In [1]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression,LogisticRegression,Lasso,Rid
```

In [2]:
```
df=pd.read_csv(r"C:\Users\USER\Downloads\GemDataEXTR\Stock Markets, US$.csv")
df.fillna(0,inplace=True)
df
```

Out[2]:

	Unnamed: 0	United Arab Emirates	Argentina	Australia	Austria	Belgium	Bulgaria	Bahrain
0	0.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	1994.0	0.00000	0.00000	34.00822	35.52493	48.62237	0.00000	0.00000
2	1995.0	0.00000	0.00000	33.98690	35.29171	52.61066	0.00000	0.00000
3	1996.0	0.00000	23.91518	40.10000	37.32716	62.57487	0.00000	0.00000
4	1997.0	0.00000	31.96844	42.72983	39.26318	72.09075	0.00000	0.00000
5	1998.0	0.00000	23.74431	37.92748	39.90513	96.45048	0.00000	0.00000
6	1999.0	0.00000	20.79165	43.27264	33.73018	95.57480	0.00000	0.00000
7	2000.0	0.00000	21.45833	41.63047	28.07375	76.22327	15.20993	0.00000
8	2001.0	0.00000	15.60000	37.80406	27.92258	69.96057	13.33601	0.00000
9	2002.0	0.00000	17.03334	38.87390	30.06042	62.99667	21.52815	0.00000
10	2003.0	0.00000	31.23999	45.01805	39.87157	62.28960	57.93172	0.00000
11	2004.0	0.00000	47.17606	58.96969	66.37044	88.58039	106.46260	0.00000
12	2005.0	0.00000	63.65470	73.40358	99.87017	110.41210	161.40700	152.71750
13	2006.0	261.13640	74.62035	86.26591	133.30150	136.54990	193.55570	147.13950
14	2007.0	261.48620	92.00042	117.79100	170.40430	168.11300	336.26270	162.97730
15	2008.0	274.42360	75.43766	98.45246	135.30620	129.62830	247.21010	178.42820
16	2009.0	106.64360	69.37860	73.60899	80.66908	83.35619	86.50294	106.92280
17	2010.0	96.59922	108.55270	97.02012	91.50184	94.37587	84.26382	99.85734
18	2011.0	87.98379	132.49440	106.18030	92.73847	93.90413	88.36709	89.38204
19	2012.0	91.33702	106.33210	101.52530	72.82167	81.94432	65.31064	75.61385
20	2013.0	140.14890	169.72330	110.36950	87.86638	99.49884	92.13893	79.03537
21	2014.0	263.19360	343.97360	109.97990	86.04282	115.05660	120.31140	96.58291
22	2015.0	219.16210	472.52220	93.21922	72.52466	111.54020	85.39854	91.81821
23	2016.0	195.63080	616.53210	89.60355	68.32895	106.37880	85.12999	78.55283
24	2017.0	207.04250	968.56580	101.10050	94.53688	121.90990	121.75700	89.06664
25	2018.0	174.03290	1296.93600	102.70640	105.94740	124.20880	123.17940	90.13910
26	2019.0	158.27540	1467.23100	102.22300	91.47970	113.59450	103.81630	100.50620
27	2020.0	131.64600	1806.09800	97.35689	74.12048	108.41230	85.20527	96.70240
28	2021.0	164.58670	2782.20700	125.90400	109.20570	134.21610	104.75880	108.41600
29	2022.0	196.66440	4923.64200	114.18860	91.17821	112.71390	103.95410	129.80530
30	2023.0	204.16430	13083.16000	113.31150	94.60561	112.65920	111.60790	130.73140

31 rows × 79 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31 entries, 0 to 30
Data columns (total 79 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              31 non-null     float64
 1   United Arab Emirates    31 non-null     float64
 2   Argentina               31 non-null     float64
 3   Australia               31 non-null     float64
 4   Austria                 31 non-null     float64
 5   Belgium                 31 non-null     float64
 6   Bulgaria                31 non-null     float64
 7   Bahrain                 31 non-null     float64
 8   Bosnia and Herzegovina  31 non-null     float64
 9   Brazil                  31 non-null     float64
 10  Canada                  31 non-null     float64
 11  Switzerland             31 non-null     float64
 12  Chile                   31 non-null     float64
 13  China                   31 non-null     float64
 14  Colombia                31 non-null     float64
 15  Czech Republic          31 non-null     float64
 16  Germany                 31 non-null     float64
 17  Denmark                 31 non-null     float64
 18  Egypt, Arab Rep.        31 non-null     float64
 19  Spain                   31 non-null     float64
 20  Estonia                 31 non-null     float64
 21  Finland                 31 non-null     float64
 22  France                  31 non-null     float64
 23  United Kingdom          31 non-null     float64
 24  Greece                  31 non-null     float64
 25  High Income Countries   31 non-null     float64
 26  Hong Kong SAR, China    31 non-null     float64
 27  Croatia                 31 non-null     float64
 28  Hungary                 31 non-null     float64
 29  Indonesia               31 non-null     float64
 30  India                   31 non-null     float64
 31  Ireland                 31 non-null     float64
 32  Iran, Islamic Rep.      31 non-null     float64
 33  Iceland                 31 non-null     float64
 34  Israel                  31 non-null     float64
 35  Italy                   31 non-null     float64
 36  Jordan                  31 non-null     float64
 37  Japan                   31 non-null     float64
 38  Kazakhstan              31 non-null     float64
 39  Kenya                   31 non-null     float64
 40  Korea, Rep.             31 non-null     float64
 41  Sri Lanka               31 non-null     float64
 42  Lithuania               31 non-null     float64
 43  Luxembourg              31 non-null     float64
 44  Latvia                  31 non-null     float64
 45  Morocco                 31 non-null     float64
 46  Mexico                  31 non-null     float64
 47  North Macedonia         31 non-null     float64
 48  Malta                   31 non-null     float64
```

```
 49  Malawi              31 non-null     float64
 50  Malaysia            31 non-null     float64
 51  Nigeria             31 non-null     float64
 52  Netherlands         31 non-null     float64
 53  Norway              31 non-null     float64
 54  New Zealand         31 non-null     float64
 55  Oman                31 non-null     float64
 56  Pakistan            31 non-null     float64
 57  Peru                31 non-null     float64
 58  Philippines         31 non-null     float64
 59  Poland              31 non-null     float64
 60  Portugal            31 non-null     float64
 61  Qatar               31 non-null     float64
 62  Romania             31 non-null     float64
 63  Russian Federation  31 non-null     float64
 64  Saudi Arabia        31 non-null     float64
 65  Singapore           31 non-null     float64
 66  Slovakia            31 non-null     float64
 67  Slovenia            31 non-null     float64
 68  Sweden              31 non-null     float64
 69  Thailand            31 non-null     float64
 70  Tunisia             31 non-null     float64
 71  Turkey              31 non-null     float64
 72  Taiwan, China       31 non-null     float64
 73  Uganda              31 non-null     float64
 74  Ukraine             31 non-null     float64
 75  United States       31 non-null     float64
 76  Venezuela, RB       31 non-null     float64
 77  Vietnam             31 non-null     float64
 78  South Africa        31 non-null     float64
dtypes: float64(79)
memory usage: 19.3 KB
```

In [4]:
```
df1=df.dropna()
```

Out[4]:

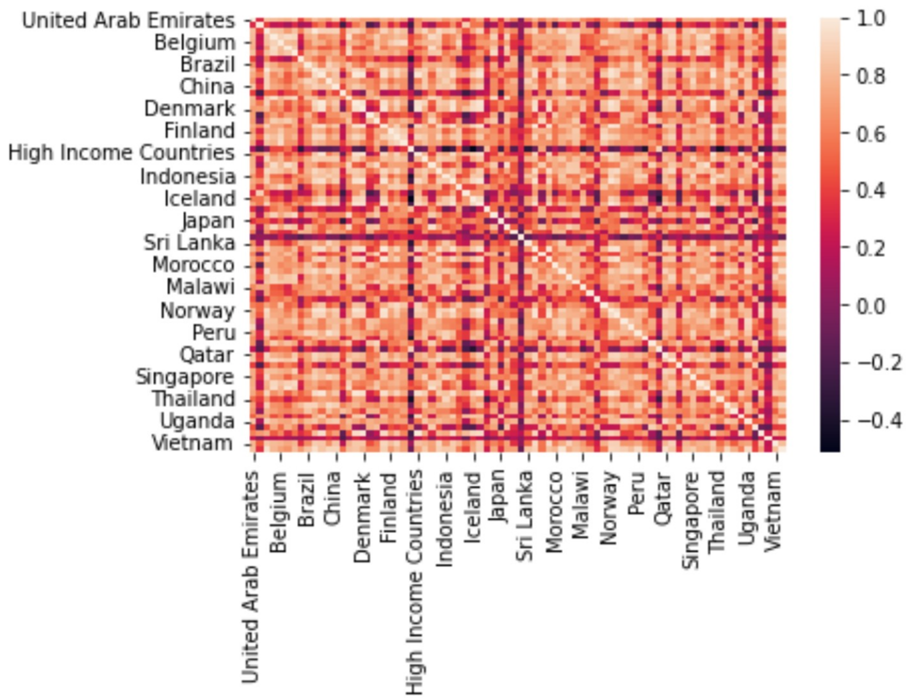
	Unnamed: 0	United Arab Emirates	Argentina	Australia	Austria	Belgium	Bulgaria	Bahrain
0	0.0	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	1994.0	0.00000	0.00000	34.00822	35.52493	48.62237	0.00000	0.00000
2	1995.0	0.00000	0.00000	33.98690	35.29171	52.61066	0.00000	0.00000
3	1996.0	0.00000	23.91518	40.10000	37.32716	62.57487	0.00000	0.00000
4	1997.0	0.00000	31.96844	42.72983	39.26318	72.09075	0.00000	0.00000
5	1998.0	0.00000	23.74431	37.92748	39.90513	96.45048	0.00000	0.00000
6	1999.0	0.00000	20.79165	43.27264	33.73018	95.57480	0.00000	0.00000
7	2000.0	0.00000	21.45833	41.63047	28.07375	76.22327	15.20993	0.00000
8	2001.0	0.00000	15.60000	37.80406	27.92258	69.96057	13.33601	0.00000
9	2002.0	0.00000	17.03334	38.87390	30.06042	62.99667	21.52815	0.00000
10	2003.0	0.00000	31.23999	45.01805	39.87157	62.28960	57.93172	0.00000
11	2004.0	0.00000	47.17606	58.96969	66.37044	88.58039	106.46260	0.00000
12	2005.0	0.00000	63.65470	73.40358	99.87017	110.41210	161.40700	152.71750
13	2006.0	261.13640	74.62035	86.26591	133.30150	136.54990	193.55570	147.13950
14	2007.0	261.48620	92.00042	117.79100	170.40430	168.11300	336.26270	162.97730
15	2008.0	274.42360	75.43766	98.45246	135.30620	129.62830	247.21010	178.42820
16	2009.0	106.64360	69.37860	73.60899	80.66908	83.35619	86.50294	106.92280
17	2010.0	96.59922	108.55270	97.02012	91.50184	94.37587	84.26382	99.85734
18	2011.0	87.98379	132.49440	106.18030	92.73847	93.90413	88.36709	89.38204
19	2012.0	91.33702	106.33210	101.52530	72.82167	81.94432	65.31064	75.61385
20	2013.0	140.14890	169.72330	110.36950	87.86638	99.49884	92.13893	79.03537
21	2014.0	263.19360	343.97360	109.97990	86.04282	115.05660	120.31140	96.58291
22	2015.0	219.16210	472.52220	93.21922	72.52466	111.54020	85.39854	91.81821
23	2016.0	195.63080	616.53210	89.60355	68.32895	106.37880	85.12999	78.55283
24	2017.0	207.04250	968.56580	101.10050	94.53688	121.90990	121.75700	89.06664
25	2018.0	174.03290	1296.93600	102.70640	105.94740	124.20880	123.17940	90.13910
26	2019.0	158.27540	1467.23100	102.22300	91.47970	113.59450	103.81630	100.50620
27	2020.0	131.64600	1806.09800	97.35689	74.12048	108.41230	85.20527	96.70240
28	2021.0	164.58670	2782.20700	125.90400	109.20570	134.21610	104.75880	108.41600
29	2022.0	196.66440	4923.64200	114.18860	91.17821	112.71390	103.95410	129.80530
30	2023.0	204.16430	13083.16000	113.31150	94.60561	112.65920	111.60790	130.73140

31 rows × 79 columns

In [5]:

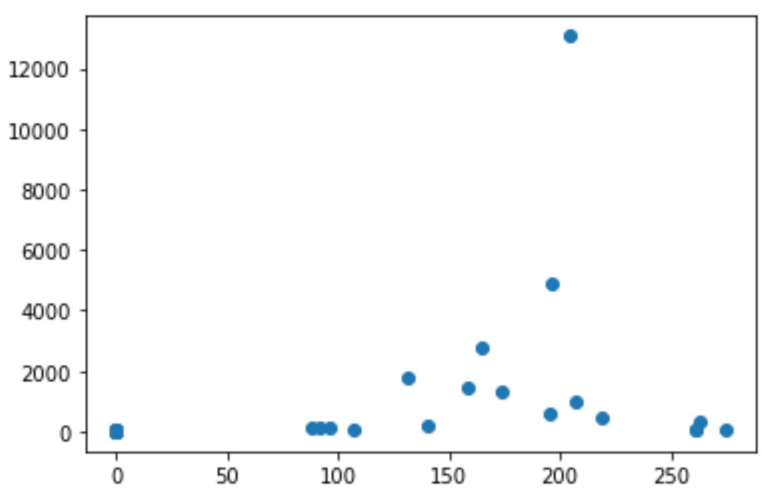
In [6]:

Out[6]: <AxesSubplot:>



In [7]:

Out[7]: [<matplotlib.lines.Line2D at 0x1708590df70>]



In [8]:

In [9]:

```
x=df1.drop(["United Arab Emirates"],axis=1)
y=df1["United Arab Emirates"]
```

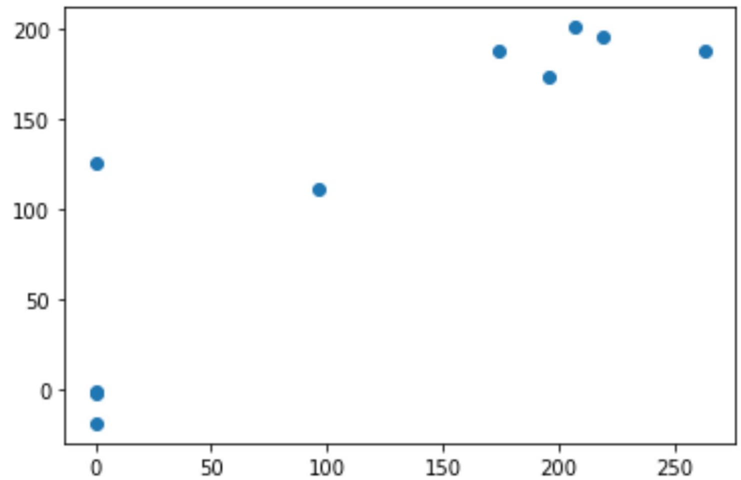
Linear

In [10]:
```
li=LinearRegression()
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
prediction=li.predict(x_test)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x17085d4ed30>
```



In [12]:

In [13]:

Out[13]:
```
0.00000        3
108.55270      1
4923.64200     1
2782.20700     1
1806.09800     1
1467.23100     1
1296.93600     1
968.56580      1
616.53210      1
472.52220      1
343.97360      1
169.72330      1
106.33210      1
132.49440      1
69.37860       1
23.91518       1
75.43766       1
92.00042       1
74.62035       1
63.65470       1
47.17606       1
31.23999       1
17.03334       1
15.60000       1
21.45833       1
20.79165       1
23.74431       1
31.96844       1
13083.16000    1
Name: Argentina, dtype: int64
```

In [14]:
```
df1.loc[df1["Argentina"]<1.40,"Argentina"]=1
df1.loc[df1["Argentina"]>1.40,"Argentina"]=2
```

Out[14]:
```
2.0    28
1.0     3
Name: Argentina, dtype: int64
```

Lasso

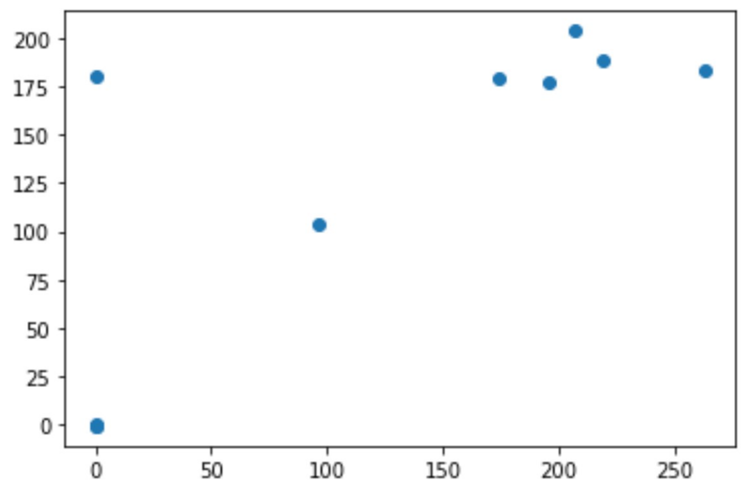
In [15]:
```
la=Lasso(alpha=5)
```

```
C:\Users\USER\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_de
scent.py:647: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations, check the scale of the features or cons
ider increasing regularisation. Duality gap: 3.428e+02, tolerance: 2.031e+01
  model = cd_fast.enet_coordinate_descent(
```

Out[15]:
```
Lasso(alpha=5)
```

In [16]:
```
prediction1=la.predict(x_test)
```

Out[16]: <matplotlib.collections.PathCollection at 0x17085e09580>



In [17]:
```
```

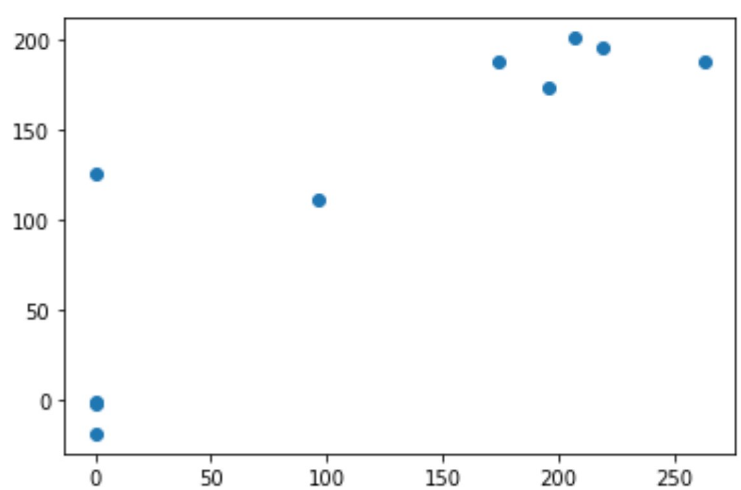
Ridge

In [18]:
```
rr=Ridge(alpha=1)
```

Out[18]: Ridge(alpha=1)

In [19]:
```
prediction2=rr.predict(x_test)
```

Out[19]: <matplotlib.collections.PathCollection at 0x17085e72fa0>



In [20]:
```
```

ElasticNet

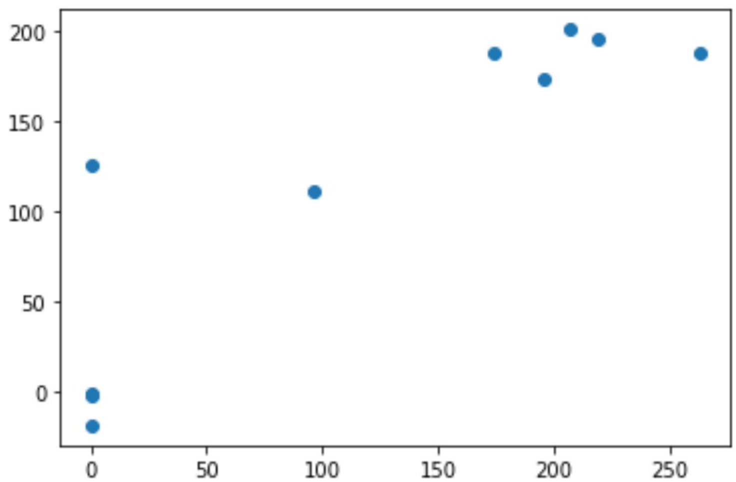
In [21]:
```
en=ElasticNet()
en.fit(x_train,y_train)
```

```
C:\Users\USER\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_de
scent.py:647: ConvergenceWarning: Objective did not converge. You might want
to increase the number of iterations, check the scale of the features or cons
ider increasing regularisation. Duality gap: 9.255e+01, tolerance: 2.031e+01
  model = cd_fast.enet_coordinate_descent(
```

Out[21]:
```
ElasticNet()
```

In [22]:
```
prediction2=rr.predict(x_test)
```

Out[22]:
```
<matplotlib.collections.PathCollection at 0x17086f2feb0>
```



In [23]:

In [24]:
```
print(rr.score(x_test,y_test))
```

```
0.7763320213189454
```

Out[24]:
```
0.9999999995333385
```

Logistic

In [25]:
```
g={"Argentina":{1.0:"Low",2.0:"High"}}
df1=df1.replace(g)
```

Out[25]:
```
High    28
Low      3
Name: Argentina, dtype: int64
```

In [26]:
```
x=df1.drop(["Argentina"],axis=1)
y=df1["Argentina"]
```

In [27]:
```
lo=LogisticRegression()
```

Out[27]: LogisticRegression()

In [28]:
```
prediction3=lo.predict(x_test)
```

Out[28]: <matplotlib.collections.PathCollection at 0x17087f759d0>

In [29]:

Random Forest

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [31]:
```
g1={"Argentina":{"Low":1.0,"High":2.0}}
```

In [32]:
```
x=df1.drop(["Argentina"],axis=1)
y=df1["Argentina"]
```

In [33]:
```
rfc=RandomForestClassifier()
```

Out[33]: RandomForestClassifier()

In [34]:
```
parameter={
    'max_depth':[1,2,4,5,6],
    'min_samples_leaf':[5,10,15,20,25],
    'n_estimators':[10,20,30,40,50]
```

In [35]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accu
```

Out[35]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 4, 5, 6],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

In [36]:

In [37]:

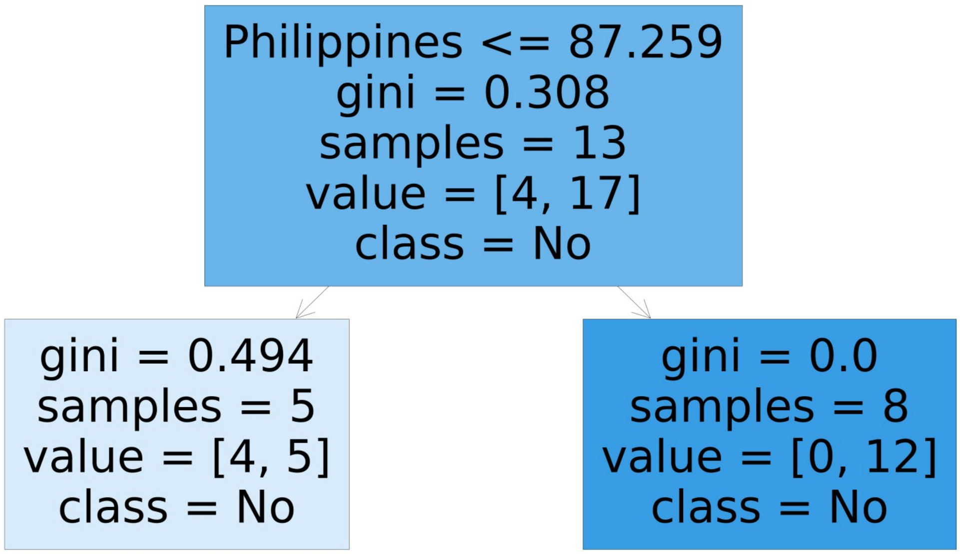
In [38]:

```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
```

Out[38]:

```
[Text(0.5, 0.75, 'Philippines <= 87.259\ngini = 0.308\nsamples = 13\nvalue =
[4, 17]\nclass = No'),
 Text(0.25, 0.25, 'gini = 0.494\nsamples = 5\nvalue = [4, 5]\nclass = No'),
 Text(0.75, 0.25, 'gini = 0.0\nsamples = 8\nvalue = [0, 12]\nclass = No')]
```



In [39]:

```
print("Linear:",lis)
print("Lasso:",las)
print("Ridge:",rrs)
print("ElasticNet:",ens)
print("Logistic:",los)
```

```
Linear: 0.7763562931025759
Lasso: 0.6157514963056996
Ridge: 0.7763320213189454
ElasticNet: 0.7062968997489696
Logistic: 1.0
Random Forest: 0.8590909090909091
```