```
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 [4]:

```
df = pd.read_csv(r"C:\Users\user\Desktop\C1_ionosphere.csv")
df
```

Out[4]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.46168	0.21266	-0.34090	0.42267	-0.54487	0.18641	-0.45300	g
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.19040	-0.11593	-0.16626	-0.06288	-0.13738	-0.02447	b
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	0.43100	-0.17365	0.60436	-0.24180	0.56045	-0.38238	g
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	1.00000	-0.20099	0.25682	1.00000	-0.32382	1.00000	b
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	0.02431	-0.62197	-0.05707	-0.59573	-0.04608	-0.65697	g
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.09223	-0.07859	0.00732	0.00000	0.00000	-0.00039	0.12011	b
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03076	...	-0.51171	0.41078	-0.46168	0.21266	-0.34099	0.42267	-0.54487	0.18641	-0.4533	g
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00123	1.00000	0.12815	0.86660	-0.10714	0.90546	-0.04307	g
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04925	0.93159	0.08168	0.94066	-0.00035	0.91483	0.04712	g
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02542	0.92120	0.02242	0.92459	0.00442	0.92697	-0.00577	g
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07760	0.82983	-0.17238	0.96022	-0.03757	0.87403	-0.16243	g
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04822	0.78207	-0.00703	0.75747	-0.06678	0.85764	-0.06151	g

350 rows × 35 columns

In [9]:

```
df.head()
```

Out[9]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03076	...	-0.51171	0.41078	-0.46168	0.21266	-0.34099	0.42267	-0.54487	0.18641	-0.4533	g
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.19040	-0.11593	-0.16626	-0.06288	-0.13738	-0.02447	b

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.0376	...	-0.51171	0.41078	-0.46168	0.21266	-0.34090	0.42267	-0.54487	0.18641	-0.4533	g
1	1	0	1.000000	-0.033655	1.000000	0.000485	1.000000	-0.120662	0.889665	0.011198	...	-0.40220	0.589784	-0.221455	0.431100	-0.173655	0.606436	-0.241800	0.560445	-0.382238	g
2	1	0	1.000000	-0.451611	1.000000	1.000000	0.712166	-1.000000	0.000000	0.000000	...	0.90695	0.516133	1.000000	1.000000	-0.200999	0.256682	1.000000	-0.323822	1.000000	b
3	1	0	1.000000	-0.024401	0.941400	0.065311	0.921066	-0.232555	0.771512	-0.163999	...	-0.65158	0.132900	-0.532066	0.024331	-0.621977	-0.057077	-0.595733	-0.046088	-0.656997	g
4	1	0	0.02337	-0.000592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.09223	-0.07859	0.00732	0.00000	0.00000	-0.00039	0.12011	b

5 rows × 35 columns

In [10]:

```
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   0.0376      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  -0.3409     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  -0.453      350 non-null    float64
 34  g           350 non-null    object
dtypes: float64(32), int64(2), object(1)
memory usage: 95.8+ KB
```

In [11]:

```
df.describe()
```

Out[11]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.0376	...	0.56811	-0.51171	0.41078	-0.46168	0.21266	-0.34091	0.42267	-0.54487	0.18641	-0.453
count	350.000000	350.0	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	...	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000	350.000000
mean	0.891429	0.0	0.640330	0.044667	0.600350	0.116154	0.549284	0.120779	0.510453	0.181756	...	0.395643	-0.069928	0.542015	-0.068417	0.378919	-0.027013	0.352313	-0.002248	0.349829	0.015816
std	0.311546	0.0	0.498059	0.442032	0.520431	0.461443	0.493124	0.520816	0.507117	0.484482	...	0.579206	0.508675	0.516896	0.550411	0.576642	0.508425	0.572289	0.513491	0.523339	0.468338
min	0.000000	0.0	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	...	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	1.000000	0.0	0.471517	-0.061	0.412555	-0.02	0.209105	-0.05	0.086785	-0.04	...	0.000000	-0.32	0.283612	-0.42	0.000000	-0.23	0.000000	-0.23	0.000000	-0.16

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.0376	...	0.56811	-0.51171	0.41078	-0.46168	0.21266	-0.3409	0.42267	-0.54487	0.18641	-0.453
				5388		4868		3483		9003			3745		8992		4935		9347		1013
50%	1.000000	0.0	0.87079	0.016700	0.80862	0.021170	0.72800	0.015085	0.68243	0.017550	...	0.54917	-0.014915	0.70853	-0.017685	0.49921	0.000000	0.44687	0.000000	0.41315	0.000000
75%	1.000000	0.0	1.000000	0.194727	1.000000	0.335317	0.97045	0.451572	0.95055	0.536192	...	0.90716	0.157922	0.99997	0.154862	0.88457	0.154218	0.85949	0.200935	0.81678	0.172105
max	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 34 columns

In [12]:

```
df.columns
```

Out[12]:

```
Index(['1', '0', '0.99539', '-0.05889', '0.85243', '0.02306', '0.83398',
       '-0.37708', '1.1', '0.0376', '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', '-0.3409', '0.42267',
       '-0.54487', '0.18641', '-0.453', 'g'],
      dtype='object')
```

In [13]:

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

In [14]:

```
fs = StandardScaler().fit_transform(feature_matrix)
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

Out[14]:

```
LogisticRegression()
```

In [15]:

```
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]]
```

In [16]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
['g']
```

In [17]:

```
logr.classes_
```

Out[17]:

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

In [18]:

```
logr.predict_proba(observation)
```

Out[18]:

```
array([[0.07006552, 0.92993448]])
```

In [19]:

```
df['g'].value_counts()
```

Out[19]:

```
g    224
b    126
Name: g, dtype: int64
```

In [20]:

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

In [21]:

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

Out[21]:

	1	**0**	**0.99539**	**-0.05889**	**0.85243**	**0.02306**	**0.83398**	**-0.37708**	**1.1**	**0.03776**	**...**	**-0.51171**	**0.41078**	**-0.46168**	**0.21266**	**-0.34**	**0.42267**	**-0.54487**	**0.18641**	**-0.453**	**g**
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.19040	-0.11593	-0.16626	-0.06288	-0.13738	-0.02447	2
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	0.43100	-0.17365	0.60436	-0.24180	0.56045	-0.38238	1
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	1.00000	-0.20099	0.25682	1.00000	-0.32382	1.00000	2
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	0.02431	-0.62197	-0.05707	-0.59573	-0.04608	-0.65697	1
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.09223	-0.07859	0.00732	0.00000	0.00000	-0.00039	0.12011	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00123	1.00000	0.12815	0.86660	-0.10714	0.90546	-0.04307	1
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04925	0.93159	0.08168	0.94066	-0.00035	0.91483	0.04712	1

	1	**0**	**0.99539**	**-0.05889**	**0.85243**	**0.02306**	**0.83398**	**-0.37708**	**1.1**	**0.03776**	**...**	**-0.51171**	**0.41078**	**-0.46168**	**0.21266**	**-0.34**	**0.42267**	**-0.54487**	**0.18641**	**-0.453**	**g**
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02542	0.92120	0.02242	0.92459	0.00442	0.92697	-0.00577	1
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07760	0.82983	-0.17238	0.96022	-0.03757	0.87403	-0.16243	1
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04822	0.78207	-0.00703	0.75747	-0.06678	0.85764	-0.06151	1

350 rows × 35 columns

In [22]:

```
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 [23]:

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

Out[23]:

```
RandomForestClassifier()
```

In [24]:

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

In [25]:

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

In [28]:

```
 from sklearn.model_selection import GridSearchCV
grid_search
=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[28]:

```
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 [29]:

```
grid_search.best_score_
```

Out[29]:

```
0.9262295081967213
```

In [30]:

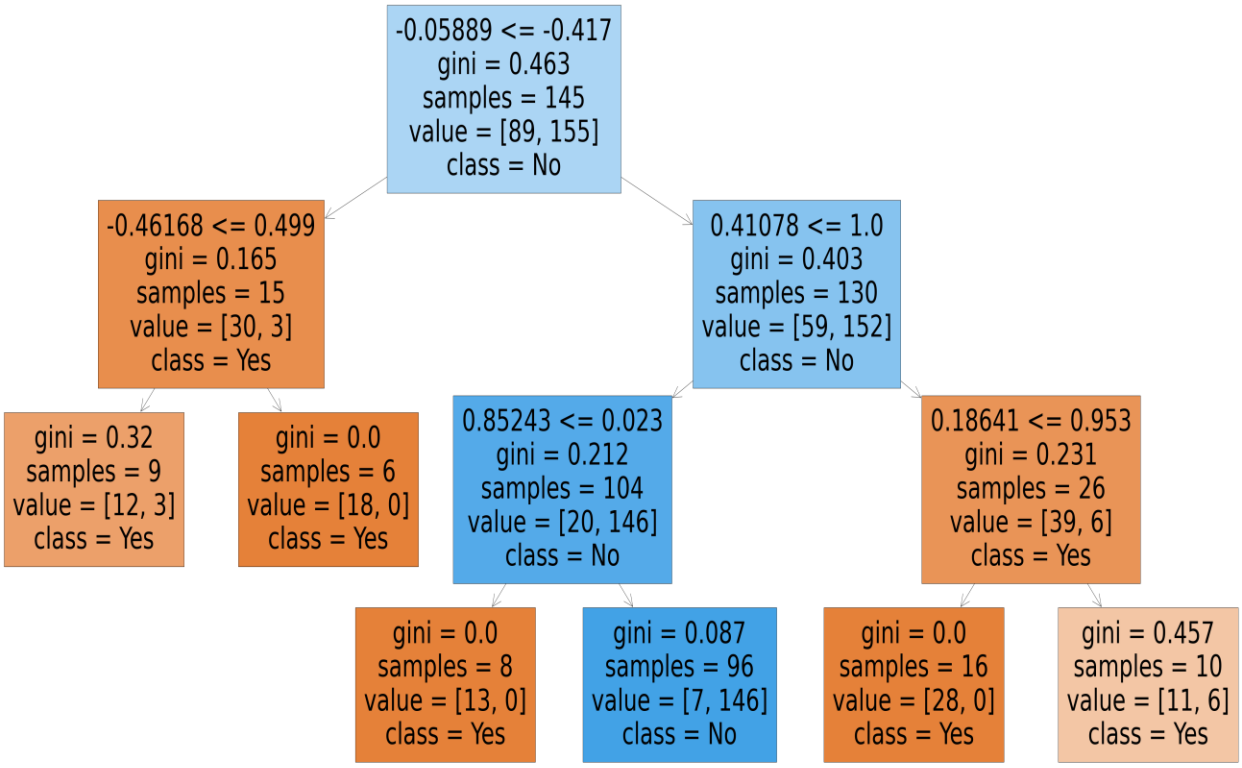
```
rfc_best = grid_search.best_estimator_
```

In [31]:

```
from sklearn.tree import plot_tree
plt.figure(figsize=(89,40))
plot_tree(rfc_best.estimators_[5], feature_names=x.columns,
class_names=['Yes','No'], filled=True)
```

Out[31]:

```
[Text(2031.6272727272726, 1902.6000000000001, '-0.05889 <= -0.417\ngini = 0
.463\nsamples = 145\nvalue = [89, 155]\nclass = No'),
 Text(902.9454545454545, 1359.0, '-0.46168 <= 0.499\ngini = 0.165\nsamples
= 15\nvalue = [30, 3]\nclass = Yes'),
 Text(451.47272727272724, 815.4000000000001, 'gini = 0.32\nsamples = 9\nval
ue = [12, 3]\nclass = Yes'),
 Text(1354.4181818181817, 815.4000000000001, 'gini = 0.0\nsamples = 6\nvalu
e = [18, 0]\nclass = Yes'),
 Text(3160.3090909090906, 1359.0, '0.41078 <= 1.0\ngini = 0.403\nsamples =
130\nvalue = [59, 152]\nclass = No'),
 Text(2257.363636363636, 815.4000000000001, '0.85243 <= 0.023\ngini = 0.212
\nsamples = 104\nvalue = [20, 146]\nclass = No'),
 Text(1805.890909090909, 271.79999999999995, 'gini = 0.0\nsamples = 8\nvalu
e = [13, 0]\nclass = Yes'),
 Text(2708.8363636363633, 271.79999999999995, 'gini = 0.087\nsamples = 96\n
value = [7, 146]\nclass = No'),
 Text(4063.254545454545, 815.4000000000001, '0.18641 <= 0.953\ngini = 0.231
\nsamples = 26\nvalue = [39, 6]\nclass = Yes'),
 Text(3611.781818181818, 271.79999999999995, 'gini = 0.0\nsamples = 16\nval
ue = [28, 0]\nclass = Yes'),
 Text(4514.727272727272, 271.79999999999995, 'gini = 0.457\nsamples = 10\nv
alue = [11, 6]\nclass = Yes')]
```


-0.05889 <= -0.417
gini = 0.463
samples = 145
value = [89, 155]
class = No
-0.46168 <= 0.499
gini = 0.165
samples = 15
value = [30, 3]
class = Yes
0.41078 <= 1.0
gini = 0.403
samples = 130
value = [59, 152]
class = No
gini = 0.32
samples = 9
value = [12, 3]
class = Yes
gini = 0.0
samples = 6
value = [18, 0]
class = Yes
0.85243 <= 0.023
gini = 0.212
samples = 104
value = [20, 146]
class = No
0.18641 <= 0.953
gini = 0.231
samples = 26
value = [39, 6]
class = Yes
gini = 0.0
samples = 8
value = [13, 0]
class = Yes
gini = 0.087
samples = 96
value = [7, 146]
class = No
gini = 0.0
samples = 16
value = [28, 0]
class = Yes
gini = 0.457
samples = 10
value = [11, 6]
class = Yes