In [1]:

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

In [2]:

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

Out[2]:

-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0.46
-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18
-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22
-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00
-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53
-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0.09
...	...	...	...	...	...	...	...	...	...	
0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00
0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04
-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02
-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07
0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04

ıns

In [3]:

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

In [5]:

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

In [7]:

```
feature_matrix.shape
```

Out[7]:

```
(350, 34)
```

In [8]:

```
target_vector.shape
```

Out[8]:

```
(350,)
```

In [13]:

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

In [15]:

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

In [16]:

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

Out[16]:

```
LogisticRegression()
```

In [17]:

```
observation=[[1,2,4,5,3,6,8,9,7,0,-5,6,-8,-11,22,31,45,65,74,37,83,29,90,4,54,32,64,23,67
```

In [18]:

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

Out[18]:

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

In [19]:

```
logr.classes_
```

Out[19]:

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

In [21]:

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

Out[21]:

```
0.0
```

In [22]:

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

Out[22]:

```
1.0
```

In [23]:

```
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
```

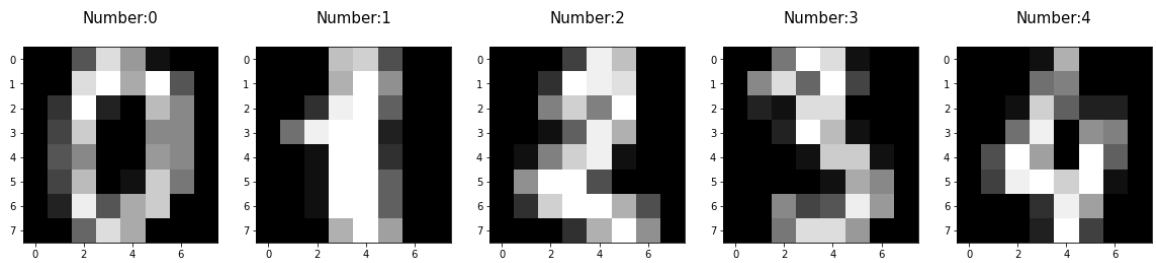
In [24]:

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

```
  'pixel_3_1',
  'pixel_3_2',
  'pixel_3_3',
  'pixel_3_4',
  'pixel_3_5',
  'pixel_3_6',
  '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',
```

In [25]:

```
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 [26]:

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

In [27]:

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

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

In [29]:

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

Out[29]:

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

In [30]:

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

Out[30]:

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

In [31]:

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

Out[31]:

```
0.9629629629629629
```

In [ ]: