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

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\bm.csv")
df
```

Out[2]:

	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 [3]:
```
feature_matrix=df.iloc[:,1:3]
target_vector=df.iloc[:,-1]
```

In [4]:
```
feature_matrix.shape
```

Out[4]: (500, 2)

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

Out[5]: (500,)

In [6]:
```
from sklearn.preprocessing import StandardScaler
```

In [7]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [8]:
```
logr=LogisticRegression()
```

In [9]:
```
logr.fit(fs,target_vector)
```

Out[9]:
```
LogisticRegression()
```

In [10]:
```
observation=[[1,2]]
```

In [11]:
```
prediction=logr.predict(observation)
```

In [12]:
```
print(prediction)
```

```
[5]
```

In [13]:
```
logr.classes_
```

Out[13]:
```
array([0, 1, 2, 3, 4, 5], dtype=int64)
```

In [14]:
```
logr.predict_proba(observation)[0][0]
```

Out[14]:
```
5.595669758253823 7e-11
```

In [15]:
```
logr.predict_proba(observation)[0][1]
```

Out[15]:
```
6.059900360819463e-10
```

Logistic Regression 2

In [16]:
```
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

In [17]:
```
digits=load_digits()
digits
```

Out[17]:
```
{'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',
```

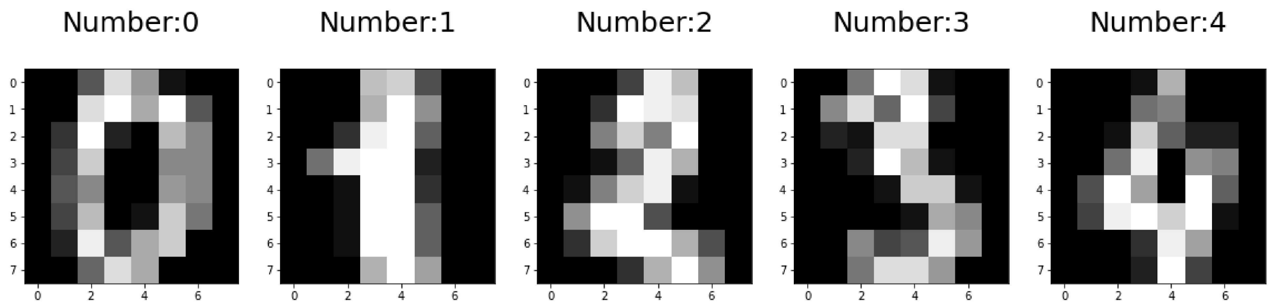
```
                 'pixel_0_2',
                 'pixel_0_3',
                 'pixel_0_4',
                 'pixel_0_5',
                 'pixel_0_6',
                 'pixel_0_7',
                 'pixel_1_0',
                 'pixel_1_1',
                 'pixel_1_2',
                 'pixel_1_3',
                 'pixel_1_4',
                 'pixel_1_5',
                 'pixel_1_6',
                 'pixel_1_7',
                 'pixel_2_0',
                 'pixel_2_1',
                 'pixel_2_2',
                 'pixel_2_3',
                 'pixel_2_4',
                 'pixel_2_5',
                 'pixel_2_6',
                 'pixel_2_7',
                 'pixel_3_0',
                 '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',
                 'pixel_5_4',
                 'pixel_5_5',
                 'pixel_5_6',
                 'pixel_5_7',
                 'pixel_6_0',
                 'pixel_6_1',
                 'pixel_6_2',
                 'pixel_6_3',
                 'pixel_6_4',
                 'pixel_6_5',
                 'pixel_6_6',
                 'pixel_6_7',
                 'pixel_7_0',
                 'pixel_7_1',
                 'pixel_7_2',
                 'pixel_7_3',
                 'pixel_7_4',
                 'pixel_7_5',
                 'pixel_7_6',
                 'pixel_7_7'],
                'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
                'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
                        [ 0.,  0., 13., ..., 15.,  5.,  0.],
```

```
        [ 0.,  3., 15., ..., 11.,  8.,  0.],
        ...,
        [ 0.,  4., 11., ..., 12.,  7.,  0.],
        [ 0.,  2., 14., ..., 12.,  0.,  0.],
        [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

       [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
        [ 0.,  0.,  0., ...,  9.,  0.,  0.],
        [ 0.,  0.,  3., ...,  6.,  0.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

       [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
        [ 0.,  0.,  3., ..., 14.,  0.,  0.],
        [ 0.,  0.,  8., ..., 16.,  0.,  0.],
        ...,
        [ 0.,  9., 16., ...,  0.,  0.,  0.],
        [ 0.,  3., 13., ..., 11.,  5.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
        [ 0.,  0., 15., ..., 15.,  0.,  0.],
        ...,
        [ 0.,  4., 16., ..., 16.,  6.,  0.],
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n---
-----------------------------------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 1797\n    :Number of Attributes: 64\n    :Attribute Information: 8
x8 image of integer pixels in the range 0..16.\n    :Missing Attribute Values: None\n
:Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a cop
y of the test set of the UCI ML hand-written digits datasets\nhttps://archive.ics.uci.ed
u/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe data set contains images
of hand-written digits: 10 classes where\neach class refers to a digit.\n\nPreprocessing
programs made available by NIST were used to extract\nnormalized bitmaps of handwritten
digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the trainin
g set and different 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping b
locks of\n4x4 and the number of on pixels are counted in each block. This generates\nan
input matrix of 8x8 where each element is an integer in the range\n0..16. This reduces d
imensionality and gives invariance to small\ndistortions.\n\nFor info on NIST preprocess
ing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P.
J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based Handprint Recognition Syste
m, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Kaynak (1995) Methods of Combi
ning Multiple Classifiers and Their\n    Applications to Handwritten Digit Recognition,
```

```
MSc Thesis, Institute of\n    Graduate Studies in Science and Engineering, Bogazici Univ
ersity.\n  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken
Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityr
eduction using relevance weighted LDA. School of\n    Electrical and Electronic Engineer
ing Nanyang Technological University.\n    2005.\n  - Claudio Gentile. A New Approximate
Maximal Margin Classification\n    Algorithm. NIPS. 2000.\n"}
```

In [18]:

```python
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=25)
```



In [19]:

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

In [20]:

```python
logr=LogisticRegression(max_iter=10000)
```

In [21]:

```python
logr.fit(x_train,y_train)
```

Out[21]:

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

In [22]:

```python
print(logr.predict(x_test))
```

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

In [23]:

```python
print(logr.score(x_test,y_test))
```

```
0.9740740740740741
```