In [31]:

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

In [33]:

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

Out[33]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114

350 rows × 35 columns

In [34]:

```
from sklearn.linear_model import LogisticRegression
```

In [35]:

```
a=a.head(10)
a
```

Out[35]:

)	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.46168	
)	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-
)	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	
)	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	
)	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	
'	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.09223	-
3	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	...	-0.81634	0.13659	-0.82510	
)	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	...	1.00000	1.00000	1.00000	
5	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	...	-0.65440	0.57577	-0.69712	
1	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	...	-0.01326	0.20645	-0.02294	
)	0.06655	1.00000	-0.18388	1.00000	-0.27320	1.00000	-0.43107	...	-0.89128	0.47211	-0.86500	

umns

In [36]:

```
c=a.iloc[:,0:16]
d=a.iloc[:,-1]
```

In [37]:

```
c.shape
```

Out[37]:

```
(10, 16)
```

In [38]:

```
d.shape
```

Out[38]:

```
(10,)
```

In [39]:

```
from sklearn.preprocessing import StandardScaler
```

In [40]:

```
fs=StandardScaler().fit_transform(c)
```

In [41]:

```
logr=LogisticRegression()
logr.fit(fs,d)
```

Out[41]:

```
LogisticRegression()
```

In [42]:

```
e=[[2,5,77,8,6,5,4,66,88,46,65,76,87,45,92,44]]
```

In [43]:

```
prediction=logr.predict(e)
prediction
```

Out[43]:

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

In [44]:

```
logr.classes_
```

Out[44]:

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

In [45]:

```
logr.predict_proba(e)[0][0]
```

Out[45]:

```
0.0
```

In [46]:

```
logr.predict_proba(e)[0][1]
```

Out[46]:

```
1.0
```

In [47]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

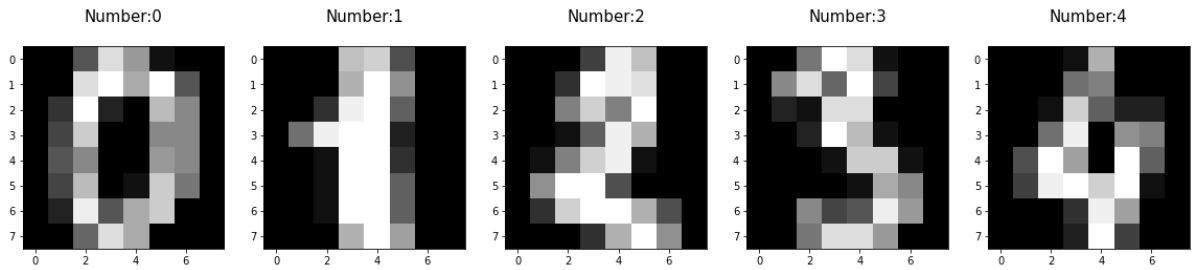
In [48]:

```
digits=load_digits()
digits
```

```
 'pixel_3_7',
 'pixel_4_0',
 'pixel_4_1',
 'pixel_4_2',
 'pixel_4_3',
 'pixel_4_4',
 'pixel_4_5',
 'pixel_4_6',
 'pixel_4_7',
 'pixel_5_0',
 'pixel_5_1',
 'pixel_5_2',
 'pixel_5_3',
 'pixel_5_4',
 'pixel_5_5',
 'pixel_5_6',
 'pixel_5_7',
 'pixel_6_0',
 'pixel_6_1',
```

In [49]:

```
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 [50]:

```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [51]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [52]:

```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[52]:

```
LogisticRegression(max_iter=10000)
```

In [53]:

```
logre.predict(x_test)
```

Out[53]:

```
array([9, 5, 0, 2, 7, 5, 4, 4, 3, 5, 7, 5, 2, 1, 7, 3, 9, 4, 2, 2, 4, 6,
       5, 9, 9, 2, 7, 5, 7, 2, 3, 7, 7, 4, 9, 4, 5, 8, 8, 3, 6, 5, 0, 7,
       7, 0, 7, 1, 7, 8, 7, 3, 4, 2, 9, 8, 6, 7, 0, 3, 9, 1, 9, 7, 5, 3,
       3, 9, 8, 7, 6, 9, 3, 7, 7, 8, 4, 7, 8, 2, 6, 0, 3, 7, 3, 6, 3, 7,
       1, 5, 5, 2, 0, 1, 5, 8, 0, 3, 7, 1, 5, 8, 4, 6, 5, 8, 9, 8, 2, 0,
       9, 0, 3, 5, 0, 0, 9, 4, 0, 6, 2, 0, 8, 9, 7, 1, 6, 6, 0, 2, 4, 6,
       1, 1, 5, 4, 7, 3, 9, 4, 3, 8, 7, 6, 9, 7, 7, 5, 5, 6, 8, 1, 9, 8,
       0, 3, 8, 7, 5, 5, 8, 8, 1, 4, 6, 2, 9, 6, 0, 1, 0, 4, 6, 2, 0, 8,
       0, 9, 3, 0, 9, 1, 3, 1, 4, 4, 1, 2, 7, 5, 3, 7, 2, 5, 9, 0, 6, 8,
       3, 4, 9, 4, 2, 6, 2, 2, 2, 0, 1, 9, 3, 0, 5, 3, 9, 9, 4, 1, 7, 4,
       5, 2, 6, 0, 3, 0, 5, 8, 4, 8, 5, 7, 7, 6, 6, 7, 2, 1, 9, 7, 6, 3,
       1, 4, 3, 8, 1, 1, 4, 4, 4, 7, 4, 8, 4, 0, 5, 1, 2, 4, 9, 6, 9, 7,
       6, 4, 7, 6, 7, 8, 0, 3, 6, 0, 2, 4, 8, 9, 2, 2, 3, 4, 4, 3, 0, 2,
       9, 0, 8, 1, 6, 2, 4, 8, 5, 8, 8, 2, 3, 0, 2, 7, 8, 6, 6, 2, 1, 1,
       5, 3, 4, 1, 7, 9, 0, 6, 5, 9, 9, 3, 6, 5, 5, 5, 0, 5, 5, 8, 6, 6,
       3, 0, 5, 4, 5, 5, 0, 0, 5, 0, 7, 3, 8, 8, 9, 4, 3, 6, 8, 8, 9, 8,
       4, 1, 3, 3, 8, 3, 8, 2, 5, 0, 5, 8, 2, 1, 3, 5, 4, 7, 0, 8, 9, 6,
       3, 4, 7, 5, 3, 6, 6, 1, 2, 1, 6, 7, 1, 6, 0, 1, 8, 9, 1, 7, 1, 2,
       7, 0, 7, 5, 6, 1, 3, 3, 8, 8, 9, 3, 4, 4, 3, 8, 3, 5, 9, 6, 4, 0,
       4, 4, 8, 9, 4, 9, 2, 5, 7, 5, 5, 4, 6, 4, 2, 4, 2, 6, 1, 3, 8, 9,
       4, 1, 5, 9, 2, 8, 8, 7, 3, 4, 7, 7, 7, 0, 4, 4, 6, 9, 3, 8, 5, 7,
       5, 2, 7, 0, 0, 0, 2, 7, 5, 6, 0, 3, 7, 1, 5, 5, 2, 8, 6, 5, 2, 7,
       0, 3, 7, 4, 1, 4, 6, 4, 0, 4, 0, 8, 1, 3, 9, 9, 7, 9, 8, 8, 9, 6,
       6, 7, 5, 3, 3, 2, 6, 3, 8, 3, 9, 4, 9, 8, 7, 8, 9, 6, 8, 1, 9, 6,
       5, 4, 8, 5, 6, 2, 5, 4, 5, 8, 1, 3])
```

In [54]:

```
logre.score(x_test,y_test)
```

Out[54]:

```
0.9611111111111111
```

In [55]:

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

In [56]:

```
a=pd.read_csv(r"C:\USERS\user\Downloads\C1_ionosphere.csv")
```

In [57]:

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

Out[57]:

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

In [58]:

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

In [59]:

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

```
     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 [60]:

```
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 [61]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [62]:

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
```

Out[62]:

```
RandomForestClassifier()
```

In [63]:

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

In [64]:

```
from sklearn.model_selection import GridSearchCV
```

In [65]:

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

Out[65]:

```
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 [66]:

```
grid_search.best_score_
```

Out[66]:

```
0.9385245901639344
```

In [67]:

```
rfc_best=grid_search.best_estimator_
```

In [68]:

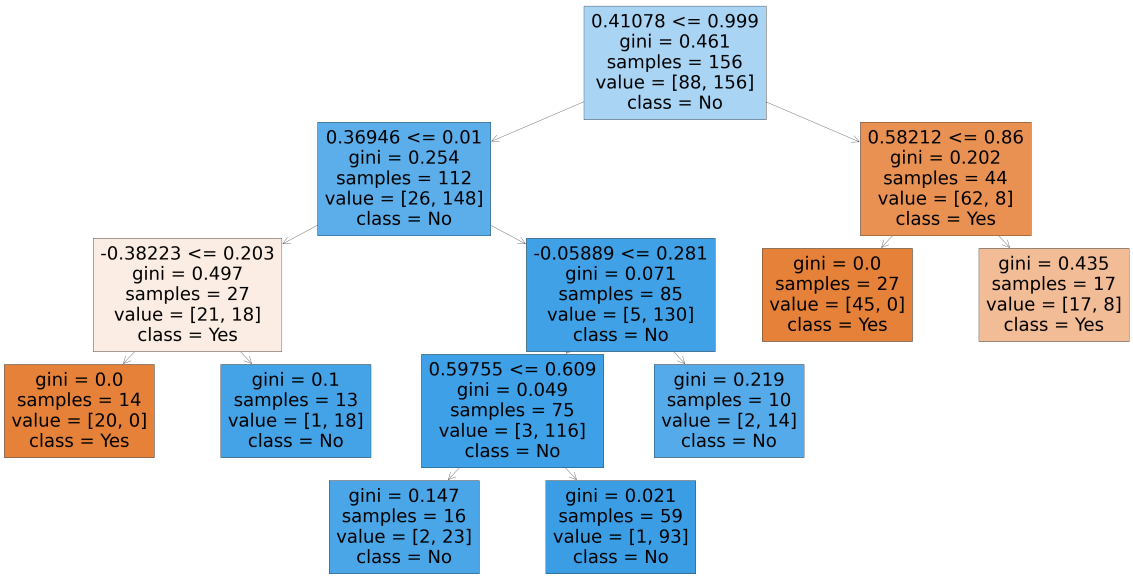
```
from sklearn.tree import plot_tree
```

In [69]:

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

Out[69]:

```
[Text(2637.818181818182, 1956.96, '0.41078 <= 0.999\ngini = 0.461\nsamples = 156
\nvalue = [88, 156]\nclass = No'),
 Text(1623.2727272727273, 1522.0800000000002, '0.36946 <= 0.01\ngini = 0.254\nsa
mples = 112\nvalue = [26, 148]\nclass = No'),
 Text(811.6363636363636, 1087.2, '-0.38223 <= 0.203\ngini = 0.497\nsamples = 27
\nvalue = [21, 18]\nclass = Yes'),
 Text(405.8181818181818, 652.3200000000002, 'gini = 0.0\nsamples = 14\nvalue =
[20, 0]\nclass = Yes'),
 Text(1217.4545454545455, 652.3200000000002, 'gini = 0.1\nsamples = 13\nvalue =
[1, 18]\nclass = No'),
 Text(2434.909090909091, 1087.2, '-0.05889 <= 0.281\ngini = 0.071\nsamples = 85
\nvalue = [5, 130]\nclass = No'),
 Text(2029.090909090909, 652.3200000000002, '0.59755 <= 0.609\ngini = 0.049\nsam
ples = 75\nvalue = [3, 116]\nclass = No'),
 Text(1623.2727272727273, 217.44000000000005, 'gini = 0.147\nsamples = 16\nvalue
= [2, 23]\nclass = No'),
 Text(2434.909090909091, 217.44000000000005, 'gini = 0.021\nsamples = 59\nvalue
= [1, 93]\nclass = No'),
 Text(2840.7272727272725, 652.3200000000002, 'gini = 0.219\nsamples = 10\nvalue
= [2, 14]\nclass = No'),
 Text(3652.3636363636365, 1522.0800000000002, '0.58212 <= 0.86\ngini = 0.202\nsa
mples = 44\nvalue = [62, 8]\nclass = Yes'),
 Text(3246.5454545454545, 1087.2, 'gini = 0.0\nsamples = 27\nvalue = [45, 0]\ncl
ass = Yes'),
 Text(4058.181818181818, 1087.2, 'gini = 0.435\nsamples = 17\nvalue = [17, 8]\nc
lass = Yes')]
```



In [ ]: