In [23]:
```
import pandas as pd
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
```

In [7]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

In [3]:
```
df=pd.read_csv("1_ionosphere.csv")
df
```

Out[3]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0

350 rows × 35 columns

In [4]:
```
x=df.iloc[:,:10]
y=df.iloc[:,-1]
f=StandardScaler().fit_transform(x)
lo=LogisticRegression()
lo.fit(f,y)
```

Out[4]:
```
LogisticRegression()
```

In [5]:
```
val=[[6,45,234,534,6,3456,345,4,53,45]]
lo.predict(val)
```

Out[5]:
```
array(['g'], dtype=object)
```

In [6]:
```
lo.score(f,y)
```

Out[6]:
```
0.8885714285714286
```

In [10]:
```
df["g"].value_counts()
```

Out[10]:
```
g    224
b    126
Name: g, dtype: int64
```

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	...	-0.51171	0
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0
...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0

350 rows × 35 columns

In [17]:
```
x1=df.drop(["g"],axis=1)
y1=df["g"]
x_train,x_test,y_train,y_test=train_test_split(x1,y1,test_size=0.3)
```

In [18]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[18]:
```
RandomForestClassifier()
```

In [19]:
```
parameter={'max_depth':[1,2,3,4,5],
           "min_samples_leaf":[5,10,15,20,25],
           "n_estimators":[10,20,30,40,50]}
```

In [20]:
```
from sklearn.model_selection import GridSearchCV

grid_search = GridSearchCV(estimator=rfc,param_grid=parameter,cv=2,scoring="accuracy
grid_search.fit(x_train,y_train)
```

Out[20]:
```
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 [21]:
```
grid_search.best_score_
```

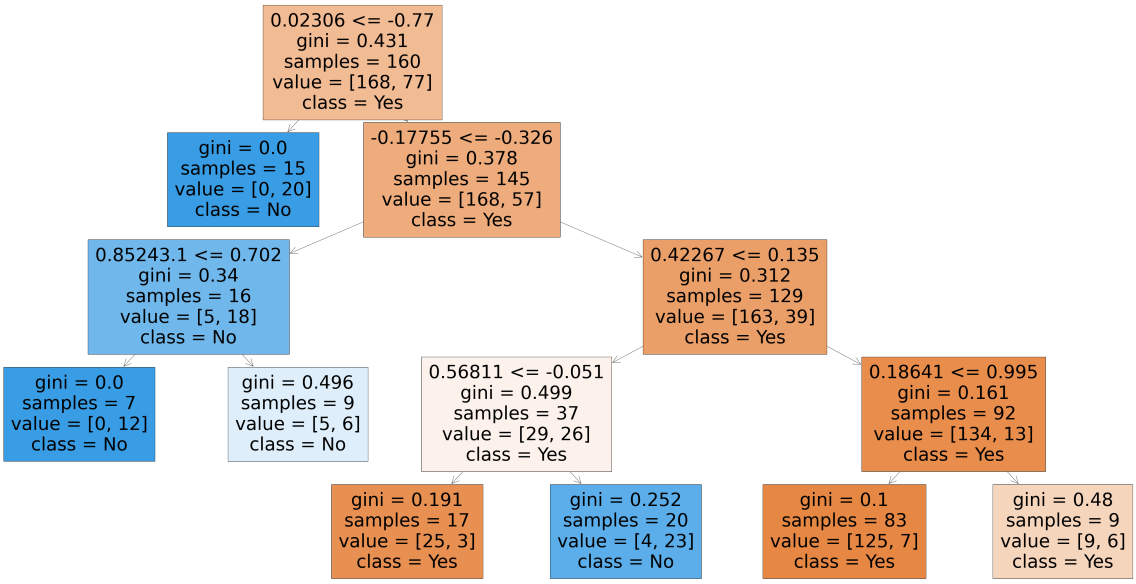
Out[21]:
```
0.9265293882447021
```

In [26]:
```
rfc_best=grid_search.best_estimator_
```

In [29]:
```
from sklearn.tree import plot_tree

plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x1.columns,class_names=['Yes','No']
```

Out[29]:
```
[Text(1420.3636363636363, 1956.96, '0.02306 <= -0.77\ngini = 0.431\nsamples = 160
\nvalue = [168, 77]\nclass = Yes'),
 Text(1014.5454545454545, 1522.0800000000002, 'gini = 0.0\nsamples = 15\nvalue =
[0, 20]\nclass = No'),
 Text(1826.1818181818182, 1522.0800000000002, '-0.17755 <= -0.326\ngini = 0.378\ns
amples = 145\nvalue = [168, 57]\nclass = Yes'),
 Text(811.6363636363636, 1087.2, '0.85243.1 <= 0.702\ngini = 0.34\nsamples = 16\nv
alue = [5, 18]\nclass = No'),
 Text(405.8181818181818, 652.3200000000002, 'gini = 0.0\nsamples = 7\nvalue = [0,
12]\nclass = No'),
 Text(1217.4545454545455, 652.3200000000002, 'gini = 0.496\nsamples = 9\nvalue =
[5, 6]\nclass = No'),
 Text(2840.7272727272725, 1087.2, '0.42267 <= 0.135\ngini = 0.312\nsamples = 129\n
value = [163, 39]\nclass = Yes'),
 Text(2029.090909090909, 652.3200000000002, '0.56811 <= -0.051\ngini = 0.499\nsamp
les = 37\nvalue = [29, 26]\nclass = Yes'),
 Text(1623.2727272727273, 217.44000000000005, 'gini = 0.191\nsamples = 17\nvalue =
[25, 3]\nclass = Yes'),
 Text(2434.909090909091, 217.44000000000005, 'gini = 0.252\nsamples = 20\nvalue =
[4, 23]\nclass = No'),
 Text(3652.3636363636365, 652.3200000000002, '0.18641 <= 0.995\ngini = 0.161\nsamp
les = 92\nvalue = [134, 13]\nclass = Yes'),
 Text(3246.5454545454545, 217.44000000000005, 'gini = 0.1\nsamples = 83\nvalue =
[125, 7]\nclass = Yes'),
 Text(4058.181818181818, 217.44000000000005, 'gini = 0.48\nsamples = 9\nvalue =
[9, 6]\nclass = Yes')]
```



In [ ]: