mk 2/08/23

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [54]:
```
from sklearn.linear_model import LogisticRegression
df=pd.read_csv(r"C:\USERS\user\Downloads\C1_ionosphere.csv")
df
```

Out[54]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.461
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.184
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.221
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.532
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.092
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.001
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.049
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.025
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.077
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.048

350 rows × 35 columns

In [55]:
```
feature_matrix=df.iloc[:,0:34]
target_vector=df.iloc[:,-1]
```

In [56]:
```
feature_matrix.shape
target_vector.shape
```

Out[56]:
```
(350,)
```

In [57]:
```
from sklearn.preprocessing import StandardScaler
fs=StandardScaler().fit_transform(feature_matrix)
```

In [58]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[58]:
```
LogisticRegression()
```

In [59]:
```
observation=[[1.4,2.3,-5.0,11,12,13,14,15,16,17,1,2,3,4,5,6,7,8,9,10,21,22,23,24,25,26,27,28,2
```

In [60]:
```
prediction=logr.predict(observation)
prediction
```

Out[60]:
```
array(['g'], dtype=object)
```

In [61]:
```
logr.classes_
logr.predict_proba(observation)[0][0]
```

Out[61]:
```
0.0
```

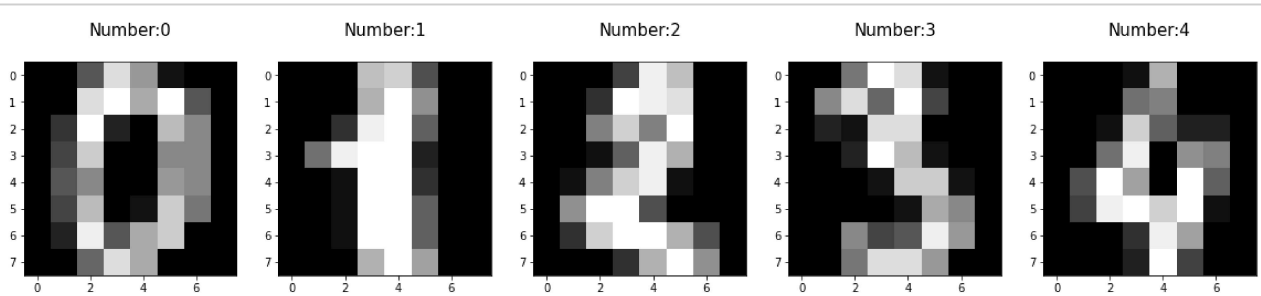
In [62]:
```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [63]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [64]:
```
digits=load_digits()
digits
```

Out[64]:
```
{'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  2., ..., 12.,  0.,  0.],
        [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
 'target': array([0, 1, 2, ..., 8, 9, 8]),
 'frame': None,
 'feature_names': ['pixel_0_0',
  'pixel_0_1',
  'pixel_0_2',
  'pixel_0_3',
  'pixel_0_4',
  'pixel_0_5',
  'pixel_0_6',
  'pixel_0_7',
  'pixel_1_0',
  'pixel_1_1',
```

In [65]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=15)
```



In [66]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [67]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

```
(1257, 64)
(540, 64)
(1257,)
(540,)
```

Out[67]:
```
LogisticRegression(max_iter=10000)
```

In [68]:
```
print(logre.predict(x_test))
```

```
[3 7 0 5 9 6 1 4 5 0 9 1 6 5 5 0 5 6 4 5 8 2 7 5 1 9 4 3 1 0 3 6 7 5 7 7 0
 3 5 1 9 1 8 0 6 1 6 4 7 4 6 7 4 2 8 0 1 9 1 5 8 9 1 3 5 2 0 7 9 7 0 7 2 1
 7 3 5 7 2 9 0 9 4 2 4 9 9 2 6 6 7 3 3 3 5 6 6 1 2 1 7 5 6 4 5 5 9 8 9 5 6
 8 7 2 3 7 1 5 7 4 3 2 8 6 6 9 2 5 5 8 6 3 4 6 3 4 9 5 2 9 4 8 4 2 1 1 6 6
 0 7 6 5 0 9 4 2 7 0 1 3 0 5 7 1 9 3 0 0 3 5 1 1 1 4 1 8 2 9 1 7 9 0 6 7 4
 6 4 7 0 8 6 9 8 6 3 4 3 3 7 5 5 9 9 3 7 6 2 0 6 1 1 9 4 2 2 0 2 9 7 1 8 3
 2 6 1 8 1 4 6 5 6 1 9 9 5 8 4 2 1 6 8 5 6 3 6 5 2 4 6 1 4 7 3 5 8 5 8 6 7
 3 7 5 5 5 4 7 7 4 1 4 3 5 0 8 0 3 0 2 5 9 8 6 1 5 9 2 4 7 5 1 1 5 3 5 2 5
 2 1 0 1 8 6 3 2 3 6 5 8 7 4 0 4 4 5 2 1 0 4 6 3 3 7 9 0 4 3 1 4 4 3 5 3 8
 6 9 8 2 9 5 1 3 6 1 0 0 6 9 4 7 9 6 9 3 7 9 8 3 7 3 5 0 7 8 8 3 7 9 4 3 6
 5 8 6 7 1 2 7 3 9 4 6 0 6 4 7 8 7 9 0 3 8 0 4 2 0 7 3 3 4 2 7 3 6 1 7 4 4
 9 5 3 5 8 0 6 6 7 7 6 1 9 3 8 1 8 1 2 1 2 0 9 9 1 3 2 0 9 6 5 1 5 9 1 9 9
 4 9 7 0 5 3 0 3 8 3 8 1 6 8 9 4 9 7 5 0 0 5 0 9 9 2 1 0 2 7 5 6 8 2 6 4 4
 2 3 1 6 1 2 6 5 6 4 9 4 4 6 5 1 5 9 9 5 4 4 4 9 7 7 7 4 7 2 8 2 3 0 6 1 2
 4 9 8 8 3 0 6 4 2 6 6 2 0 9 4 3 1 3 2 0 9 6]
```

In [69]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [70]:
```
df=pd.read_csv(r"C:\USERS\user\Downloads\C1_ionosphere.csv")
df
```

Out[70]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.461
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.184
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.221
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.532
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.092
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.001
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.049
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.025
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.077
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.048

350 rows × 35 columns

In [71]:
```
df['g'].value_counts()
```

Out[71]:
```
g    224
b    126
Name: g, dtype: int64
```

In [72]:
```
x=df.drop('g',axis=1)
y=df['g']
```

In [73]:
```
g1={"g":{'g':1,'b':2}}
df=df.replace(g1)
print(df)
```

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

     0.03760  ...  -0.51171  0.41078  -0.46168  0.21266  -0.34090  0.42267  \
0   -0.04549  ...  -0.26569 -0.20468  -0.18401 -0.19040  -0.11593 -0.16626
1    0.01198  ...  -0.40220  0.58984  -0.22145  0.43100  -0.17365  0.60436
2    0.00000  ...   0.90695  0.51613   1.00000  1.00000  -0.20099  0.25682
3   -0.16399  ...  -0.65158  0.13290  -0.53206  0.02431  -0.62197 -0.05707
4    0.06637  ...  -0.01535 -0.03240   0.09223 -0.07859   0.00732  0.00000
..       ...  ...       ...      ...       ...      ...       ...      ...
345 -0.04622  ...  -0.04202  0.83479   0.00123  1.00000   0.12815  0.86660
346  0.01606  ...   0.01361  0.93522   0.04925  0.93159   0.08168  0.94066
347  0.02446  ...   0.03193  0.92489   0.02542  0.92120   0.02242  0.92459
348  0.00110  ...  -0.02099  0.89147  -0.07760  0.82983  -0.17238  0.96022
349 -0.09139  ...  -0.15114  0.81147  -0.04822  0.78207  -0.00703  0.75747

     -0.54487  0.18641  -0.45300  g
0    -0.06288 -0.13738  -0.02447  2
1    -0.24180  0.56045  -0.38238  1
2     1.00000 -0.32382   1.00000  2
3    -0.59573 -0.04608  -0.65697  1
4     0.00000 -0.00039   0.12011  2
..        ...      ...       ... ..
345  -0.10714  0.90546  -0.04307  1
346  -0.00035  0.91483   0.04712  1
347   0.00442  0.92697  -0.00577  1
348  -0.03757  0.87403  -0.16243  1
349  -0.06678  0.85764  -0.06151  1

[350 rows x 35 columns]
```

In [74]:
```
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 [75]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [76]:
```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[76]:
```
RandomForestClassifier()
```

In [77]:
```
parameters={'max_depth':[1,2,3,4,5],
            'min_samples_leaf':[5,10,15,20,25],
            'n_estimators':[10,20,30,40,50]}
```

In [78]:
```
from sklearn.model_selection import GridSearchCV
```

In [79]:
```
grid_search=GridSearchCV(estimator=rfc,param_grid=parameters,cv=2,scoring="accuracy")
grid_search.fit(x_train,y_train)
```

Out[79]:
```
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 [80]:
```
grid_search.best_score_
```

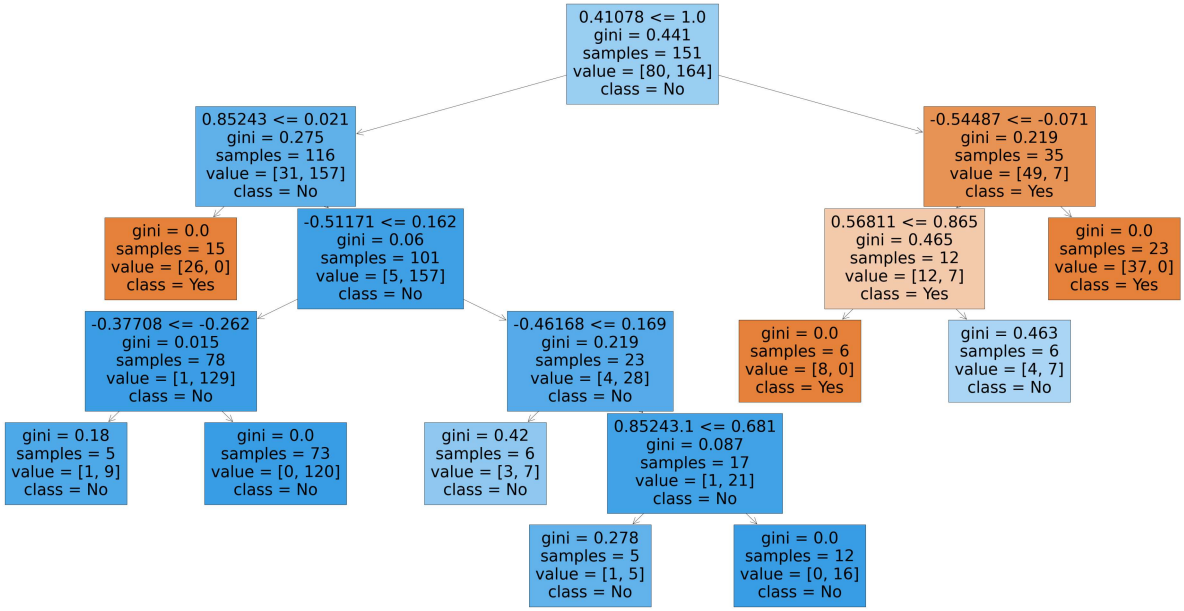
Out[80]:
```
0.930327868852459
```

In [81]:
```
rfc_best=grid_search.best_estimator_
```

In [82]:
```
from sklearn.tree import plot_tree
```

In [83]:
```
plt.figure(figsize=(80,40))
plot_tree(rfc_best.estimators_[5],feature_names=x.columns,class_names=['Yes','No'],filled=True
```

Out[83]:
```
[Text(2418.0, 1993.2, '0.41078 <= 1.0\ngini = 0.441\nsamples = 151\nvalue = [80, 164]\nclass
= No'),
 Text(1116.0, 1630.8000000000002, '0.85243 <= 0.021\ngini = 0.275\nsamples = 116\nvalue = [3
1, 157]\nclass = No'),
 Text(744.0, 1268.4, 'gini = 0.0\nsamples = 15\nvalue = [26, 0]\nclass = Yes'),
 Text(1488.0, 1268.4, '-0.51171 <= 0.162\ngini = 0.06\nsamples = 101\nvalue = [5, 157]\nclass
= No'),
 Text(744.0, 906.0, '-0.37708 <= -0.262\ngini = 0.015\nsamples = 78\nvalue = [1, 129]\nclass
= No'),
 Text(372.0, 543.5999999999999, 'gini = 0.18\nsamples = 5\nvalue = [1, 9]\nclass = No'),
 Text(1116.0, 543.5999999999999, 'gini = 0.0\nsamples = 73\nvalue = [0, 120]\nclass = No'),
 Text(2232.0, 906.0, '-0.46168 <= 0.169\ngini = 0.219\nsamples = 23\nvalue = [4, 28]\nclass =
No'),
 Text(1860.0, 543.5999999999999, 'gini = 0.42\nsamples = 6\nvalue = [3, 7]\nclass = No'),
 Text(2604.0, 543.5999999999999, '0.85243.1 <= 0.681\ngini = 0.087\nsamples = 17\nvalue = [1,
21]\nclass = No'),
 Text(2232.0, 181.19999999999982, 'gini = 0.278\nsamples = 5\nvalue = [1, 5]\nclass = No'),
 Text(2976.0, 181.19999999999982, 'gini = 0.0\nsamples = 12\nvalue = [0, 16]\nclass = No'),
 Text(3720.0, 1630.8000000000002, '-0.54487 <= -0.071\ngini = 0.219\nsamples = 35\nvalue = [4
9, 7]\nclass = Yes'),
 Text(3348.0, 1268.4, '0.56811 <= 0.865\ngini = 0.465\nsamples = 12\nvalue = [12, 7]\nclass =
Yes'),
 Text(2976.0, 906.0, 'gini = 0.0\nsamples = 6\nvalue = [8, 0]\nclass = Yes'),
 Text(3720.0, 906.0, 'gini = 0.463\nsamples = 6\nvalue = [4, 7]\nclass = No'),
 Text(4092.0, 1268.4, 'gini = 0.0\nsamples = 23\nvalue = [37, 0]\nclass = Yes')]
```



In [ ]:

In [ ]: