LOGISTIC REGRESSION

IMPORTING LIBRARIES

In [29]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
```

In [4]:
```
df = pd.read_csv("1_ionosphere.csv")
```

Out[4]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01
...	...	...	...	...	...	...	...	...	...	...	...	...
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15

350 rows × 35 columns

In [5]:
```
f_m=df.iloc[:,0:34]
```

In [6]:
```
```

Out[6]:
```
(350, 34)
```

In [7]:
```
```

Out[7]:
```
(350,)
```

In [8]:
```
```

In [11]:
```
```

```
In [14]: logr=LogisticRegression()
```

```
Out[14]: LogisticRegression()
```

```
In [22]:
```

```
In [23]: prediction=logr.predict(observation)
```

```
Out[23]: array(['g'], dtype=object)
```

```
In [24]:
```

```
Out[24]: array(['b', 'g'], dtype=object)
```

```
In [25]:
```

```
Out[25]: 0.037620171662807955
```

```
In [26]:
```

```
Out[26]: 0.962379828337192
```

LOGISTICREGRESSION - II

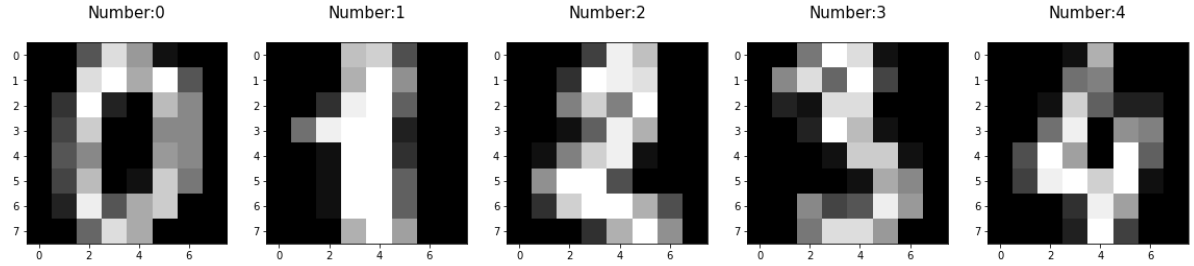
```
In [32]: import re
         from sklearn.datasets import load_digits
         from sklearn.linear_model import LogisticRegression
```

```
In [33]: digits=load_digits()
```

```
Out[33]: {'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 [36]:

```
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)
```



In [37]:

In [40]:

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

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

In [42]:

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

Out[42]:

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

In [44]:

```
Out[44]: array([8, 9, 4, 2, 6, 4, 8, 1, 3, 3, 9, 8, 0, 8, 5, 9, 3, 9, 8, 1, 0, 3,
                7, 7, 5, 6, 0, 6, 0, 7, 4, 4, 7, 4, 8, 5, 3, 5, 8, 2, 9, 4, 4, 1,
                4, 8, 2, 5, 8, 4, 2, 7, 1, 8, 7, 4, 2, 4, 7, 6, 5, 4, 4, 3, 9, 4,
                7, 5, 5, 5, 1, 0, 0, 7, 8, 8, 0, 9, 0, 3, 2, 5, 5, 0, 4, 3, 8, 0,
                0, 5, 5, 9, 6, 8, 4, 1, 3, 7, 6, 5, 0, 3, 6, 0, 8, 1, 6, 0, 0, 8,
                8, 2, 7, 3, 9, 2, 9, 2, 2, 2, 9, 5, 5, 2, 3, 9, 9, 4, 4, 6, 7, 5,
                2, 4, 3, 3, 4, 3, 4, 7, 0, 6, 0, 6, 2, 0, 7, 3, 1, 9, 0, 4, 3, 1,
                3, 1, 2, 4, 1, 2, 3, 2, 4, 7, 6, 2, 5, 4, 4, 9, 4, 7, 9, 2, 2, 9,
                4, 5, 6, 6, 3, 1, 9, 5, 7, 4, 6, 2, 7, 9, 3, 5, 4, 8, 7, 1, 3, 1,
                3, 3, 9, 6, 1, 6, 4, 4, 4, 6, 0, 4, 4, 5, 9, 5, 1, 1, 1, 3, 8, 9,
                2, 0, 0, 8, 5, 3, 8, 4, 5, 3, 5, 5, 0, 9, 3, 4, 2, 0, 0, 7, 9, 5,
                7, 5, 8, 7, 5, 1, 0, 8, 3, 5, 7, 4, 3, 2, 7, 7, 2, 6, 5, 8, 4, 1,
                4, 9, 2, 5, 1, 8, 6, 3, 7, 4, 0, 5, 1, 0, 1, 5, 0, 9, 9, 5, 1, 7,
                9, 5, 0, 4, 4, 6, 0, 2, 3, 5, 1, 9, 9, 4, 6, 8, 2, 0, 7, 1, 9, 6,
                2, 8, 1, 9, 7, 6, 3, 4, 4, 8, 8, 9, 4, 3, 5, 5, 1, 8, 6, 5, 8, 8,
                7, 1, 2, 9, 7, 5, 5, 9, 3, 2, 4, 7, 1, 0, 5, 1, 9, 6, 2, 8, 4, 5,
                6, 3, 1, 7, 2, 7, 6, 5, 3, 0, 1, 5, 6, 4, 2, 5, 5, 7, 2, 0, 3, 6,
                5, 0, 7, 4, 5, 0, 4, 9, 4, 5, 2, 5, 2, 2, 2, 8, 5, 1, 0, 8, 0, 0,
                5, 2, 0, 3, 9, 0, 7, 6, 1, 8, 6, 3, 2, 9, 1, 3, 4, 2, 9, 4, 6, 6,
                8, 1, 8, 6, 3, 1, 4, 2, 2, 5, 3, 9, 3, 3, 5, 6, 2, 9, 5, 6, 3, 2,
                8, 4, 6, 2, 4, 3, 9, 8, 1, 4, 5, 2, 3, 9, 5, 5, 5, 5, 8, 7, 8, 9,
                4, 2, 1, 7, 7, 2, 6, 9, 6, 7, 7, 6, 5, 6, 9, 9, 1, 2, 4, 1, 8, 5,
                9, 1, 9, 1, 1, 7, 5, 0, 0, 2, 8, 1, 1, 9, 1, 6, 5, 9, 0, 0, 8, 2,
                8, 3, 2, 6, 3, 1, 5, 3, 5, 5, 7, 8, 6, 1, 1, 8, 0, 8, 8, 2, 3, 7,
                6, 9, 7, 8, 9, 0, 8, 9, 1, 0, 6, 1])
```

In [43]:

```
Out[43]: 0.9648148148148148
```