In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv(r"1_ionosphere.csv")
df
```

Out[3]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.511
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.265
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.402
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.906
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.651
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.015
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.042
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.013
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.031
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.020
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.151

350 rows × 35 columns

In [4]:
```
feature_matrix=df.iloc[:,0:34]
target_vector=df.iloc[:,-1]
```

In [5]:
```
feature_matrix.shape
```

Out[5]:
```
(350, 34)
```

In [6]:
```
target_vector.shape
```

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

In [9]:
```
from sklearn.preprocessing import StandardScaler
```

In [12]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [13]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[13]:
```
LogisticRegression()
```

In [17]:
```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,2
```

In [18]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
['g']
```

In [19]:
```
logr.classes_
```

Out[19]:
```
array(['b', 'g'], dtype=object)
```

In [20]:
```
logr.predict_proba(observation)[0][0]
```

Out[20]:
```
7.175815497362237e-12
```

In [21]:
```
logr.predict_proba(observation)[0][1]
```

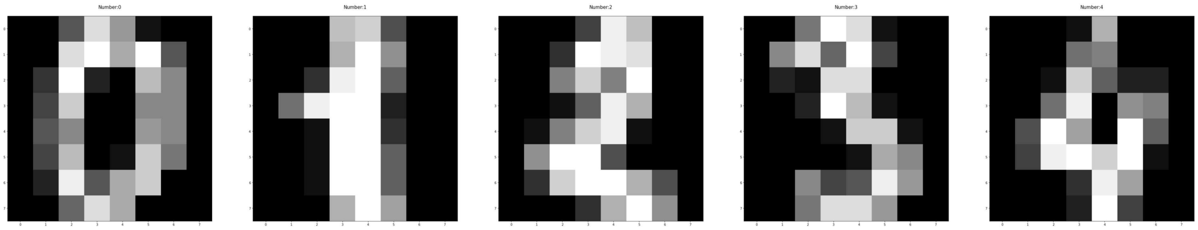
Out[21]:
```
0.9999999999928242
```

In [33]:
```
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
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [34]:
```
digit=load_digits()
digit
```

Out[34]:
```
{'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 [48]:
```
plt.figure(figsize=(69,69))
for index,(image,label) in enumerate(zip(digit.data[0:5],digit.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 [39]:
```
x_train,x_test,y_train,y_test=train_test_split(digit.data,digit.target,test_si
```

In [40]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [41]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[41]:
```
LogisticRegression(max_iter=10000)
```

In [42]:
```
print(logre.predict(x_test))
```

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

In [43]:
```
print(logre.score(x_test,y_test))
```

```
0.9629629629629629
```

In [ ]: