```
LINEAR REGRESSION
DATA
```

In [10]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [8]:

```
data1 = pd.read_csv(r"C:\Users\AKHILA\Downloads\1_2015.csv")
```

In [9]:

```
data1.describe()
```

Out[9]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730

In [11]:

```
data1.head()
```

Out[11]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	F
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	

In [12]:

```
data1.tail()
```

Out[12]:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	F
153	Rwanda	Sub-Saharan Africa	154	3.465	0.03464	0.22208	0.77370	0.42864	
154	Benin	Sub-Saharan Africa	155	3.340	0.03656	0.28665	0.35386	0.31910	
155	Syria	Middle East and Northern Africa	156	3.006	0.05015	0.66320	0.47489	0.72193	
156	Burundi	Sub-Saharan Africa	157	2.905	0.08658	0.01530	0.41587	0.22396	
157	Togo	Sub-Saharan Africa	158	2.839	0.06727	0.20868	0.13995	0.28443	

In [13]:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        158 non-null    object 
 1   Region                         158 non-null    object 
 2   Happiness Rank                 158 non-null    int64  
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

In [14]:

```
data1.columns
```

Out[14]:

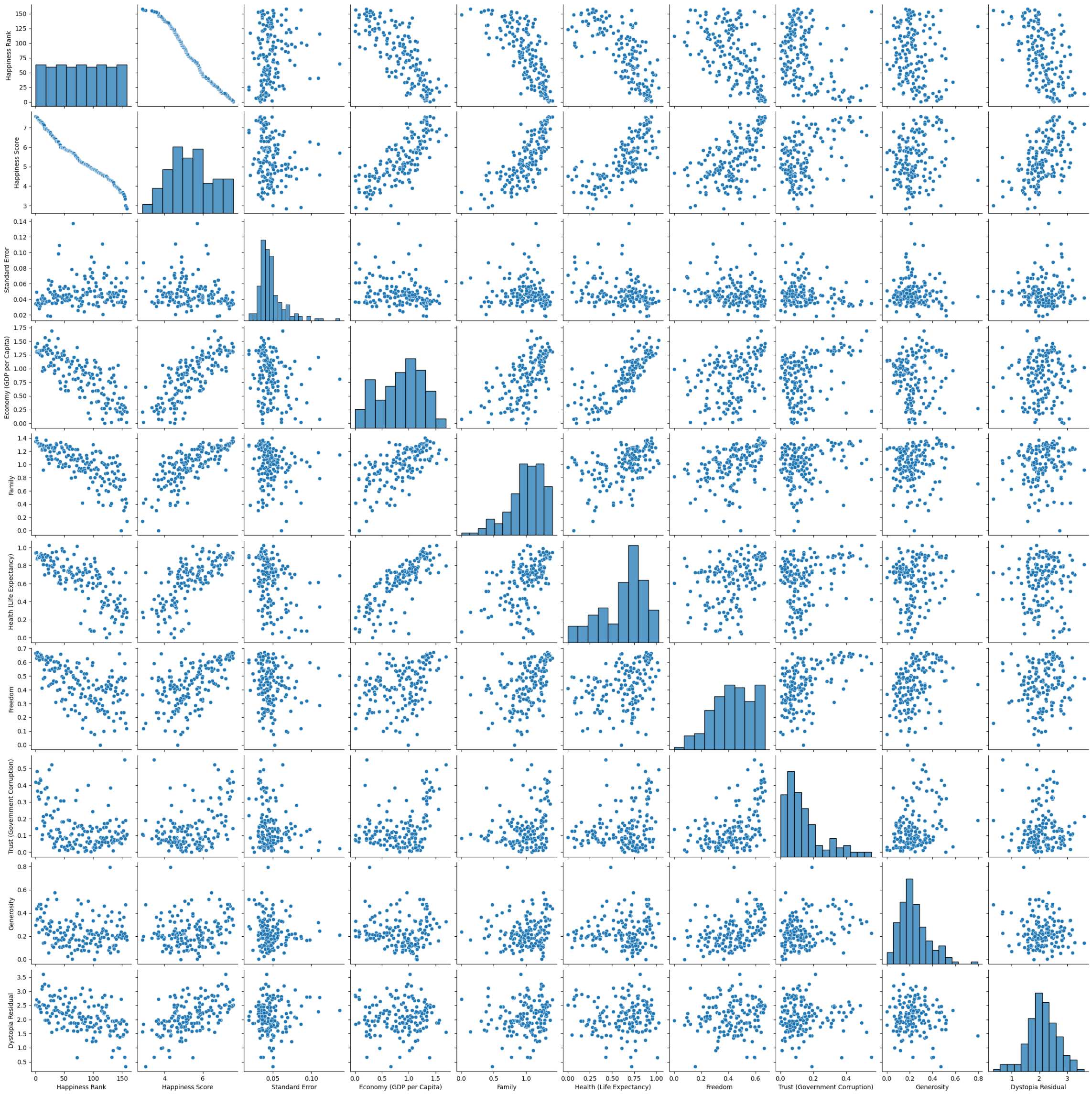
```
Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruptio
n)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

In [15]:

```
sns.pairplot(data1)
```

Out[15]:

<seaborn.axisgrid.PairGrid at 0x1fe4fa25190>

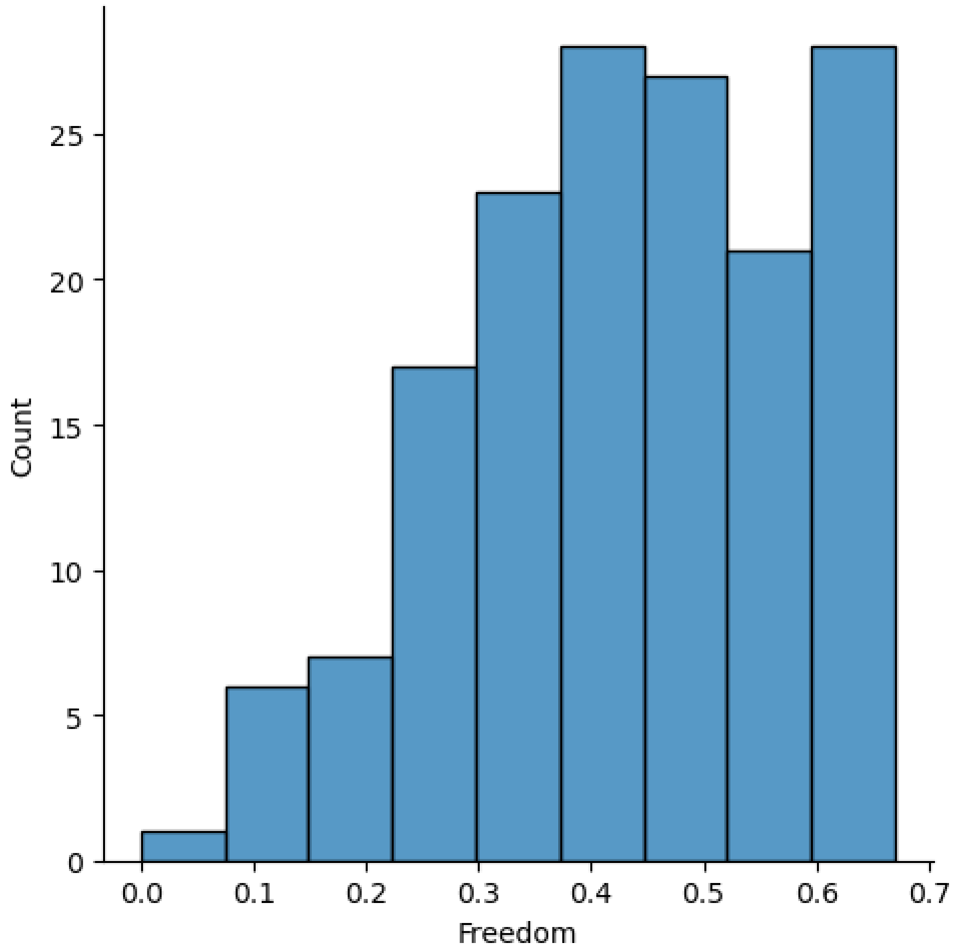


In [16]:

```
sns.displot(data1['Freedom'])
```

Out[16]:

<seaborn.axisgrid.FacetGrid at 0x1fe545c94d0>



In [17]:

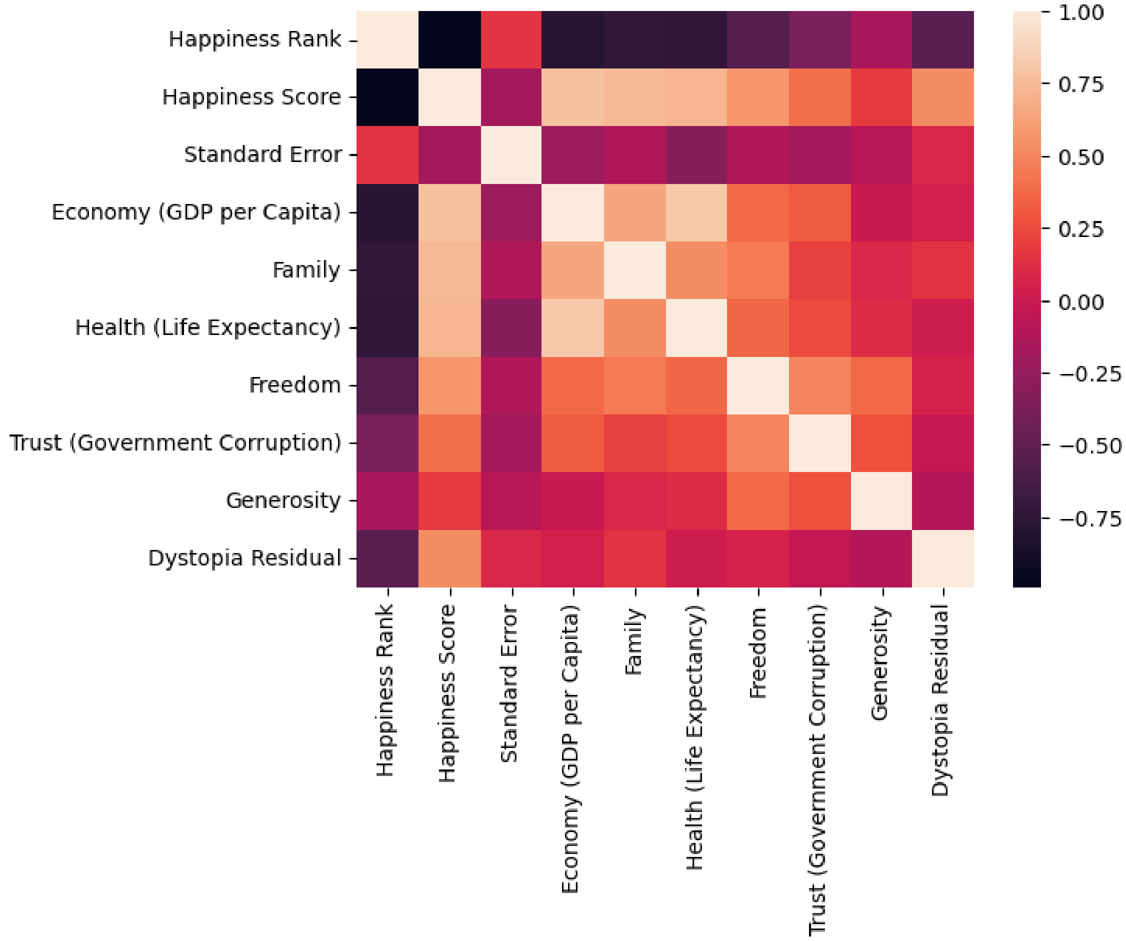
```
new_data1 = data1[['Happiness Rank', 'Happiness Score',
 'Standard Error', 'Economy (GDP per Capita)', 'Family',
 'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
 'Generosity', 'Dystopia Residual']]
```

In [18]:

```
sns.heatmap(new_data1.corr())
```

Out[18]:

```
<Axes: >
```




```
MODEL BUILDING FOR DATA1
```

In [19]:

```
X = new_data1[['Happiness Rank', 'Happiness Score',
 'Standard Error', 'Economy (GDP per Capita)', 'Family',
 'Health (Life Expectancy)', 'Trust (Government Corruption)',
 'Generosity', 'Dystopia Residual']]
y = data1['Freedom']
```

In [21]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.30)
```

In [22]:

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[22]:

```
▾ LinearRegression
LinearRegression()
```

In [24]:

```
#Prediction
predX = lr.predict(X_test)
print(predX)
```

```
[0.62453962 0.48895909 0.3887443  0.66573731 0.41483713 0.25147207
 0.2443236  0.51506286 0.46058424 0.59632617 0.53221435 0.24520173
 0.46858129 0.58419561 0.41685166 0.53489455 0.31865778 0.44011625
 0.53866218 0.39505185 0.53164933 0.42182889 0.38243466 0.46635209
 0.60354277 0.45527026 0.20091261 0.30631689 0.6512724  0.57723779
 0.60322881 0.4229416  0.43440833 0.41626831 0.61755439 0.39724233
 0.26309463 0.5298249  0.49093038 0.64016108 0.10017369 0.17260508
 0.4769477  0.66021869 0.28480765 0.55692497 0.37036783 0.11883106]
```

In [25]:

```
#Accuracy
print(lr.score(X_test,y_test))
```

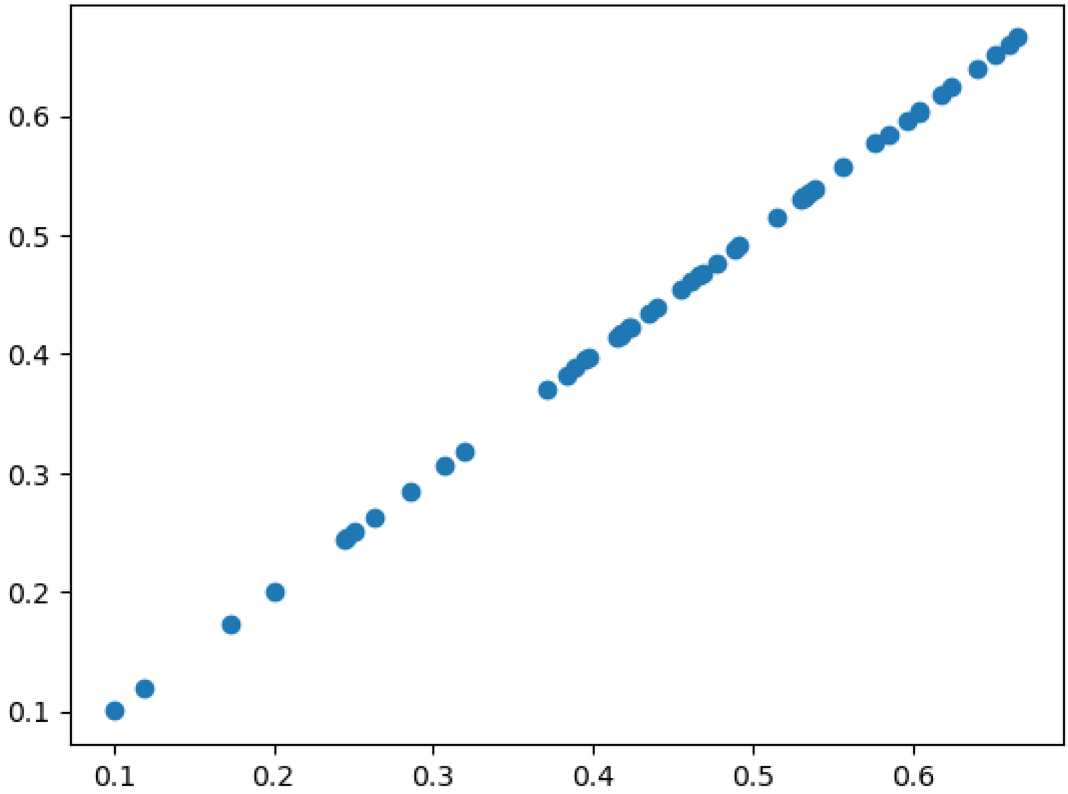
```
0.9999955087324011
```

In [26]:

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

Out[26]:

```
<matplotlib.collections.PathCollection at 0x1fe5804d190>
```




```
DATA 2
```

In [27]:

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

In [28]:

```
data2 = pd.read_csv(r"C:\Users\AKHILA\Downloads\4_Drug200.csv")
```

In [29]:

```
data2.describe()
```

Out[29]:

	Age	Na_to_K
count	200.000000	200.000000
mean	44.315000	16.084485
std	16.544315	7.223956
min	15.000000	6.269000
25%	31.000000	10.445500
50%	45.000000	13.936500
75%	58.000000	19.380000
max	74.000000	38.247000

In [30]:

```
data2.head()
```

Out[30]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

In [31]:

```
data2.tail()
```

Out[31]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
195	56	F	LOW	HIGH	11.567	drugC
196	16	M	LOW	HIGH	12.006	drugC
197	52	M	NORMAL	HIGH	9.894	drugX
198	23	M	NORMAL	NORMAL	14.020	drugX
199	40	F	LOW	NORMAL	11.349	drugX

In [32]:

```
data2.columns
```

Out[32]:

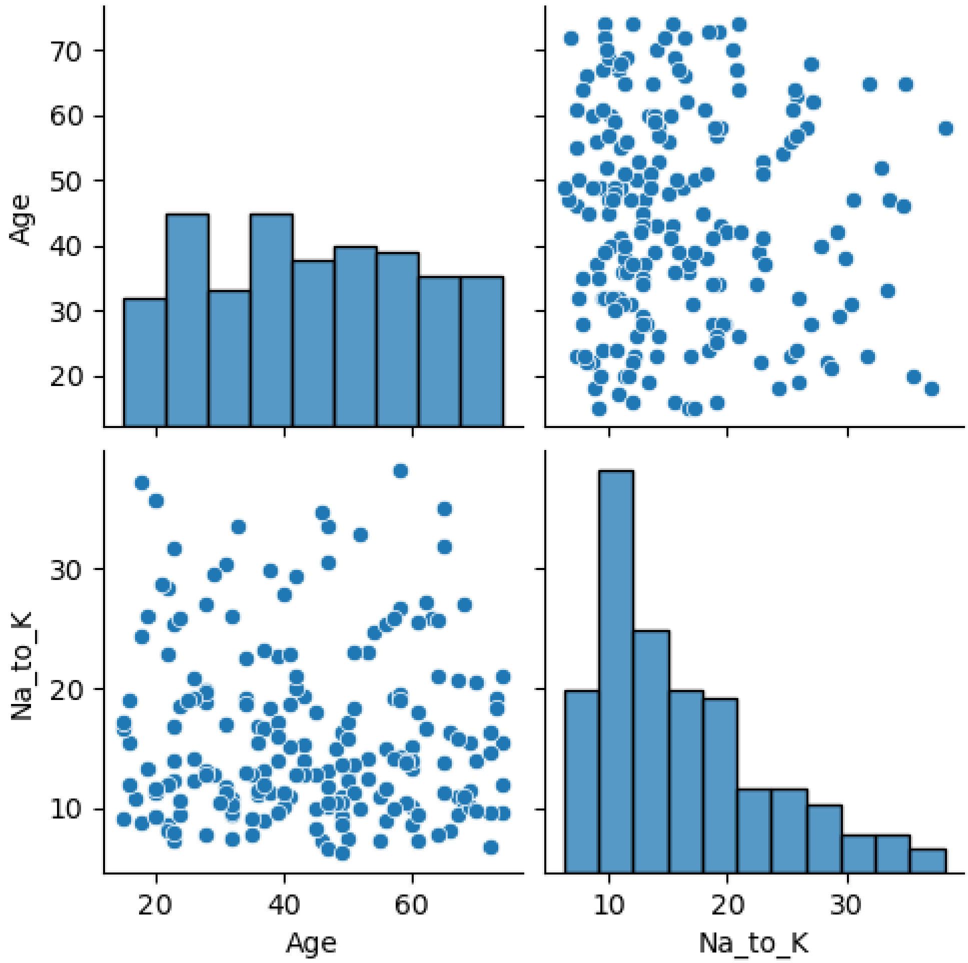
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='objec
t')
```

In [33]:

```
sns.pairplot(data2)
```

Out[33]:

```
<seaborn.axisgrid.PairGrid at 0x1fe5801e0d0>
```



In [34]:

```
#Changing High, Normal and Low to 2, 1 and 0 respectively...
BP = {"BP":{"LOW":0,"NORMAL":1,"HIGH":2}}
data2 = data2.replace(BP)
Cholesterol = {"Cholesterol":{"LOW":0,"NORMAL":1,"HIGH":2}}
data2 = data2.replace(Cholesterol)
data2
```

Out[34]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	2	2	25.355	drugY
1	47	M	0	2	13.093	drugC
2	47	M	0	2	10.114	drugC
3	28	F	1	2	7.798	drugX
4	61	F	0	2	18.043	drugY
...	...	...	...	...	...	...
195	56	F	0	2	11.567	drugC
196	16	M	0	2	12.006	drugC
197	52	M	1	2	9.894	drugX
198	23	M	1	1	14.020	drugX
199	40	F	0	1	11.349	drugX

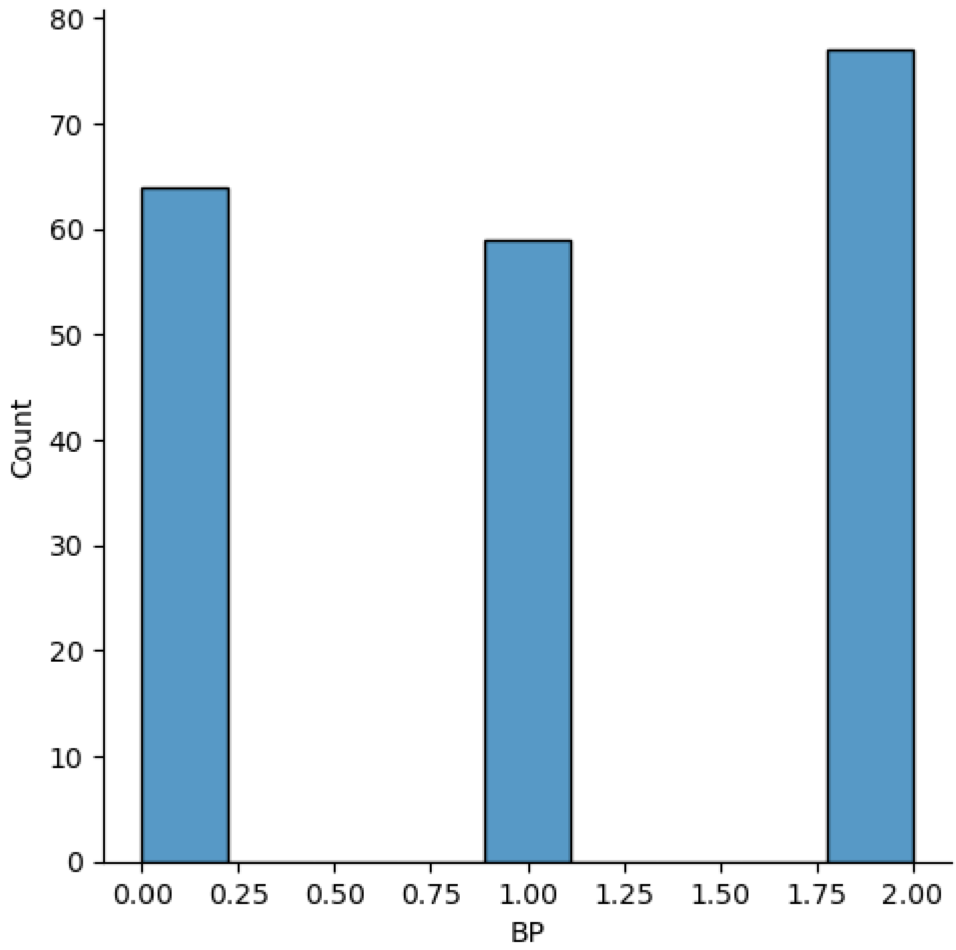
200 rows × 6 columns

In [35]:

```
sns.displot(data2['BP'])
```

Out[35]:

```
<seaborn.axisgrid.FacetGrid at 0x1fe580a1650>
```



In [36]:

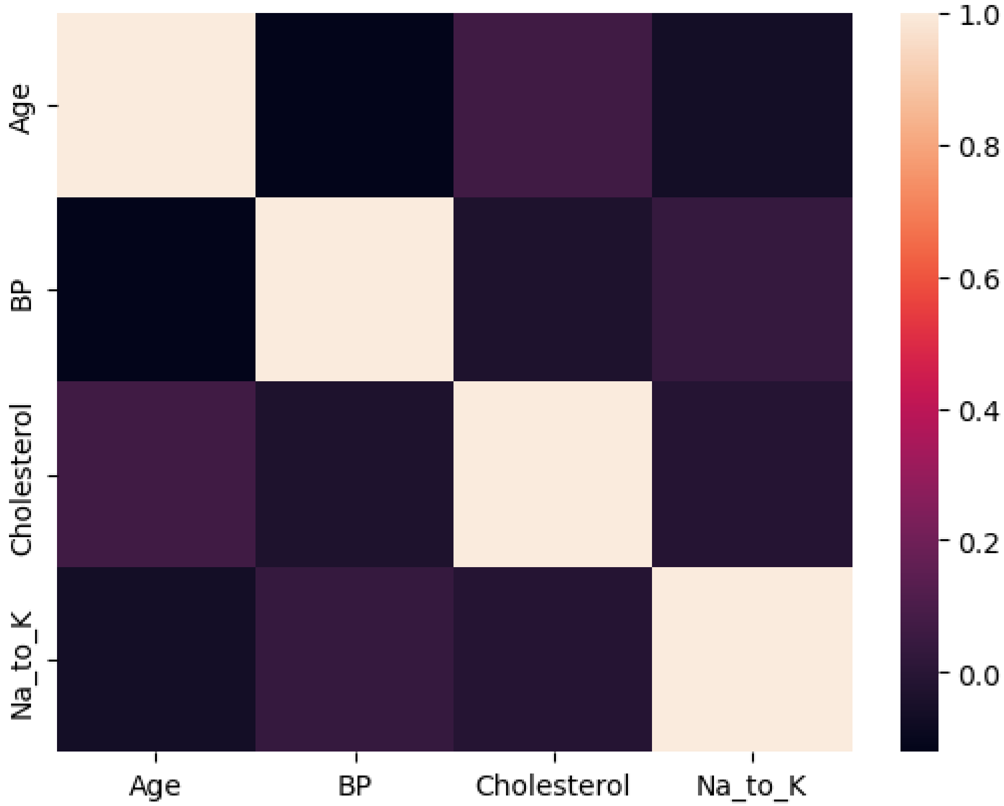
```
new_data2 = data2[['Age', 'BP', 'Cholesterol', 'Na_to_K']]
```

In [37]:

```
sns.heatmap(new_data2.corr())
```

Out[37]:

```
&lt;Axes: &gt;
```




```
MODEL BUILDING FOR DATA2
```

In [38]:

```
X = new_data2[['Age', 'Cholesterol', 'Na_to_K']]
y = data2['BP']
```

In [39]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.30)
```

In [40]:

```
from sklearn.linear_model import LinearRegression
```

In [41]:

```
lr=LinearRegression()
lr.fit(X_train,y_train)
```

Out[41]:

```
▾ LinearRegression

LinearRegression()
```

In [42]:

```
#Prediction
predX = lr.predict(X_test)
print(predX)
```

```
[1.23841392 1.23072263 1.03160715 1.23108124 1.22226682 1.05516537
 0.98868878 0.92616347 1.05516375 1.26518257 1.15913418 0.97170118
 0.95071506 1.1920163  0.86384882 0.90074657 1.20160695 1.10109498
 1.24977656 1.11134019 1.1365969  1.14711042 1.00025615 0.91728279
 1.22031298 0.98893481 1.19730414 1.16033698 1.0529568  1.05137644
 1.27602965 1.1605424  1.23058139 1.1892954  1.10247716 1.01907854
 1.11352342 0.91459748 1.1631028  0.92957931 1.25872482 1.12921236
 0.86797911 1.10071873 1.21715257 1.07726067 1.13209191 1.2956641
 1.27438726 0.86892208 1.11292479 1.18576759 1.11521296 1.0193755
 1.1126615  1.18999666 1.31843808 1.01562787 0.96851743 0.96875359]
```

In [43]:

```
#Accuracy
print(lr.score(X_test,y_test))
```

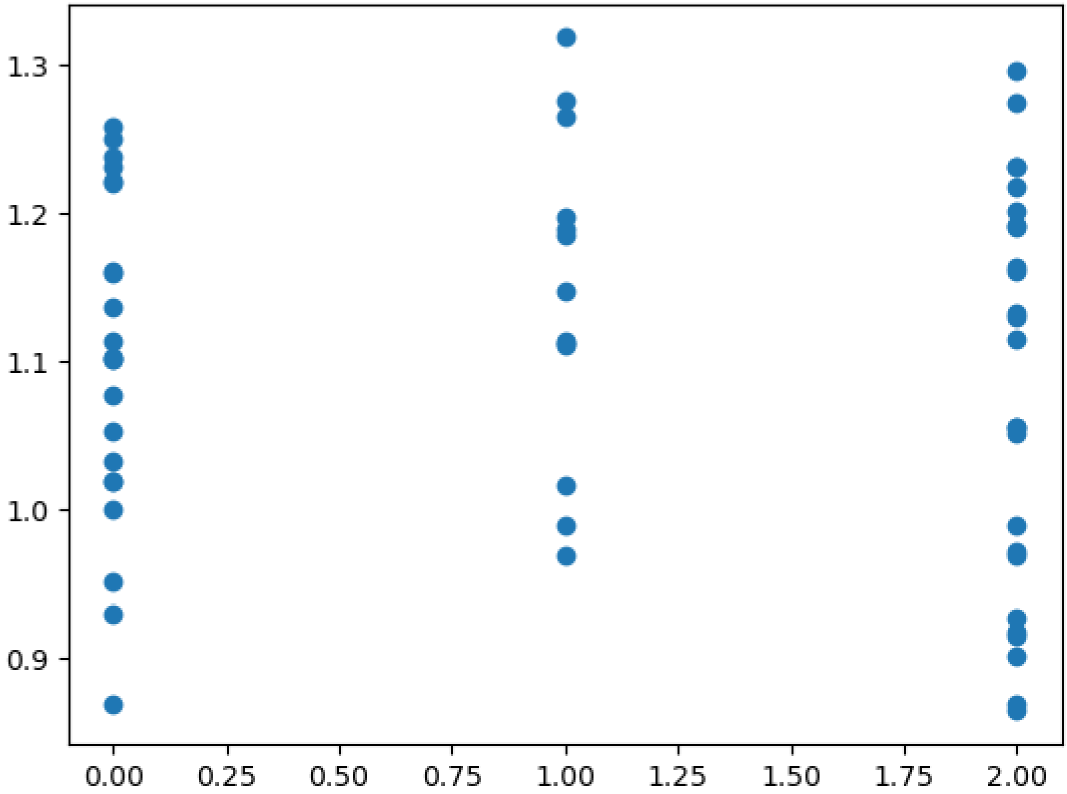
```
-0.0452325646804046
```

In [44]:

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

Out[44]:

```
<matplotlib.collections.PathCollection at 0x1fe589bb410>
```




```
DATA 3
```

In [48]:

```
data3 = pd.read_csv(r"C:\Users\AKHILA\Downloads\7_Uber.csv")
data3
```

Out[48]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pick
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	
...	...	...	...	...	...	
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	

200000 rows × 9 columns

In [49]:

```
data3.describe()
```

Out[49]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dro
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	1
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	
75%	4.155530e+07	12.500000	-73.967154	40.767158	-73.963658	
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	

In [50]:

```
data3.head()
```

Out[50]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_lat
0	24238194	2015-05-07 19:52:06.0000003	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.7
1	27835199	2009-07-17 20:04:56.0000002	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.7
2	44984355	2009-08-24 21:45:00.00000061	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.7
3	25894730	2009-06-26 08:22:21.0000001	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.7
4	17610152	2014-08-28 17:47:00.000000188	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.7

In [51]:

```
data3.tail()
```

Out[51]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	picku
199995	42598914	2012-10-28 10:49:00.00000053	3.0	2012-10-28 10:49:00 UTC	-73.987042	
199996	16382965	2014-03-14 01:09:00.0000008	7.5	2014-03-14 01:09:00 UTC	-73.984722	
199997	27804658	2009-06-29 00:42:00.00000078	30.9	2009-06-29 00:42:00 UTC	-73.986017	
199998	20259894	2015-05-20 14:56:25.0000004	14.5	2015-05-20 14:56:25 UTC	-73.997124	
199999	11951496	2010-05-15 04:08:00.00000076	14.1	2010-05-15 04:08:00 UTC	-73.984395	

In [52]:

```
data3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   Unnamed: 0         200000 non-null  int64
 1   key                200000 non-null  object
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [53]:

```
data3.columns
```

Out[53]:

```
Index(['Unnamed: 0', 'key', 'fare_amount', 'pickup_datetime',
       'pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
       'dropoff_latitude', 'passenger_count'],
      dtype='object')
```

```
RANDOM FOREST
```

In [54]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [55]:

```
df1=pd.read_csv(r"C:\Users\AKHILA\Downloads\C1_Ionosphere.csv")
df1
```

Out[55]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-(
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-(
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	(
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-(
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-(
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-(
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	(
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	(
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-(
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-

350 rows × 35 columns

In [56]:

```
df1.describe()
```

Out[56]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.
count	350.000000	350.0	350.000000	350.000000	350.000000	350.000000	350.000000	350.0
mean	0.891429	0.0	0.640330	0.044667	0.600350	0.116154	0.549284	0.1
std	0.311546	0.0	0.498059	0.442032	0.520431	0.461443	0.493124	0.5
min	0.000000	0.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0
25%	1.000000	0.0	0.471517	-0.065388	0.412555	-0.024868	0.209105	-0.0
50%	1.000000	0.0	0.870795	0.016700	0.808620	0.021170	0.728000	0.0
75%	1.000000	0.0	1.000000	0.194727	1.000000	0.335317	0.970445	0.4
max	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.0

8 rows × 34 columns

In [57]:

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 35 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   1          350 non-null    int64  
 1   0          350 non-null    int64  
 2   0.99539    350 non-null    float64
 3   -0.05889   350 non-null    float64
 4   0.85243    350 non-null    float64
 5   0.02306    350 non-null    float64
 6   0.83398    350 non-null    float64
 7   -0.37708   350 non-null    float64
 8   1.1        350 non-null    float64
 9   0.03760    350 non-null    float64
 10  0.85243.1  350 non-null    float64
 11  -0.17755   350 non-null    float64
 12  0.59755    350 non-null    float64
 13  -0.44945   350 non-null    float64
 14  0.60536    350 non-null    float64
 15  -0.38223   350 non-null    float64
 16  0.84356    350 non-null    float64
 17  -0.38542   350 non-null    float64
 18  0.58212    350 non-null    float64
 19  -0.32192   350 non-null    float64
 20  0.56971    350 non-null    float64
 21  -0.29674   350 non-null    float64
 22  0.36946    350 non-null    float64
 23  -0.47357   350 non-null    float64
 24  0.56811    350 non-null    float64
 25  -0.51171   350 non-null    float64
 26  0.41078    350 non-null    float64
 27  -0.46168   350 non-null    float64
 28  0.21266    350 non-null    float64
 29  -0.34090   350 non-null    float64
 30  0.42267    350 non-null    float64
 31  -0.54487   350 non-null    float64
 32  0.18641    350 non-null    float64
 33  -0.45300   350 non-null    float64
 34  g          350 non-null    object 
dtypes: float64(32), int64(2), object(1)
memory usage: 95.8+ KB
```

In [58]:

```python
g = {"g":{"g":1,"b":2}}
df1 = df1.replace(g)
df1
```

Out[58]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-(
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-(
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-(
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	(
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-(
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-(
...	...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-(
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	(
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	(
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-(
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-

350 rows × 35 columns

In [59]:

```python
df1["g"].value_counts()
```

Out[59]:

```
1    224
2    126
Name: g, dtype: int64
```

In [60]:

```python
x = df1.drop("g",axis=1)
y = df1["g"]
```

In [61]:

```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.40)
```

In [62]:

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

Out[62]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [63]:

```
rf = RandomForestClassifier()
```

In [64]:

```
params = {"max_depth":[1,2,3,4,5],
 "min_samples_leaf":[2,4,6,8,10],
 "n_estimators":[1,3,5,7]
 }
```

In [65]:

```
from sklearn.model_selection import GridSearchCV
gs = GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
gs.fit(x_train,y_train)
```

Out[65]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
    ▸ RandomForestClassifier
```

In [66]:

```
rf_best = gs.best_estimator_
rf_best
```

Out[66]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=4, min_samples_leaf=6, n_estimators=7)
```

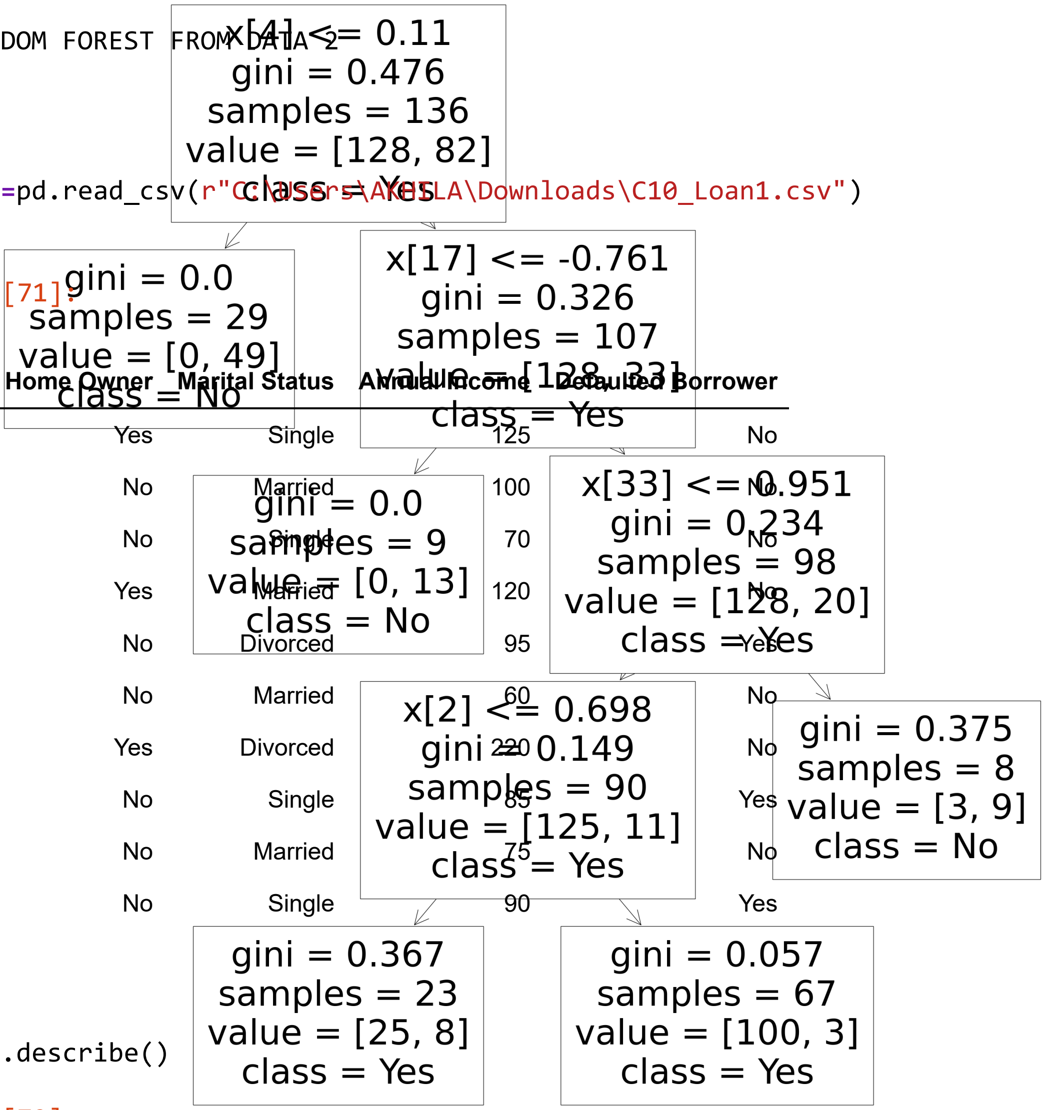
In [67]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(40,40))
plot_tree(rf_best.estimators_[4],feature_names=None,class_names=['Yes','No'])
```

Out[67]:

```
[Text(0.3333333333333333, 0.9, 'x[4] <= 0.11\ngini = 0.476\nsamples = 136
\nvalue = [128, 82]\nclass = Yes'),
 Text(0.16666666666666666, 0.7, 'gini = 0.0\nsamples = 29\nvalue = [0, 49]
\nclass = No'),
 Text(0.5, 0.7, 'x[17] <= -0.761\ngini = 0.326\nsamples = 107\nvalue = [12
8, 33]\nclass = Yes'),
 Text(0.3333333333333333, 0.5, 'gini = 0.0\nsamples = 9\nvalue = [0, 13]\n
class = No'),
 Text(0.6666666666666666, 0.5, 'x[33] <= 0.951\ngini = 0.234\nsamples = 98
\nvalue = [128, 20]\nclass = Yes'),
 Text(0.5, 0.3, 'x[2] <= 0.698\ngini = 0.149\nsamples = 90\nvalue = [125,
11]\nclass = Yes'),
 Text(0.3333333333333333, 0.1, 'gini = 0.367\nsamples = 23\nvalue = [25,
8]\nclass = Yes'),
 Text(0.6666666666666666, 0.1, 'gini = 0.057\nsamples = 67\nvalue = [100,
3]\nclass = Yes'),
 Text(0.8333333333333334, 0.3, 'gini = 0.375\nsamples = 8\nvalue = [3, 9]
\nclass = No')]
```

```
RANDOM FOREST FROM DATA 2
```




```
df2=pd.read_csv(r"C:\Users\AKHILA\Downloads\C10_Loan1.csv")
df2
```

Out[71]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	Yes	Single	125	No
1	No	Married	100	No
2	No	Single	70	No
3	Yes	Married	120	No
4	No	Divorced	95	Yes
5	No	Married	60	No
6	Yes	Divorced	220	No
7	No	Single	85	Yes
8	No	Married	75	No
9	No	Single	90	Yes

```
df2.describe()
```

Out[72]:

	Annual Income
count	10.000000
mean	104.000000
std	45.631373
min	60.000000
25%	77.500000
50%	92.500000
75%	115.000000
max	220.000000

In [73]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Home Owner          10 non-null     object
 1   Marital Status      10 non-null     object
 2   Annual Income       10 non-null     int64 
 3   Defaulted Borrower  10 non-null     object
dtypes: int64(1), object(3)
memory usage: 452.0+ bytes
```

In [74]:

```
Home_Owner = {"Home Owner":{"Yes":1,"No":2}}
df2 = df2.replace(Home_Owner)
Defaulted_Borrower = {"Defaulted Borrower":{"Yes":1,"No":2}}
df2 = df2.replace(Defaulted_Borrower)
Marital_Status = {"Marital Status":{"Divorced":0,"Single":1,"Married":2}}
df2 = df2.replace(Marital_Status)
df2
```

Out[74]:

	Home Owner	Marital Status	Annual Income	Defaulted Borrower
0	1	1	125	2
1	2	2	100	2
2	2	1	70	2
3	1	2	120	2
4	2	0	95	1
5	2	2	60	2
6	1	0	220	2
7	2	1	85	1
8	2	2	75	2
9	2	1	90	1

In [75]:

```
df2["Home Owner"].value_counts()
```

Out[75]:

```
2    7
1    3
Name: Home Owner, dtype: int64
```

In [76]:

```
df2["Defaulted Borrower"].value_counts()
```

Out[76]:

```
2    7
1    3
Name: Defaulted Borrower, dtype: int64
```

In [77]:

```
x = df2.drop("Marital Status",axis=1)
y = df2["Marital Status"]
```

In [78]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.40)
```

In [79]:

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

Out[79]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [80]:

```
rf = RandomForestClassifier()
```

In [81]:

```
params = {"max_depth":[1,2,3,4,5],
 "min_samples_leaf":[2,4,6,8,10],
 "n_estimators":[1,3,5,7]
 }
```

In [82]:

```
from sklearn.model_selection import GridSearchCV
gs = GridSearchCV(estimator=rf,param_grid=params,cv=2,scoring='accuracy')
gs.fit(x_train,y_train)
```

Out[82]:

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

In [83]:

```
rf_best = gs.best_estimator_
rf_best
```

Out[83]:

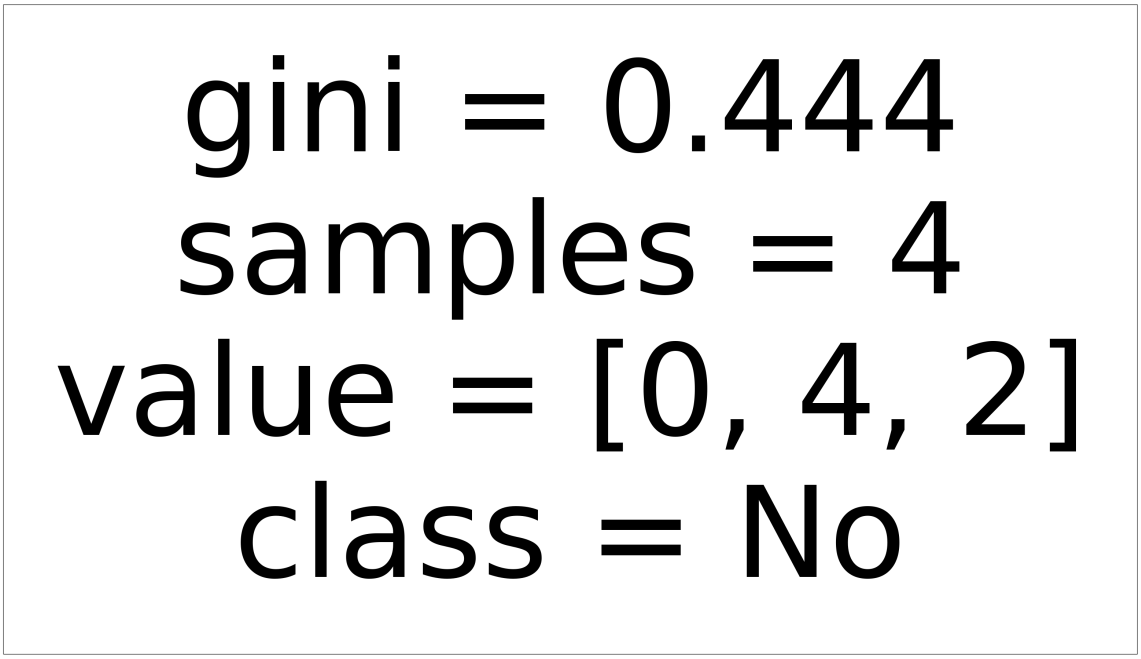
```
▾                     RandomForestClassifier
RandomForestClassifier(max_depth=1, min_samples_leaf=2, n_estimators=5)
```

In [84]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(40,40))
plot_tree(rf_best.estimators_[4],feature_names=None,class_names=['Yes','No'])
```

Out[84]:

```
[Text(0.5, 0.5, 'gini = 0.444\nsamples = 4\nvalue = [0, 4, 2]\nclass = N
o')]
```



In [ ]:

```

```