

# MINI PROJECT

**PROBLEM STATEMENT:**which model is suitable for flight price prediction

**importing packages**

```
In [1]: import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt
```

**Read the Data**

```
In [2]: df=pd.read_csv(r"C:\Users\USER\Downloads\FlightPricePrediction_Practise\Data_Train.csv")
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
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
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

10683 rows × 11 columns

## Data collection and preprocessing

In [3]: df.head()

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()

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.shape
```

```
Out[7]: (10683, 11)
```

```
In [8]: df.isnull().sum()
```

```
Out[8]: 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 [9]: df.isna().any()
```

```
Out[9]: 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 [10]: df['Source'].value_counts()
```

```
Out[10]: Source
Delhi      4537
Kolkata   2871
Banglore   2197
Mumbai     697
Chennai    381
Name: count, dtype: int64
```

```
In [11]: df['Destination'].value_counts()
```

```
Out[11]: Destination
Cochin      4537
Banglore    2871
Delhi       1265
New Delhi   932
Hyderabad   697
Kolkata    381
Name: count, dtype: int64
```

```
In [12]: h={"Destination":{"Banglore":1,"Kolkata":2,"Delhi":3,"Cochin":4,"New Delhi":5,"Hyderabad":6}}
df=df.replace(h)
df
```

Out[12]:

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

10683 rows × 11 columns

```
In [13]: df['Airline'].value_counts()
```

```
Out[13]: 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 [67]: h={'Airline':{'Jet Airways':1,'IndiGo':2,'Air India':3,'Multiple carriers':4,'SpiceJet':5,'Vistara':6,'Air Asia':7,'Multiple carriers Premium economy':9,'Jet Airways Business':10,'Vistara Premium economy':11,'Tru...':12}}  
df=df.replace(h)  
df
```

Out[67]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	2	9	1	5	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	3	14	2	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	1	4	3	4	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	2	15	2	1	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	2	21	1	5	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	7	24	2	1	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	3	30	2	1	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	1	30	1	3	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	6	21	1	5	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	3	6	3	4	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

```
In [15]: h={'Source':{"Banglore":1,"Kolkata":2,"Delhi":3,"Mumbai":4,"Chennai":5}}
df=df.replace(h)
df
```

Out[15]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	2	24/03/2019	1	5	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	3	1/05/2019	2	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	1	9/06/2019	3	4	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	2	12/05/2019	2	1	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	2	01/03/2019	1	5	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	7	9/04/2019	2	1	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	3	27/04/2019	2	1	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	1	27/04/2019	1	3	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	6	01/03/2019	1	5	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	3	9/05/2019	3	4	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

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

Out[16]: Date\_of\_Journey

18/05/2019	504
6/06/2019	503
21/05/2019	497
9/06/2019	495
12/06/2019	493
9/05/2019	484
21/03/2019	423
15/05/2019	405
27/05/2019	382
27/06/2019	355
24/06/2019	351
1/06/2019	342
3/06/2019	333
15/06/2019	328
24/03/2019	323
6/03/2019	308
27/03/2019	299
24/05/2019	286
6/05/2019	282
1/05/2019	277
12/05/2019	259
1/04/2019	257
3/03/2019	218
9/03/2019	200
15/03/2019	162
18/03/2019	156
01/03/2019	152
12/03/2019	142
9/04/2019	125
3/04/2019	110
21/06/2019	109
18/06/2019	105
09/03/2019	102
6/04/2019	100
03/03/2019	97
06/03/2019	95
27/04/2019	94
24/04/2019	92
3/05/2019	90
15/04/2019	89
21/04/2019	82
18/04/2019	67

```
12/04/2019      63  
1/03/2019      47  
Name: count, dtype: int64
```

```
In [63]: h={'Date_of_Journey':{'18/05/2019':1,'6/06/2019':2,'21/05/2019':3,'9/06/2019':4,'12/06/2019':5,'9/05/2019':6,'15/05/2019':8,'24/03/2019':9,'6/03/2019':10,'27/03/2019':11,'24/05/2019':12,'6/05/2019':13,'12/05/2019':15,'1/04/2019':16,'3/03/2019':17,'9/03/2019':18,'15/03/2019':19,'18/03/2019':20,'12/03/2019':22,'9/04/2019':24,'3/04/2019':25,'21/06/2019':26,'18/06/2019':27,'09/03/2019':28,'27/04/2019':30,'24/04/2019':31,'3/05/2019':32,'15/04/2019':33,'21/04/2019':34,'18/04/2019':35,'1/03/2019':36,'3/06/2019':37,'27/05/2019':38,'27/06/2019':39,'24/06/2019':40,'1/06/2019':41,'3/06/2019':42,'15/06/2019':43,'03/03/2019':44,'06/03/2019':45}}  
df=df.replace(h)  
df
```

Out[63]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	2	9	1	5	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	3	14	2	1	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	1	4	3	4	DEL ? LKO ? BOM ? COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	2	15	2	1	CCU ? NAG ? BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	2	21	1	5	BLR ? NAG ? DEL	16:50	21:35	4h 45m	1 stop	No info	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	7	24	2	1	CCU ? BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	3	30	2	1	CCU ? BLR	20:45	23:20	2h 35m	non-stop	No info	4145
10680	1	30	1	3	BLR ? DEL	08:20	11:20	3h	non-stop	No info	7229
10681	6	21	1	5	BLR ? DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	3	6	3	4	DEL ? GOI ? BOM ? COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

```
In [40]: df["Total_Stops"].value_counts()
```

```
Out[40]: Total_Stops
1 stop      5625
non-stop    3492
2 stops     1520
3 stops      45
4 stops       1
Name: count, dtype: int64
```

```
In [41]: x=['Source','Destination']
y=['1','2','3','4','non-stop']
all_inputs=df[x]
all_classes=df["Total_Stops"]
```

## Linear Regression

```
In [42]: df.fillna(method='ffill',inplace=True)
```

```
In [43]: x=np.array(df['Source']).reshape(-1,1)
y=np.array(df['Destination']).reshape(-1,1)
```

```
In [44]: df.dropna(inplace=True)
```

```
In [45]: 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)
```

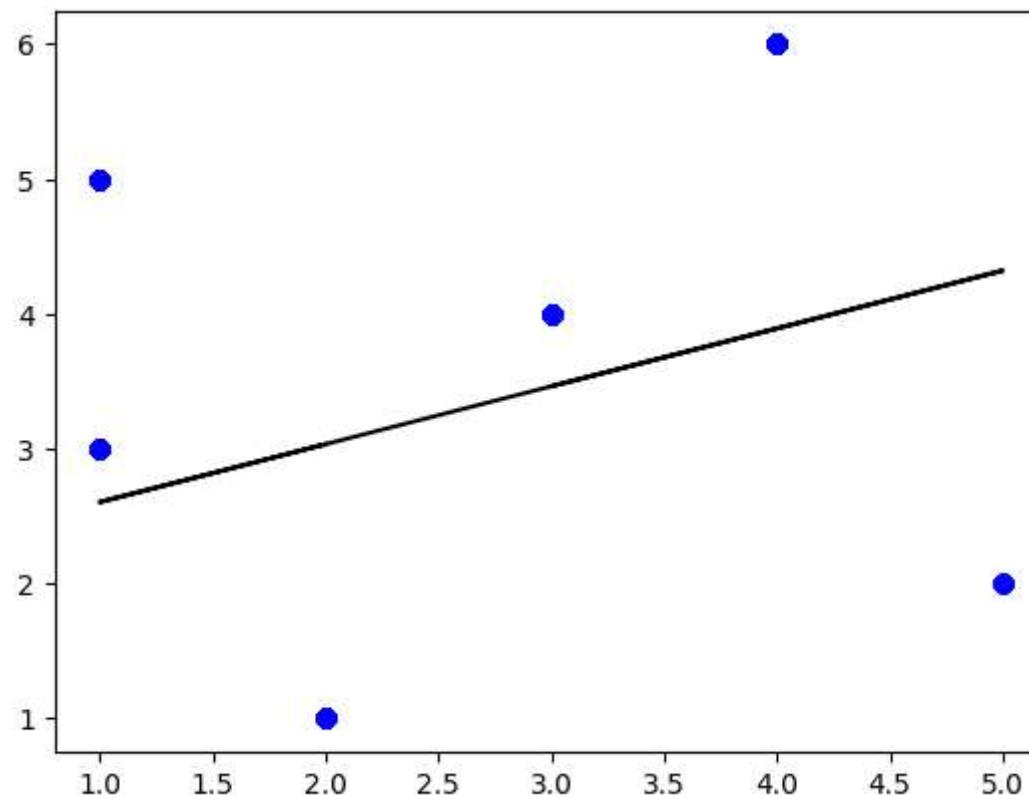
```
In [81]: from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

0.06720311887346575

```
In [82]: 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[82]: 
  ▾ LinearRegression
    LinearRegression()
```

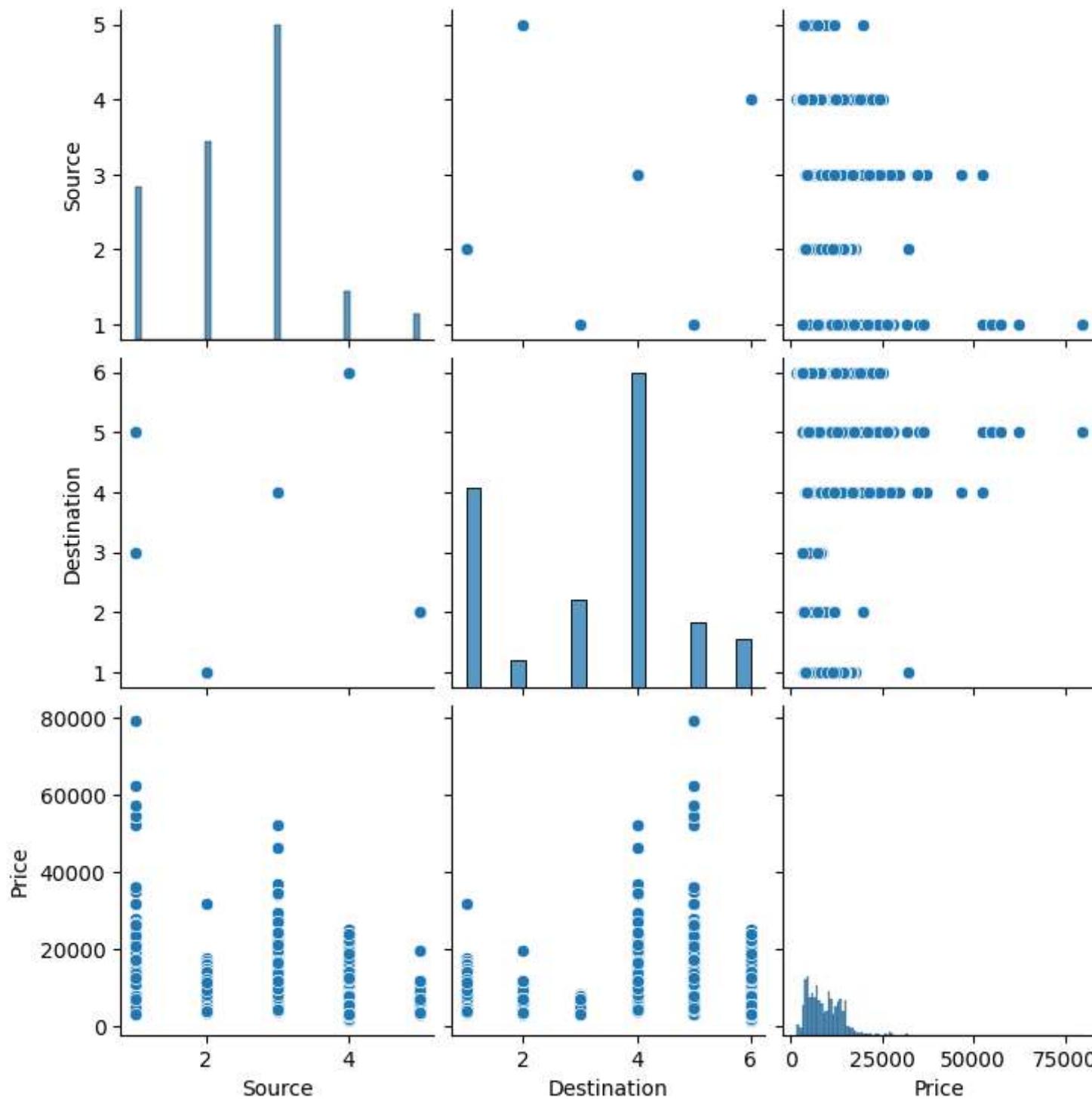
```
In [47]: y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



```
In [48]: sns.pairplot(df)
```

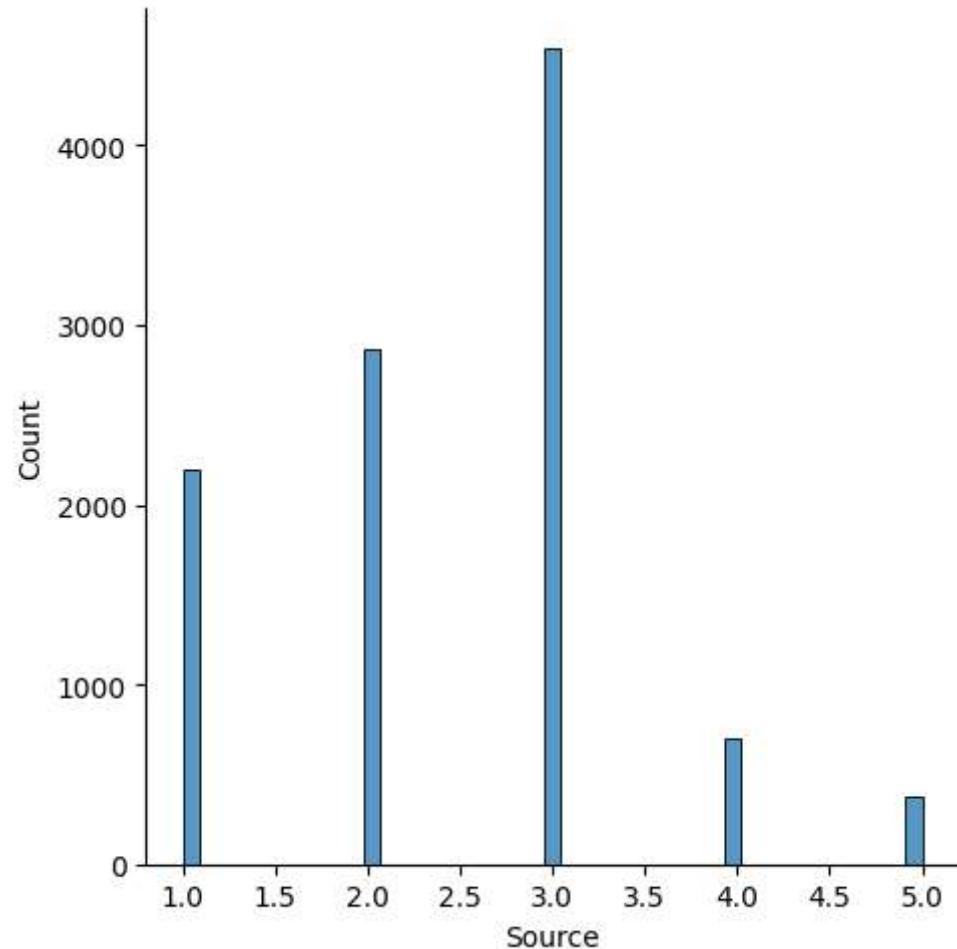
```
Out[48]: <seaborn.axisgrid.PairGrid at 0x1d6c2cc1090>
```





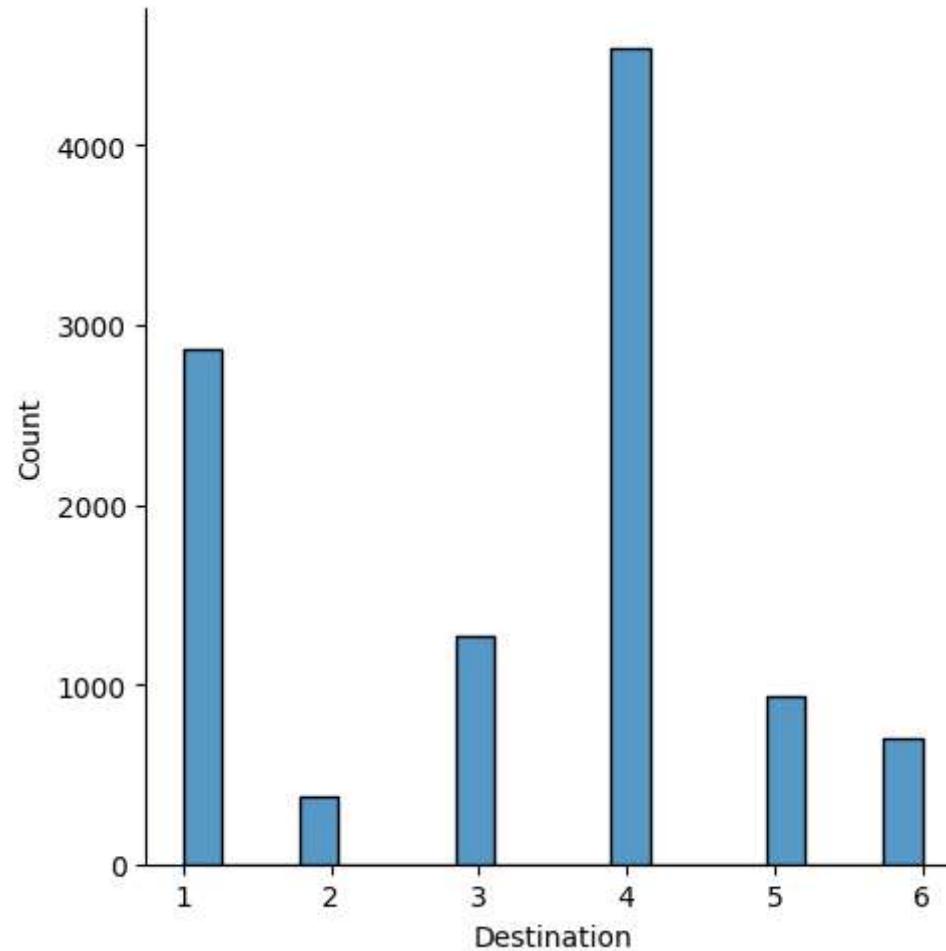
```
In [49]: sns.displot(df['Source'])
```

```
Out[49]: <seaborn.axisgrid.FacetGrid at 0x1d6c555c210>
```



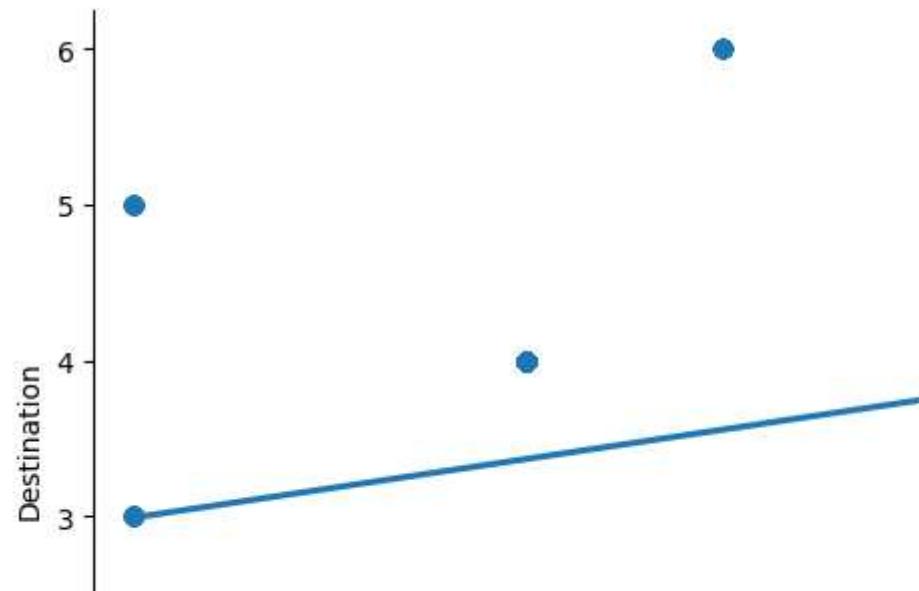
```
In [50]: sns.displot(df['Destination'])
```

```
Out[50]: <seaborn.axisgrid.FacetGrid at 0x1d6c55951d0>
```



```
In [51]: df500=df[:][:500]
sns.lmplot(x="Source",y="Destination",data=df500,order=1,ci=None)
```

```
Out[51]: <seaborn.axisgrid.FacetGrid at 0x1d6b1a3cd0>
```



```
In [52]: df.fillna(method='ffill',inplace=True)
```

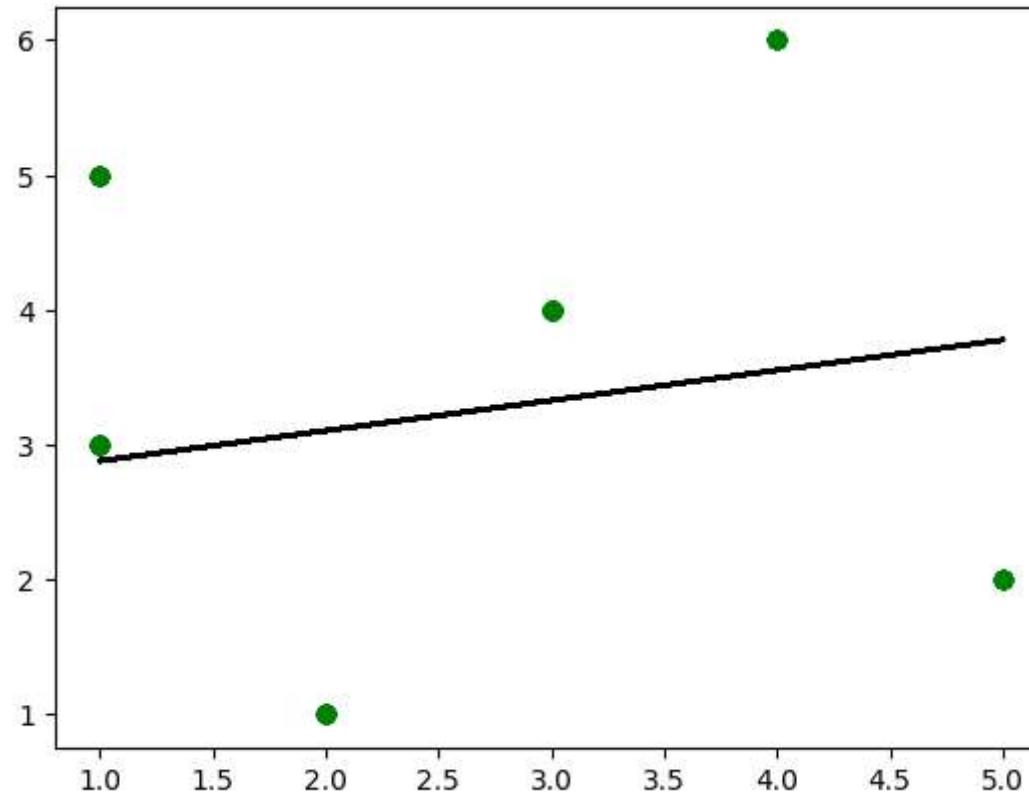
```
In [53]: x=np.array(df500['Source']).reshape(-1,1)
y=np.array(df500['Destination']).reshape(-1,1)
```

```
In [54]: 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)
```

```
In [55]: from sklearn.linear_model import LinearRegression
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

-0.023780796026120488

```
In [56]: y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



```
In [57]: from sklearn.linear_model import LinearRegression
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('r2score:',r2)
```

r2score: -0.023780796026120488

```
In [58]: print(model.intercept_)
```

```
[2.65642869]
```

```
In [68]: features = df.columns[0:2]
```

```
target = df.columns[-1]
```

```
#X and y values
```

```
x = df[features].values
```

```
y = df[target].values
```

```
#splot
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=17)
```

```
print("The dimension of X_train is {}".format(x_train.shape))
```

```
print("The dimension of X_test is {}".format(x_test.shape))
```

```
#Scale features
```

```
scaler = StandardScaler()
```

```
x_train = scaler.fit_transform(x_train)
```

```
x_test = scaler.transform(x_test)
```

```
The dimension of X_train is (7478, 2)
```

```
The dimension of X_test is (3205, 2)
```

```
In [69]: lr = LinearRegression()
```

```
#Fit model
```

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

```
#predict
```

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

```
#actual
```

```
actual = y_test
```

```
train_score_lr = lr.score(x_train, y_train)
```

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

```
print("\nLinear Regression Model:\n")
```

```
print("The train score for lr model is {}".format(train_score_lr))
```

```
print("The test score for lr model is {}".format(test_score_lr))
```

```
#print(lr.score(y_test,prediction))
```

Linear Regression Model:

The train score for lr model is 0.09081762402064897

The test score for lr model is 0.06554871188661315

# Ridge Regression

```
In [70]: from sklearn.linear_model import Ridge,RidgeCV,Lasso  
from sklearn.preprocessing import StandardScaler
```

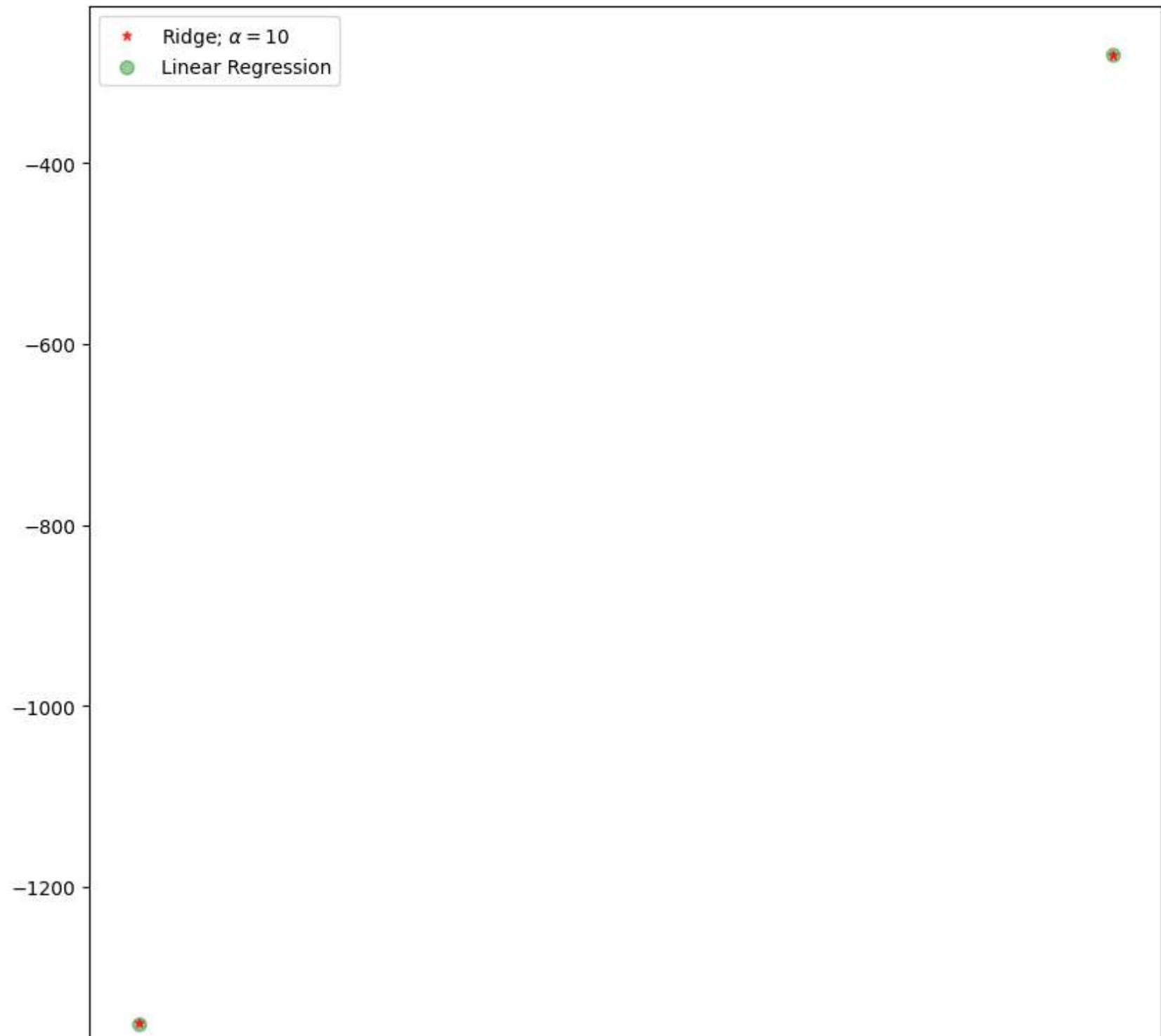
```
In [71]: ridgeReg=Ridge(alpha=10)  
ridgeReg.fit(x_train,y_train)  
train_score_ridge=ridgeReg.score(x_train,y_train)  
test_score_ridge=ridgeReg.score(x_test,y_test)  
print("\nRidge model:\n")  
print("The train score for ridge model is {}".format(train_score_ridge))  
print("The test score for ridge model is {}".format(test_score_ridge))
```

Ridge model:

```
The train score for ridge model is 0.09081746673031499  
The test score for ridge model is 0.06557858312507359
```

```
In [72]: plt.figure(figsize = (10, 10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*',markersize=5,color='red',label=r'Ridge')
# plt.plot(rr100.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'Ridge; $\alpha = 100$')
plt.plot(features,lr.coef_,alpha=0.4,linestyle='none',marker='o',markersize=7,color='green',label='Linear Regressio
plt.xticks(rotation = 90)
plt.legend()
plt.show()
```







## Lasso Regression

```
In [73]: from sklearn.linear_model import LassoCV  
lasso_cv=LassoCV(alphas=[0.0001,0.001,0.01,0.1,1,10],random_state=0).fit(x_train,y_train)  
print(lasso_cv.score(x_train,y_train))  
print(lasso_cv.score(x_test,y_test))
```

```
0.09081762402064841  
0.06554871227546422
```

```
In [74]: print("\nLasso model:\n")  
lasso=Lasso(alpha=10)  
lasso.fit(x_train,y_train)  
train_score_ls=lasso.score(x_train,y_train)  
test_score_ls=lasso.score(x_test,y_test)  
print("The train score for ls model is {}".format(train_score_ls))  
print("The test score for ls model is {}".format(test_score_ls))
```

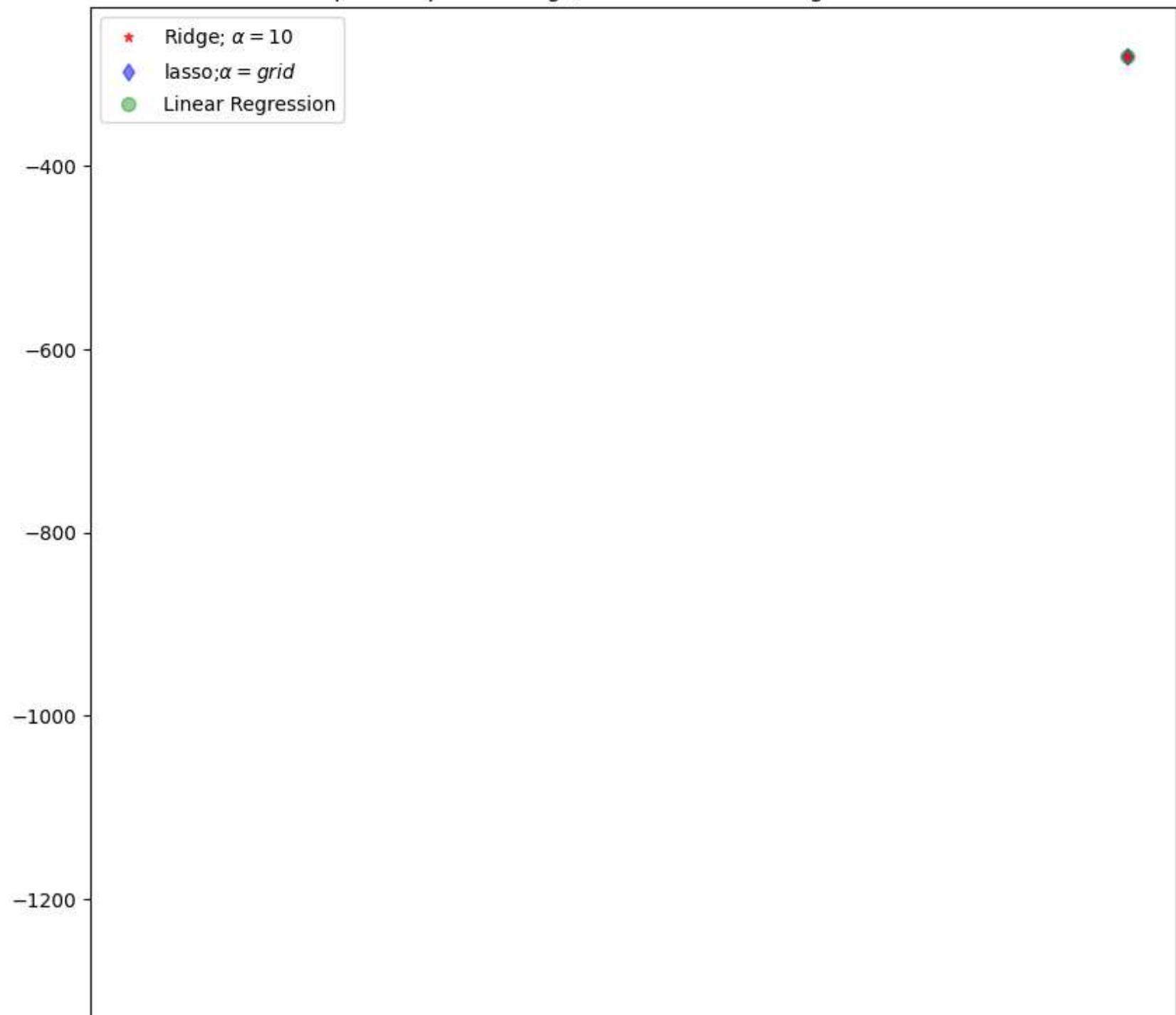
Lasso model:

```
The train score for ls model is 0.09080859244797257  
The test score for ls model is 0.06562448530001963
```

```
In [75]: plt.figure(figsize=(10,10))
plt.plot(features,ridgeReg.coef_,alpha=0.7,linestyle='none',marker='*', markersize=5, color='red',label=r'Ridge')
plt.plot(lasso_cv.coef_,alpha=0.5,linestyle='none',marker='d',markersize=6,color='blue',label=r'lasso;\alpha=0.5')
plt.plot(features, lr.coef_, alpha=0.4, linestyle='none', marker='o', markersize=7, color='green',label='Linear')
plt.xticks(rotation=90)
plt.legend()
plt.title('Comparison plot of Ridge,Lasso and LinearRegression model')
plt.show()
```



## Comparison plot of Ridge,Lasso and LinearRegression model





## Elastic Net

```
In [76]: from sklearn.linear_model import ElasticNet  
regr=ElasticNet()  
regr.fit(x_train,y_train)  
print(regr.coef_)  
print(regr.intercept_)
```

```
[-903.1071039 -198.453344 ]  
9088.521396095213
```

```
In [77]: y_pred_elastic=regr.predict(x_train)
```

```
In [79]: mean_squared_error=np.mean((y_pred_elastic- y_train)**2)  
print("mean_squared_error",mean_squared_error)
```

```
mean_squared_error 19593419.936366197
```

## Logistic Regression

```
In [89]: import re
from sklearn.linear_model import LogisticRegression
x=np.array(df['Price']).reshape(-1,1)
y=np.array(df['Total_Stops']).reshape(-1,1)
df.dropna(inplace=True)
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=1)
LogisticRegr=LogisticRegression(max_iter=10000)
```

```
In [90]: LogisticRegr.fit(x_train,y_train)
```

C:\Users\USER\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 to (n\_samples, ), for example using ravel().  
y = column\_or\_1d(y, warn=True)

```
Out[90]: LogisticRegression
```

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

```
In [93]: score=LogisticRegr.score(x_test,y_test)
print(score)
```

0.7101404056162246

## Decision Tree

```
In [94]: from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=0)
```

```
In [95]: clf.fit(x_train,y_train)
```

```
Out[95]: DecisionTreeClassifier
```

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

```
In [97]: score=(clf.score(x_test,y_test))  
print(score)
```

```
0.9360374414976599
```

## Random Forest

```
In [98]: from sklearn.ensemble import RandomForestClassifier  
rfc=RandomForestClassifier()  
rfc.fit(x_train,y_train)
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_3468\4250526780.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().  
    rfc.fit(x_train,y_train)
```

```
Out[98]: RandomForestClassifier()  
        |  
        | RandomForestClassifier()
```

```
In [99]: rf=RandomForestClassifier()
```

```
In [100]: 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 [101]: 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)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    estimator.fit(X_train, y_train, **fit_params)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    estimator.fit(X_train, y_train, **fit_params)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    estimator.fit(X_train, y_train, **fit_params)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    estimator.fit(X_train, y_train, **fit_params)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
    estimator.fit(X_train, y_train, **fit_params)
C:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\model_selection\_validation.py:686: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

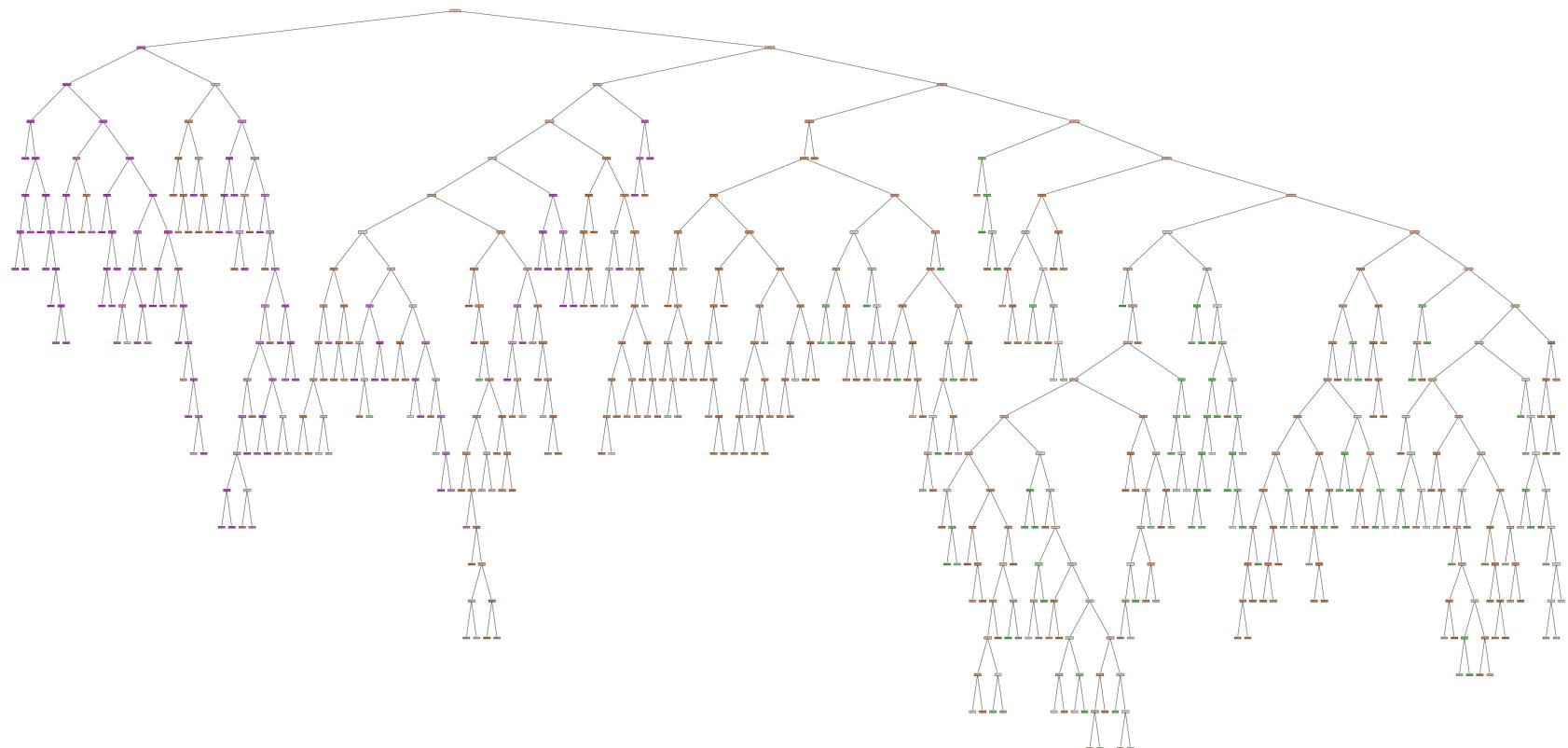
```
In [102]: grid_search.best_score_
```

```
Out[102]: 0.8697512703931533
```

```
In [103]: rf_best=grid_search.best_estimator_
print(rf_best)
```

```
RandomForestClassifier(max_depth=20, min_samples_leaf=5, n_estimators=30)
```

```
In [106]: from sklearn.tree import plot_tree
plt.figure(figsize=(80,40))
plot_tree(rf_best.estimators_[5], class_names=['0','1','2','3','4'], filled=True);
```



```
In [109]: score=rfc.score(x_test,y_test)
print(rfc)
```

```
RandomForestClassifier()
```

## conclusion

In [ ]: for the above data set we use different types of models,for that each and every models we get different types of accuracies.Based on that accuracies we can conclude which model is best fit for my our dataset. here we get different types of accuracies for that different types of accuracies decision tree is get more accuracy among all the models.so,that we can conclude that for our model decision tree is best fit

In [ ]: