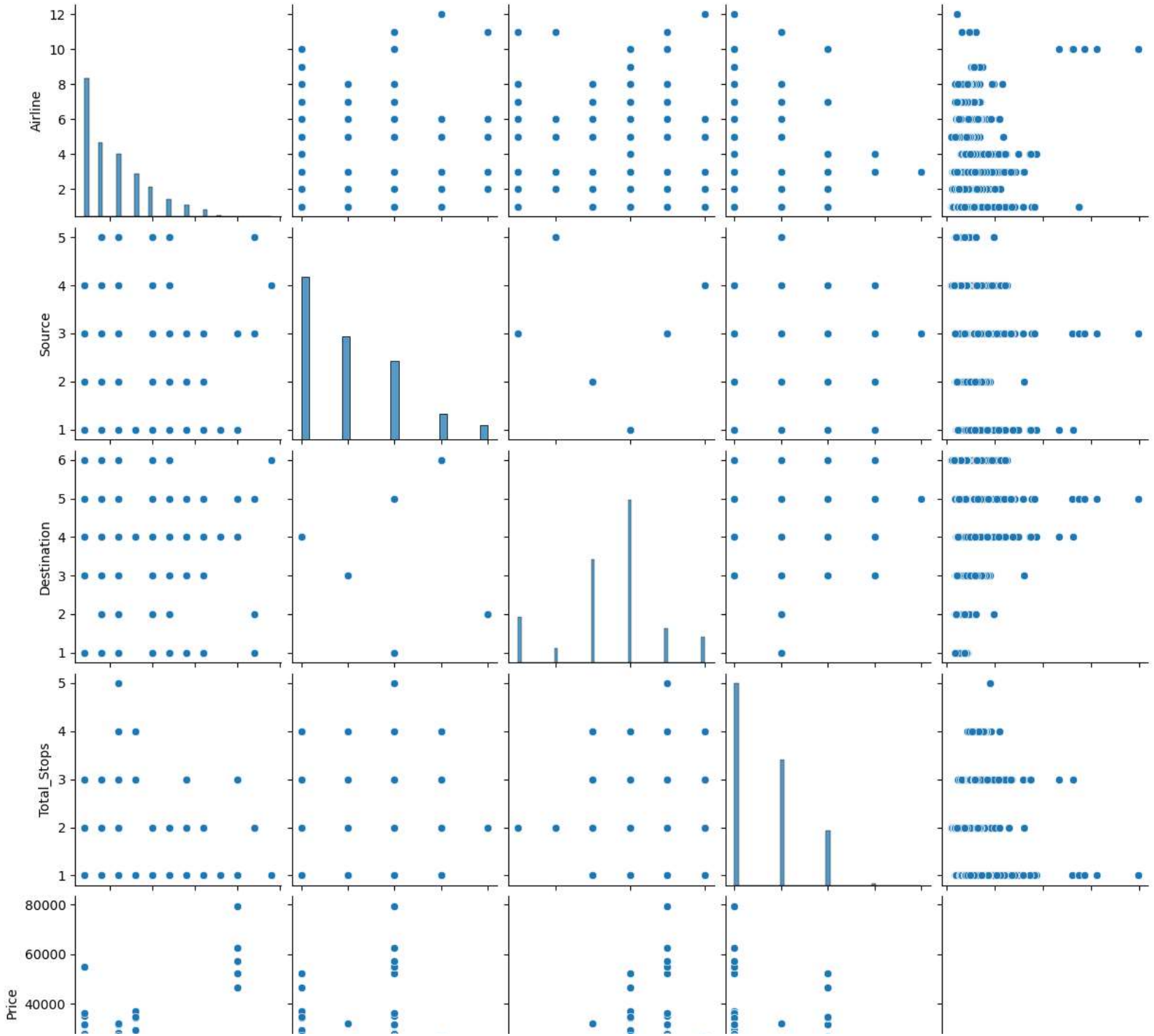
Out[24]:

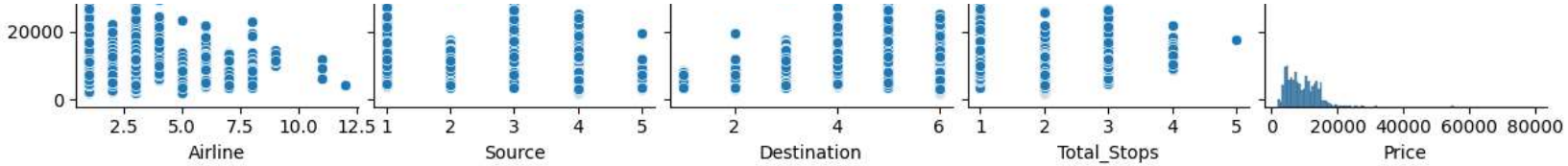
	Airline	Source	Destination	Total_Stops	Price
0	2	3	5	2	3897
1	3	2	3	3	7662
2	1	1	4	3	13882
3	2	2	3	1	6218
4	2	3	5	1	13302
...	...	...	...	...	...
10678	7	2	3	2	4107
10679	3	2	3	2	4145
10680	1	3	1	2	7229
10681	6	3	5	2	12648
10682	3	1	4	3	11753

10683 rows × 5 columns

In [25]:
```
sns.pairplot(df)   #plotting pairplot to check whether the relation between the columns
```

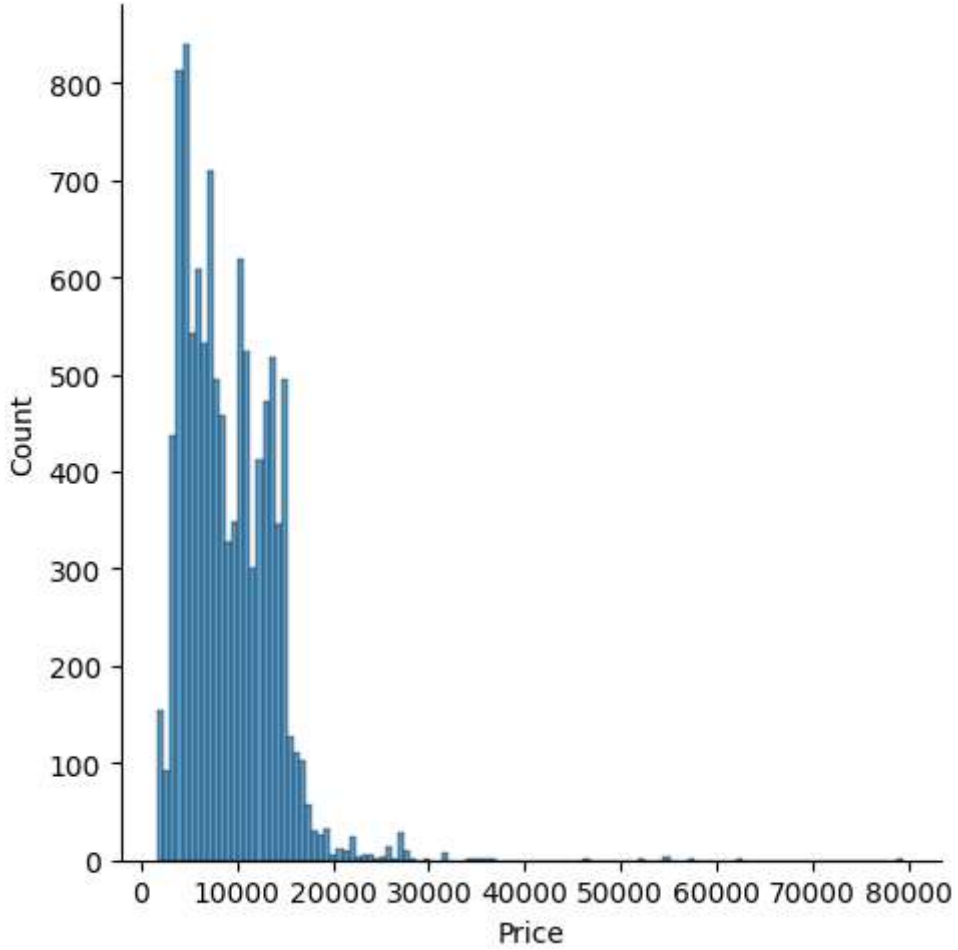
Out[25]:
```
<seaborn.axisgrid.PairGrid at 0x27d29af9990>
```


Airline
Source
Destination
Total_Stops
Price



In [26]:
```
sns.displot(df["Price"])
```

Out[26]:
```
<seaborn.axisgrid.FacetGrid at 0x27d3b2ada80>
```



LinearRegression

In [27]:
```
df
```

Out[27]:

	Airline	Source	Destination	Total_Stops	Price
0	2	3	5	2	3897
1	3	2	3	3	7662
2	1	1	4	3	13882
3	2	2	3	1	6218
4	2	3	5	1	13302
...	...	...	...	...	...
10678	7	2	3	2	4107
10679	3	2	3	2	4145
10680	1	3	1	2	7229
10681	6	3	5	2	12648
10682	3	1	4	3	11753

10683 rows × 5 columns

In [28]:
```
features=df.columns[0:4]   #input columns
features
```

Out[28]:
```
Index(['Airline', 'Source', 'Destination', 'Total_Stops'], dtype='object')
```

In [29]:
```
target=df.columns[-1]   #output column
target
```

Out[29]:
```
'Price'
```

In [30]:
```
#Training our model
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
x=df[features].values
y=df[target].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.50,random_state=17)
x_train.shape
x_test.shape
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In [31]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()      #calling linearregression method
```

In [32]:
```
reg.fit(x_train,y_train)
```

Out[32]:
```
LinearRegression
LinearRegression()
```

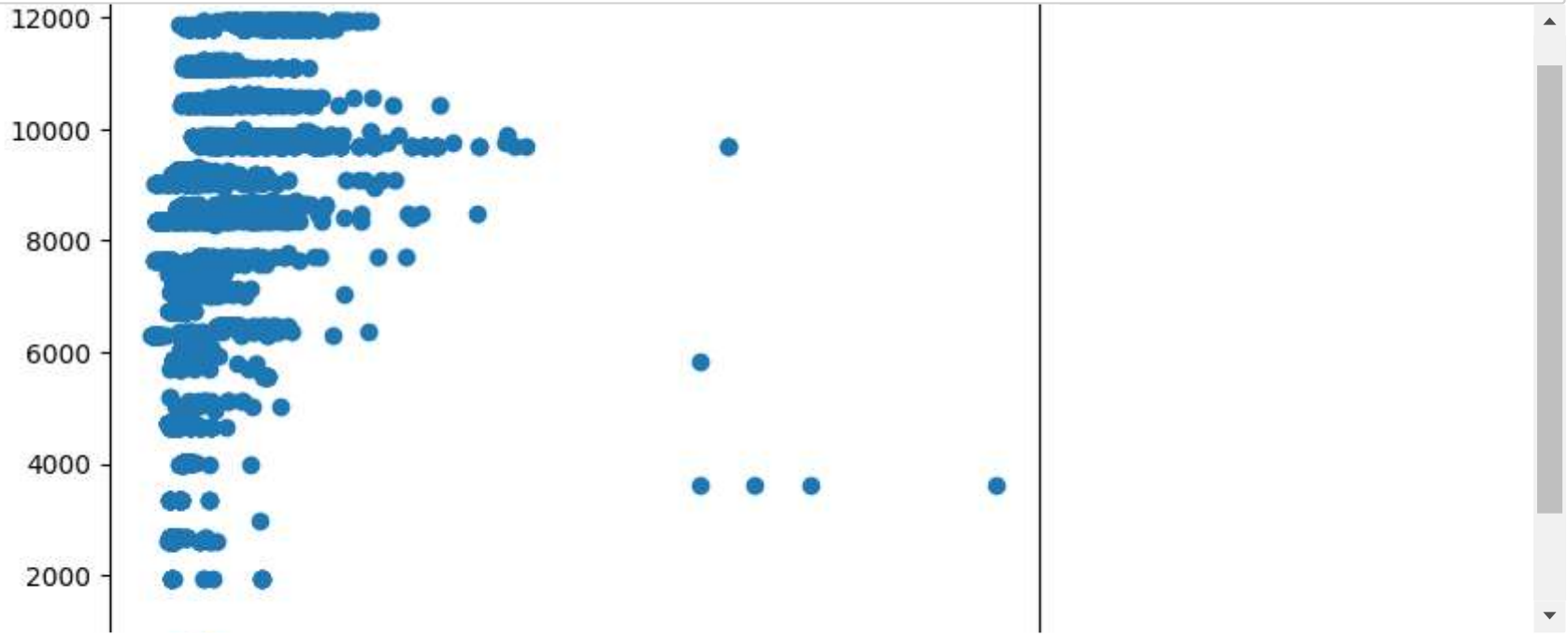
In [33]:
```
#accuracy score
print(reg.score(x_test,y_test))
```

```
0.20017298220407231
```

In [34]:
```
#accuracy score
print(reg.score(x_train,y_train))
```

```
0.2481156353749281
```

In [35]:
```
y_pred=reg.predict(x_test)       #predicting target variable
plt.scatter(y_test,y_pred)
plt.show()
```




```
""" Conclusion:
                Here we have price as output and the remaining as input.The accuracy is very low
                Therefore,it is not a suitable model for this dataset"""
```

In [36]:
```
df
```

Out[36]:

	Airline	Source	Destination	Total_Stops	Price
0	2	3	5	2	3897
1	3	2	3	3	7662
2	1	1	4	3	13882
3	2	2	3	1	6218
4	2	3	5	1	13302
...	...	...	...	...	...
10678	7	2	3	2	4107
10679	3	2	3	2	4145
10680	1	3	1	2	7229
10681	6	3	5	2	12648
10682	3	1	4	3	11753

10683 rows × 5 columns

```
'''Since the accuracy is very low we have changed the feature and target columns .Here number of stops
as target and the remaining columns as input"""
```

In [37]:
```
x=df[["Airline","Source","Destination","Price"]]
x.columns=["Airline","Source","Destination","Price"]
x
```

Out[37]:

	Airline	Source	Destination	Price
0	2	3	5	3897
1	3	2	3	7662
2	1	1	4	13882
3	2	2	3	6218
4	2	3	5	13302
...	...	...	...	...
10678	7	2	3	4107
10679	3	2	3	4145
10680	1	3	1	7229
10681	6	3	5	12648
10682	3	1	4	11753

10683 rows × 4 columns

In [38]:
```
y=df.columns[-2]
y
```

Out[38]:
```
'Total_Stops'
```

In [39]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()      #calling linearregression method
```

In [40]:
```
reg.fit(x_train,y_train)
```

Out[40]:
```
LinearRegression
LinearRegression()
```

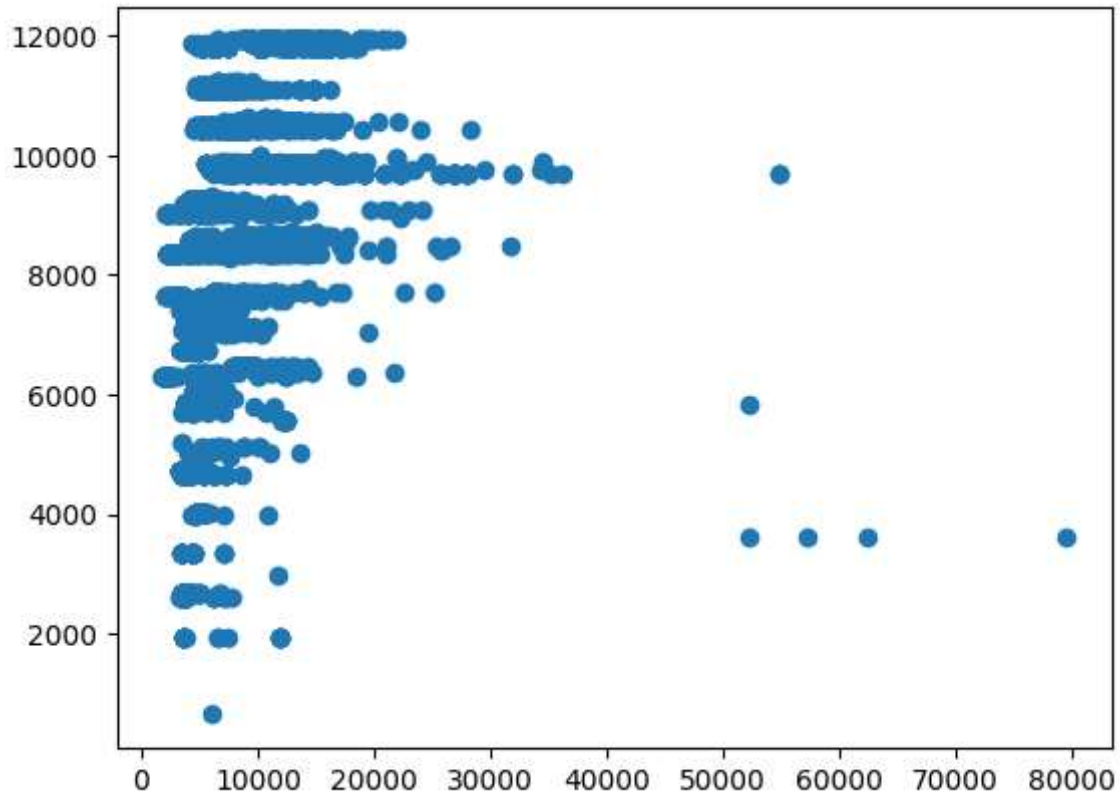
In [41]:
```
#accuracy score
print(reg.score(x_test,y_test))
```

```
0.20017298220407231
```

In [42]:
```
#accuracy score
print(reg.score(x_train,y_train))
```

```
0.2481156353749281
```

In [43]:
```
y_pred=reg.predict(x_test)       #predicting target variable
plt.scatter(y_test,y_pred)
plt.show()
```




```
""" Conclusion:
          The accuracy remains same i.e.,very low .So,LinearRegression is not suitable for
```

```
            this dataset"""
```

Logistic Regression

In [44]:
```
from sklearn.linear_model import LogisticRegression
```

In [45]:
```
lor=LogisticRegression()
```

In [46]:
```
lor.fit(x_train,y_train)
```

```
C:\Users\my pc\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[46]:
```
LogisticRegression()
```

In [47]:
```
print(lor.score(x_test,y_test))   #accuracy score for test data
```

```
0.15649569449644327
```

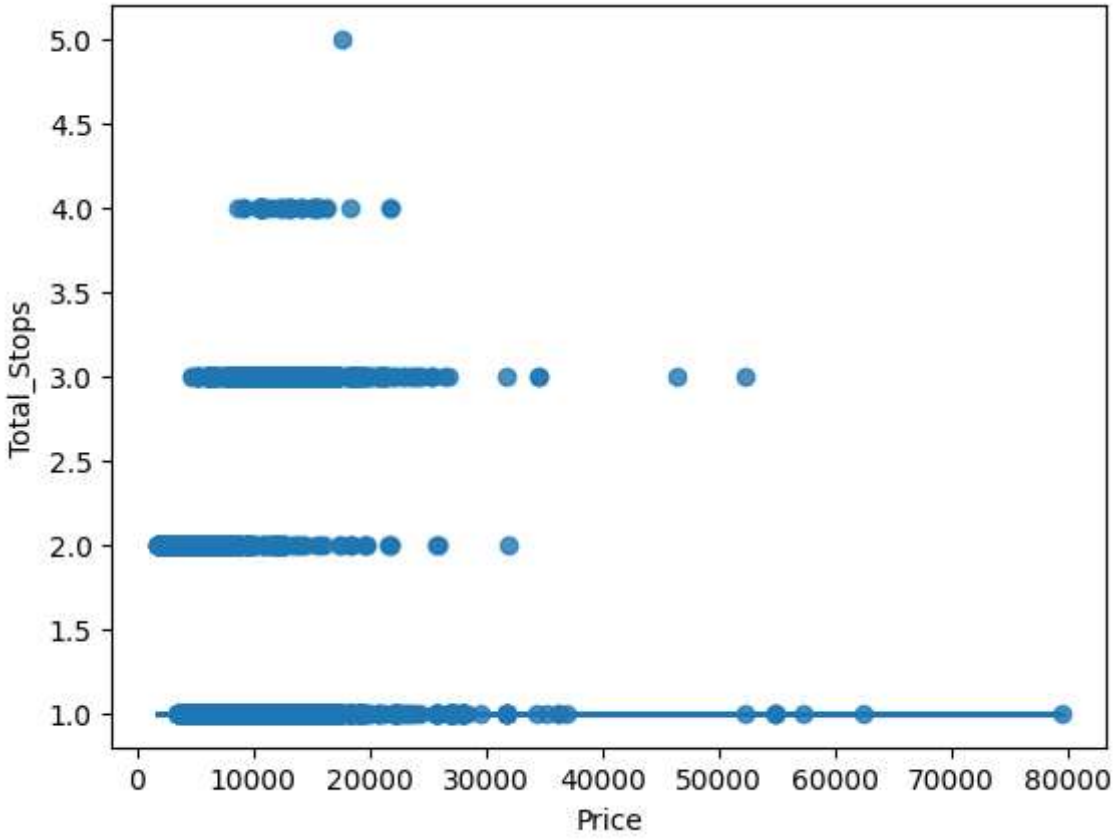
In [48]:
```
print(lor.score(x_train,y_train))   #accuracy score for train data
```

```
0.17917992885227485
```

In [49]:
```
sns.regplot(x=df["Price"],y=y,data=df,logistic=True,ci=None)  #plotting graph
```

```
C:\Users\my pc\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmodels\genmod\families\links.p
y:198: RuntimeWarning: overflow encountered in exp
  t = np.exp(-z)
```

Out[49]: <Axes: xlabel='Price', ylabel='Total_Stops'>




```
""" Conclusion:
              The accuracy score is too low compared to LinearRegression. So it is unfit."""
```

Decision Tree

In [50]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [51]:
```
x=["Airline","Source","Destination","Price"]
y=[1,2,3,4,5]
all_inputs=df[x]
all_classes=df["Total_Stops"]
```

In [52]:
```
x_train,x_test,y_train,y_test=train_test_split(all_inputs,all_classes,test_size=0.25)
```

In [53]:
```
clf=DecisionTreeClassifier(random_state=0)
```

In [54]:
```
clf.fit(x_train,y_train)                #Fitting training into th model (DecisionTreeClassifier)
```

Out[54]:
```
DecisionTreeClassifier(random_state=0)
```

In [55]:
```
clf.score(x_test,y_test)#To find the score for test data
```

Out[55]:
```
0.9550730063646574
```

In [56]:
```
clf.score(x_train,y_train)#To find the score for test data
```

Out[56]:
```
0.9955067398901648
```

```
""" Conclusion:
            The accuracy score is very good i.e 99% using DecisionTree algorithm.So, it is a
            best fitted model for this dataframe"""
```

Random Forest

In [58]:
```
df
```

Out[58]:

	Airline	Source	Destination	Total_Stops	Price
0	2	3	5	2	3897
1	3	2	3	3	7662
2	1	1	4	3	13882
3	2	2	3	1	6218
4	2	3	5	1	13302
...	...	...	...	...	...
10678	7	2	3	2	4107
10679	3	2	3	2	4145
10680	1	3	1	2	7229
10681	6	3	5	2	12648
10682	3	1	4	3	11753

10683 rows × 5 columns

In [60]:
```
x=df.drop("Total_Stops",axis=1)
y=df["Total_Stops"]
```

In [61]:
```
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=0)
x_train.shape,x_test.shape
```

Out[61]:
```
((8012, 4), (2671, 4))
```

In [62]:
```
x
```

Out[62]:

	Airline	Source	Destination	Price
0	2	3	5	3897
1	3	2	3	7662
2	1	1	4	13882
3	2	2	3	6218
4	2	3	5	13302
...	...	...	...	...
10678	7	2	3	4107
10679	3	2	3	4145
10680	1	3	1	7229
10681	6	3	5	12648
10682	3	1	4	11753

10683 rows × 4 columns

In [63]:
```
y
```

Out[63]:
```
0        2
1        3
2        3
3        1
4        1
        ..
10678    2
10679    2
10680    2
10681    2
10682    3
Name: Total_Stops, Length: 10683, dtype: int64
```

In [64]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[64]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [65]:
```
rf=RandomForestClassifier()
```

In [66]:
```
params={'max_depth':[2,3,5,10,20],
       'min_samples_leaf':[5,10,20,50,100,500],
       'n_estimators':[10,25,30,50,100,1000]}
```

In [67]:
```
from sklearn.model_selection import GridSearchCV
grid_search=GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[67]:
```
GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier
```

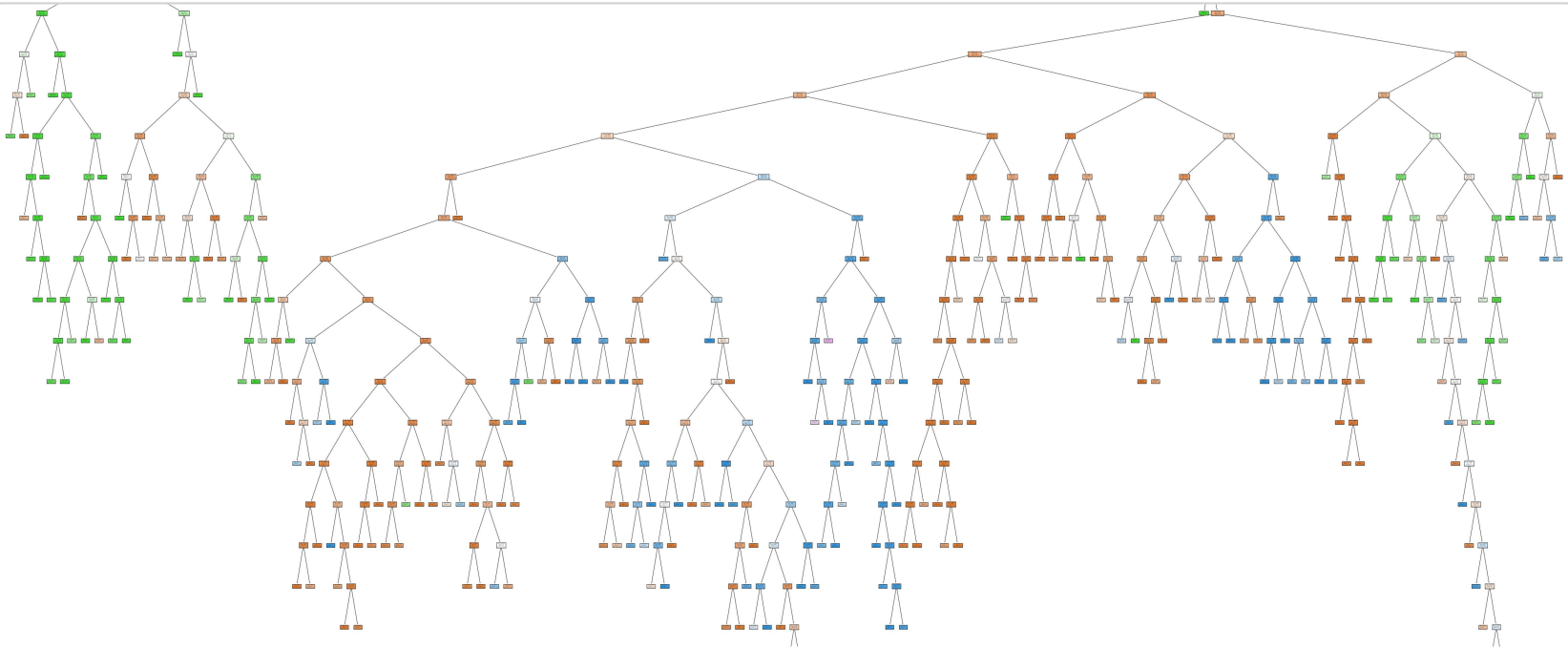
In [68]:
```
grid_search.best_score_    #finding accuracy
```

Out[68]:
```
0.9254867698452323
```

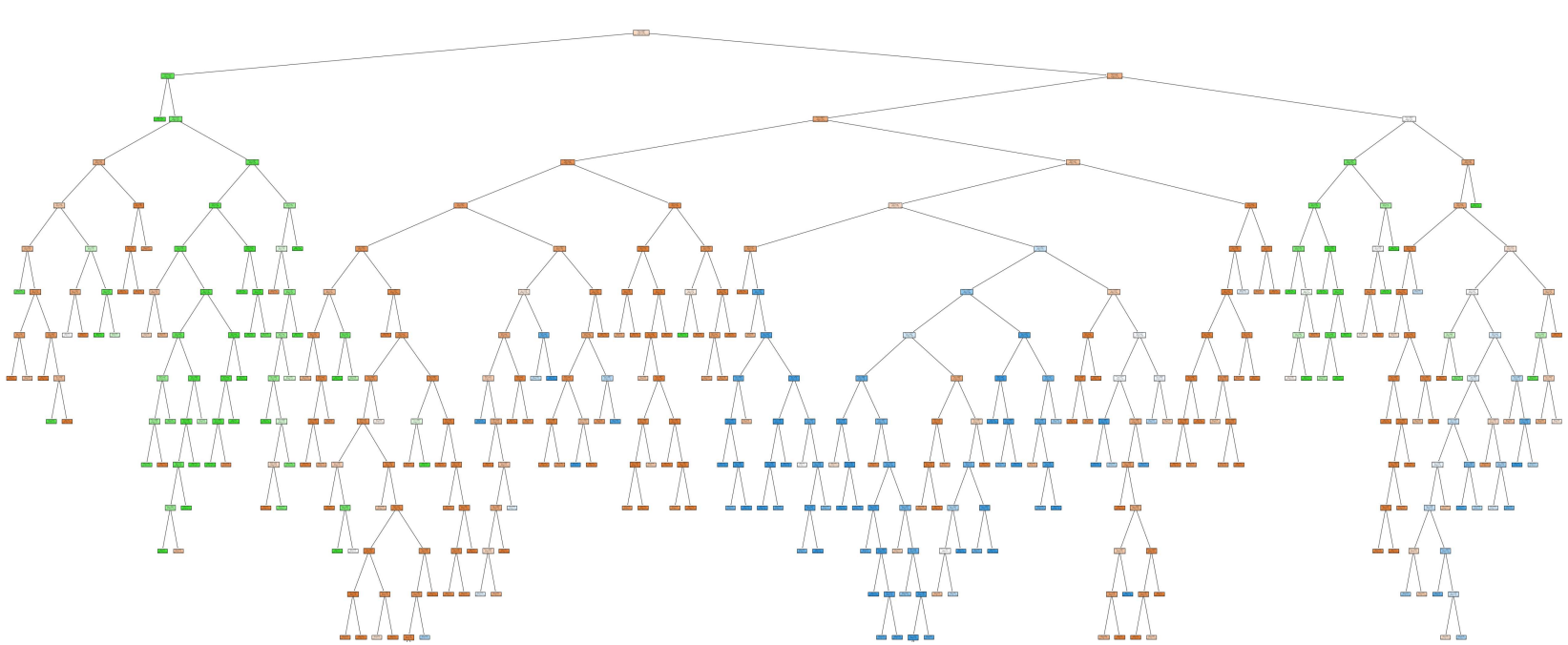
In [69]:
```
rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=50)
```

In [72]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5],feature_names=x.columns,filled=True)
```



In [73]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[7],feature_names=x.columns,filled=True)
```



In [74]:
```
rf_best.feature_importances_
```

Out[74]:
```
array([0.14478511, 0.13959696, 0.14377755, 0.57184038])
```

In [75]:
```
imp_df=pd.DataFrame({"varname":x_train.columns,"Imp":rf_best.feature_importances_})
imp_df.sort_values(by="Imp",ascending=False)
```

Out[75]:

	varname	Imp
3	Price	0.571840
0	Airline	0.144785
2	Destination	0.143778
1	Source	0.139597

In [76]:
```
print(rfc.score(x_train,y_train))
```

```
0.9961308037943085
```

In [77]:
```
print(rfc.score(x_test,y_test))
```

```
0.948333957319356
```

```
"""Conclusion:
            The accuracy score is very high .Therefore it is a best fitted model
```

Conclusion:

Here for all the models the feature columns are "Airline","source","Destination", "price" and the target column the number of stops it can have based on above feature columns. After checking and comparing the scores for all other models ""Decision Tree" and "Random Forest" is best suited for this dataframe with the accuracy of 99.5% and 99.6% respectively