In [1]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sn
```

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

In [3]:
```
1 df=pd.read_csv(r"C6_bmi")
```

In [4]:
```
1 df
```

Out[4]:

	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 [5]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Gender  500 non-null    object
 1   Height  500 non-null    int64
 2   Weight  500 non-null    int64
 3   Index   500 non-null    int64
dtypes: int64(3), object(1)
memory usage: 15.8+ KB
```

In [6]:
```
1 df.dropna(inplace=True)
```

In [7]:
```
1 df.columns
```

Out[7]:
```
Index(['Gender', 'Height', 'Weight', 'Index'], dtype='object')
```

In [8]:
```
1 df
```

Out[8]:

	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 [9]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [10]:
```
1 logr =LogisticRegression()
```

In [11]:
```
1 feature_matrix=df[['Height', 'Weight', 'Index']]
2 target_vector=df['Gender']
```

In [12]:
```
1 feature_matrix.shape
```

Out[12]:
```
(500, 3)
```

In [13]:
```
1 target_vector.shape
```

Out[13]:
```
(500,)
```

In [14]:
```
1 from sklearn.preprocessing import StandardScaler
```

In [15]:
```
1 fs=StandardScaler().fit_transform(feature_matrix)
```

In [16]:
```
1 logr=LogisticRegression()
2 logr.fit(fs,target_vector)
```

Out[16]:
```
LogisticRegression()
```

In [19]:
```
1 observation=[[1,2,3]]
```

In [20]:
```
1 prediction = logr.predict(observation)
2 print(prediction)
```

```
['Male']
```

In [21]:
```
1 logr.classes_
```

Out[21]:
```
array(['Female', 'Male'], dtype=object)
```

In [22]:
```
1 logr.predict_proba(observation)[0][1]
```

Out[22]:
```
0.5571020917548749
```

In [23]:
```
1 logr.predict_proba(observation)[0][0]
```

Out[23]:
```
0.4428979082451251
```

Linear regression 2

In [24]:

```
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
```

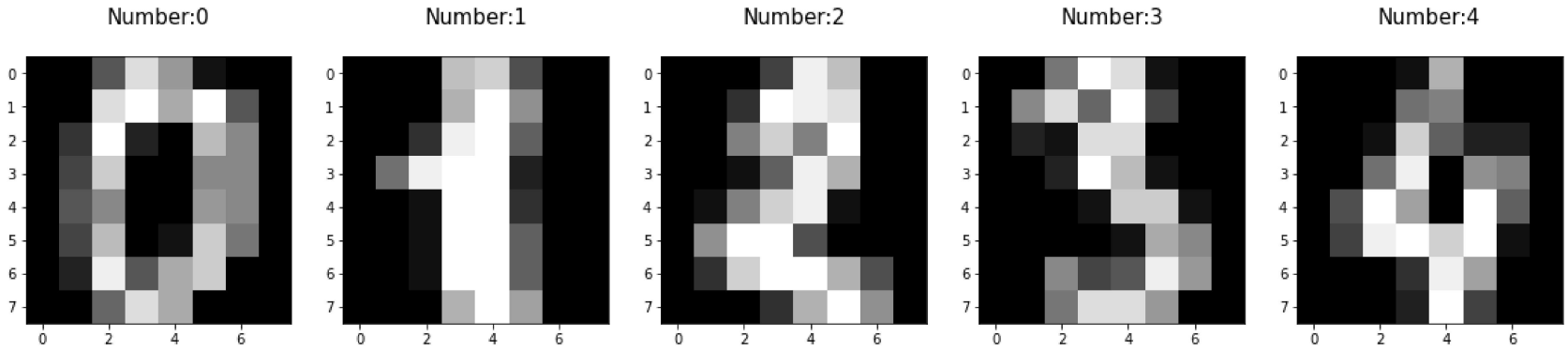
In [25]:

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

Out[25]:

```
{'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 [26]:
```
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 [27]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [28]:
```
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]:
```
print(logre.predict(x_test))
```

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

In [31]:
```
print(logre.score(x_test,y_test))
```

```
0.9703703703703703
```