In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Logistic Regression

In [2]:

```
from sklearn.linear_model import LogisticRegression
```

In [4]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\C6_bmi.csv")
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]:

```
df.columns
```

Out[5]:

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

In [6]:

```
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 [8]:

```
feature_matrix=df[[ 'Height', 'Weight', 'Index']]
target_vector = df[['Index']]
```

In [9]:

```
feature_matrix.shape
```

Out[9]:

```
(500, 3)
```

In [10]:

```
target_vector.shape
```

Out[10]:

```
(500, 1)
```

In [11]:

```
from sklearn.preprocessing import StandardScaler
```

In [12]:

```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [13]:

```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63:
DataConversionWarning: A column-vector y was passed when a 1d array was ex
pected. Please change the shape of y to (n_samples, ), for example using r
avel().
  return f(*args, **kwargs)
```

Out[13]:

```
LogisticRegression()
```

In [15]:

```
observation=[[1.4,2,3]]
# Take Random Values
```

In [16]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
[5]
```

In [17]:

```
logr.classes_
```

Out[17]:

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

In [18]:

```
logr.predict_proba(observation)[0][0]
```

Out[18]:

```
1.4948362317773233e-24
```

In [19]:

```
logr.predict_proba(observation)[0][1]
```

Out[19]:

```
7.130311227881209e-21
```

Logistic Regression-2

In [20]:

```
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 [21]:

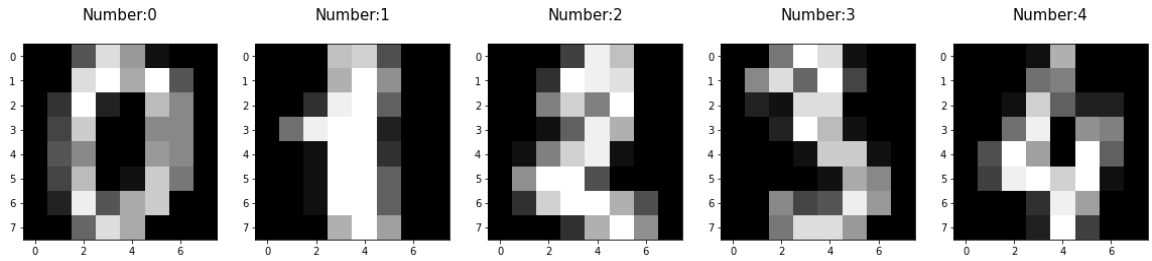
```
digits = load_digits()
digits
```

Out[21]:

```
{'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',
```

In [22]:

```
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 [23]:

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

In [24]:

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

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

In [25]:

```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[25]:

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

In [26]:

```
print(logre.predict(x_test))
```

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

In [27]:

```
print(logre.score(x_test,y_test))
```

```
0.9537037037037037
```