In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [2]:
```
df=pd.read_csv("ionosphere.csv")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.511
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.265
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.402
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.906
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.651
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.015
...	...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.042
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.013
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.031
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.020
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.151

350 rows × 35 columns

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

Out[3]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535

5 rows × 35 columns

Data Cleaning and Data Preprocessing

In [4]:
```
df.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 [5]:
```
df.describe()
```

Out[5]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.3770
count	350.000000	350.0	350.000000	350.000000	350.000000	350.000000	350.000000	350.00000
mean	0.891429	0.0	0.640330	0.044667	0.600350	0.116154	0.549284	0.12077
std	0.311546	0.0	0.498059	0.442032	0.520431	0.461443	0.493124	0.52081
min	0.000000	0.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.00000
25%	1.000000	0.0	0.471517	-0.065388	0.412555	-0.024868	0.209105	-0.05348
50%	1.000000	0.0	0.870795	0.016700	0.808620	0.021170	0.728000	0.01508
75%	1.000000	0.0	1.000000	0.194727	1.000000	0.335317	0.970445	0.45157
max	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

8 rows × 34 columns

In [6]:
```
df.columns
```

Out[6]:
```
Index(['1', '0', '0.99539', '-0.05889', '0.85243', '0.02306', '0.83398',
       '-0.37708', '1.1', '0.03760', '0.85243.1', '-0.17755', '0.59755',
       '-0.44945', '0.60536', '-0.38223', '0.84356', '-0.38542', '0.58212',
       '-0.32192', '0.56971', '-0.29674', '0.36946', '-0.47357', '0.56811',
       '-0.51171', '0.41078', '-0.46168', '0.21266', '-0.34090', '0.42267',
       '-0.54487', '0.18641', '-0.45300', 'g'],
      dtype='object')
```

In [7]:
```
feature_matrix = df.iloc[:,0:34]
target_vector = df.iloc[:,-1]
```

In [8]:
```
fs = StandardScaler().fit_transform(feature_matrix)
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[8]:
```
LogisticRegression()
```

In [9]:
```
observation=[[1.0,0.0,1.0,-0.18829,0.93035,
 -0.36156,
 -0.10868,
 -0.93597,
 1.0,
 -0.04549,
 0.50874,
 -0.67743,
 0.34432,
 -0.69707,
 -0.51685,
 -0.97515,
 0.05499,
 -0.62237,
 0.33109,
 -1.0,
 -0.13151,
 -0.453,
 -0.18056,
 -0.35734,
 -0.20332,
 -0.26569,
 -0.20468,
 -0.18401,
 -0.1904,
 -0.11593,
 -0.16626,
 -0.06288,
 -0.13738,
 -0.02447]]
prediction = logr.predict(observation)
print(prediction)
```

```
['g']
```

In [10]:
```
logr.classes_
```

Out[10]:
```
array(['b', 'g'], dtype=object)
```

In [11]:
```
logr.predict_proba(observation)
```

Out[11]:
```
array([[0.07006552, 0.92993448]])
```

In [12]:
```
df['g'].value_counts()
```

Out[12]:
```
g    224
b    126
Name: g, dtype: int64
```

In [13]:
```
x=df.drop('g', axis=1)
y=df['g']
```

In [14]:
```
g1={"g":{"g":1, "b":2}}
df=df.replace(g1)
df
```

Out[14]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.511
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.265
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.402
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.906
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.651
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.015
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.042
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.013
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.031
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.020
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.151

350 rows × 35 columns

In [15]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.70)
```

In [16]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[16]:
```
RandomForestClassifier()
```

In [20]:
```
parameters = {'max_depth':[1,2,3,4,5],'min_samples_leaf':[5,10,15,20,25],'n_est
```

In [21]:
```
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(estimator=rfc,param_grid= parameters,cv=2,scoring =
grid_search.fit(x_train,y_train)
```

Out[21]:
```
GridSearchCV(cv=2, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [1, 2, 3, 4, 5],
                         'min_samples_leaf': [5, 10, 15, 20, 25],
                         'n_estimators': [10, 20, 30, 40, 50]},
             scoring='accuracy')
```

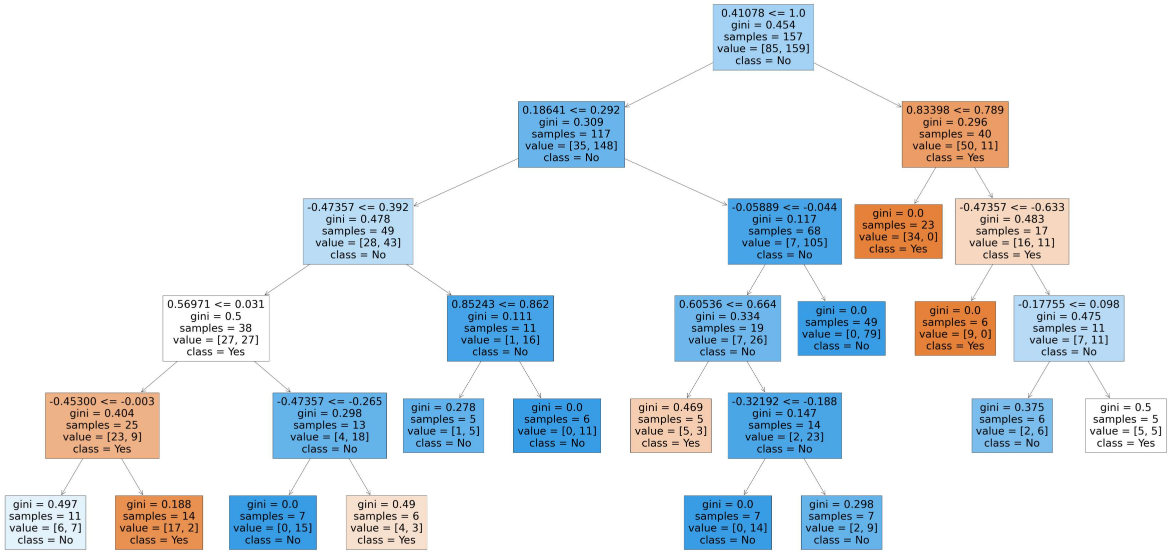
In [22]:
```
grid_search.best_score_
```

Out[22]:
```
0.9221311475409837
```

In [25]:
```
rfc_best = grid_search.best_estimator_
```

In [27]:
```
from sklearn.tree import plot_tree
plt.figure(figsize = (80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names = ['Yes',
```

Out[27]:
```
[Text(2896.2857142857147, 1993.2, '0.41078 <= 1.0\ngini = 0.454\nsamples = 15
7\nvalue = [85, 159]\nclass = No'),
 Text(2178.857142857143, 1630.8000000000002, '0.18641 <= 0.292\ngini = 0.309
\nsamples = 117\nvalue = [35, 148]\nclass = No'),
 Text(1381.7142857142858, 1268.4, '-0.47357 <= 0.392\ngini = 0.478\nsamples =
49\nvalue = [28, 43]\nclass = No'),
 Text(850.2857142857143, 906.0, '0.56971 <= 0.031\ngini = 0.5\nsamples = 38\n
value = [27, 27]\nclass = Yes'),
 Text(425.14285714285717, 543.5999999999999, '-0.45300 <= -0.003\ngini = 0.40
4\nsamples = 25\nvalue = [23, 9]\nclass = Yes'),
 Text(212.57142857142858, 181.19999999999982, 'gini = 0.497\nsamples = 11\nva
lue = [6, 7]\nclass = No'),
 Text(637.7142857142858, 181.19999999999982, 'gini = 0.188\nsamples = 14\nval
ue = [17, 2]\nclass = Yes'),
 Text(1275.4285714285716, 543.5999999999999, '-0.47357 <= -0.265\ngini = 0.29
8\nsamples = 13\nvalue = [4, 18]\nclass = No'),
 Text(1062.857142857143, 181.19999999999982, 'gini = 0.0\nsamples = 7\nvalue
= [0, 15]\nclass = No'),
 Text(1488.0, 181.19999999999982, 'gini = 0.49\nsamples = 6\nvalue = [4, 3]\n
class = Yes'),
 Text(1913.1428571428573, 906.0, '0.85243 <= 0.862\ngini = 0.111\nsamples = 1
1\nvalue = [1, 16]\nclass = No'),
 Text(1700.5714285714287, 543.5999999999999, 'gini = 0.278\nsamples = 5\nvalu
e = [1, 5]\nclass = No'),
 Text(2125.714285714286, 543.5999999999999, 'gini = 0.0\nsamples = 6\nvalue =
[0, 11]\nclass = No'),
 Text(2976.0, 1268.4, '-0.05889 <= -0.044\ngini = 0.117\nsamples = 68\nvalue
= [7, 105]\nclass = No'),
 Text(2763.4285714285716, 906.0, '0.60536 <= 0.664\ngini = 0.334\nsamples = 1
9\nvalue = [7, 26]\nclass = No'),
 Text(2550.857142857143, 543.5999999999999, 'gini = 0.469\nsamples = 5\nvalue
= [5, 3]\nclass = Yes'),
 Text(2976.0, 543.5999999999999, '-0.32192 <= -0.188\ngini = 0.147\nsamples =
14\nvalue = [2, 23]\nclass = No'),
 Text(2763.4285714285716, 181.19999999999982, 'gini = 0.0\nsamples = 7\nvalue
= [0, 14]\nclass = No'),
 Text(3188.571428571429, 181.19999999999982, 'gini = 0.298\nsamples = 7\nvalu
e = [2, 9]\nclass = No'),
 Text(3188.571428571429, 906.0, 'gini = 0.0\nsamples = 49\nvalue = [0, 79]\nc
lass = No'),
 Text(3613.714285714286, 1630.8000000000002, '0.83398 <= 0.789\ngini = 0.296
\nsamples = 40\nvalue = [50, 11]\nclass = Yes'),
 Text(3401.1428571428573, 1268.4, 'gini = 0.0\nsamples = 23\nvalue = [34, 0]
\nclass = Yes'),
 Text(3826.2857142857147, 1268.4, '-0.47357 <= -0.633\ngini = 0.483\nsamples
= 17\nvalue = [16, 11]\nclass = Yes'),
 Text(3613.714285714286, 906.0, 'gini = 0.0\nsamples = 6\nvalue = [9, 0]\ncla
ss = Yes'),
 Text(4038.857142857143, 906.0, '-0.17755 <= 0.098\ngini = 0.475\nsamples = 1
1\nvalue = [7, 11]\nclass = No'),
 Text(3826.2857142857147, 543.5999999999999, 'gini = 0.375\nsamples = 6\nvalu
e = [2, 6]\nclass = No'),
 Text(4251.428571428572, 543.5999999999999, 'gini = 0.5\nsamples = 5\nvalue =
[5, 5]\nclass = Yes')]
```


0.41078 <= 1.0
gini = 0.454
samples = 157
value = [85, 159]
class = No
0.18641 <= 0.292
gini = 0.309
samples = 117
value = [35, 148]
class = No
0.83398 <= 0.789
gini = 0.296
samples = 40
value = [50, 11]
class = Yes
-0.47357 <= 0.392
gini = 0.478
samples = 49
value = [28, 43]
class = No
-0.05889 <= -0.044
gini = 0.117
samples = 68
value = [7, 105]
class = No
gini = 0.0
samples = 23
value = [34, 0]
class = Yes
-0.47357 <= -0.633
gini = 0.483
samples = 17
value = [16, 11]
class = Yes
0.56971 <= 0.031
gini = 0.5
samples = 38
value = [27, 27]
class = Yes
0.85243 <= 0.862
gini = 0.111
samples = 11
value = [1, 16]
class = No
0.60536 <= 0.664
gini = 0.334
samples = 19
value = [7, 26]
class = No
gini = 0.0
samples = 49
value = [0, 79]
class = No
gini = 0.0
samples = 6
value = [9, 0]
class = Yes
-0.17755 <= 0.098
gini = 0.475
samples = 11
value = [7, 11]
class = No
-0.45300 <= -0.003
gini = 0.404
samples = 25
value = [23, 9]
class = Yes
-0.47357 <= -0.265
gini = 0.298
samples = 13
value = [4, 18]
class = No
gini = 0.278
samples = 5
value = [1, 5]
class = No
gini = 0.0
samples = 6
value = [0, 11]
class = No
gini = 0.469
samples = 5
value = [5, 3]
class = Yes
-0.32192 <= -0.188
gini = 0.147
samples = 14
value = [2, 23]
class = No
gini = 0.375
samples = 6
value = [2, 6]
class = No
gini = 0.5
samples = 5
value = [5, 5]
class = Yes
gini = 0.497
samples = 11
value = [6, 7]
class = No
gini = 0.188
samples = 14
value = [17, 2]
class = Yes
gini = 0.0
samples = 7
value = [0, 15]
class = No
gini = 0.49
samples = 6
value = [4, 3]
class = Yes
gini = 0.0
samples = 7
value = [0, 14]
class = No
gini = 0.298
samples = 7
value = [2, 9]
class = No