In [27]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [28]:

In [29]:
```
dftrain=pd.read_csv(r"C:\USERS\user\Downloads\C6_bmi - C6_bmi.csv")
```

Out[29]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
...	...	...	...	...
495	Female	150	153	5
496	Female	184	121	4
497	Female	141	136	5
498	Male	150	95	5
499	Male	173	131	5

500 rows × 4 columns

In [30]:

Out[30]:
```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [31]:
```
a=dftrain[['Height','Weight','Index']]
```

Out[31]:

	Height	Weight	Index
0	174	96	4
1	189	87	2
2	185	110	4
3	195	104	3
4	149	61	3
...	...	...	...
495	150	153	5
496	184	121	4
497	141	136	5
498	150	95	5
499	173	131	5

500 rows × 3 columns

In [32]:
```
b=dftrain.head(10)
```

Out[32]:

	Gender	Height	Weight	Index
0	Male	174	96	4
1	Male	189	87	2
2	Female	185	110	4
3	Female	195	104	3
4	Male	149	61	3
5	Male	189	104	3
6	Male	147	92	5
7	Male	154	111	5
8	Male	174	90	3
9	Female	169	103	4

In [33]:
```
a=b[['Height','Weight','Index']]
```

Out[33]:

	Height	Weight	Index
0	174	96	4
1	189	87	2
2	185	110	4
3	195	104	3
4	149	61	3
5	189	104	3
6	147	92	5
7	154	111	5
8	174	90	3
9	169	103	4

In [34]:
```
c=a.iloc[:,0:3]
```

In [35]:

Out[35]:
```
(10, 3)
```

In [36]:

Out[36]:
```
(10,)
```

In [37]:

In [38]:

In [39]:
```
logr=LogisticRegression()
```

Out[39]:
```
LogisticRegression()
```

In [40]:

In [41]:
```
prediction=logr.predict(observation)
```

Out[41]:
```
array([2], dtype=int64)
```

In [42]:

Out[42]:
```
array([2, 3, 4, 5], dtype=int64)
```

In [43]:

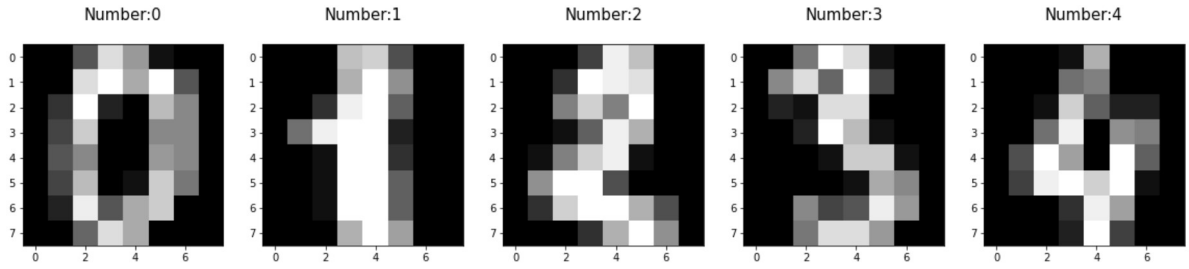
Out[43]:
```
0.6255526253350464
```

In [44]:
```
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
```

In [45]:
```
digits=load_digits()
```

Out[45]:
```
{'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 [46]:
```
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 [47]:
```
```

In [48]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
```

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

In [49]:
```
logre=LogisticRegression(max_iter=10000)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:

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

In [51]:

```
0.9685185185185186
```

In [ ]: