In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\C6_bmi - C6_bmi.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]:
```
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 [4]:
```
from sklearn.linear_model import LogisticRegression
```

In [10]:
```
df1=df[['Height','Weight','Index']]
```

In [42]:
```
feature_matrix = df1.iloc[:,0:3]
target_vector = df1.iloc[:,1]
```

In [43]:
```
feature_matrix.shape
```

Out[43]:
```
(500, 3)
```

In [44]:
```
target_vector.shape
```

Out[44]:
```
(500,)
```

In [27]:
```
from sklearn.preprocessing import StandardScaler
```

In [28]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [29]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[29]:
```
LogisticRegression()
```

In [30]:
```
observation=[[5,7,9]]
```

In [31]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[153]
```

In [32]:
```
logr.classes_
```

Out[32]:
```
array([ 50,  51,  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,
        63,  64,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,
        76,  77,  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,
        89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101,
       102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 114, 115,
       116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128,
       129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
       142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
       155, 156, 157, 158, 159, 160], dtype=int64)
```

In [33]:
```
logr.predict_proba(observation)[0][0]
```

Out[33]:
```
4.012934825172367e-30
```

In [34]:
```
logr.predict_proba(observation)[0][0]
```

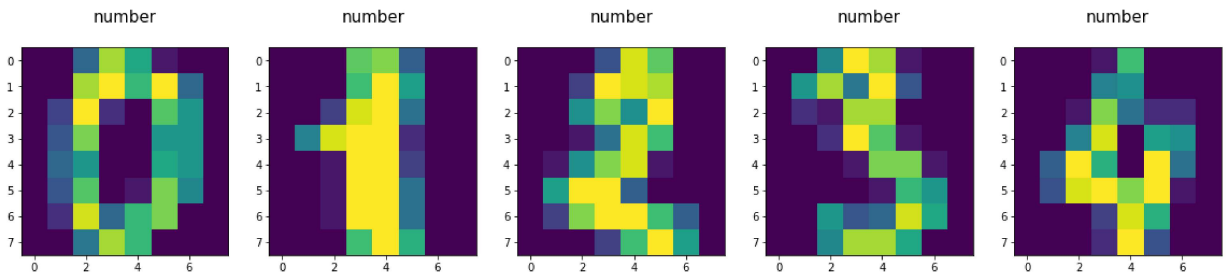
Out[34]:
```
4.012934825172367e-30
```

In [35]:
```
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 [36]:
```
digits = load_digits()
digits
```

```
'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',
```

In [37]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:8])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)))
    plt.title("number\n"%label,fontsize=15)
```



In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_siz
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [39]:
```
logre=LogisticRegression()
logre.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

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

In [40]:
```
print(logre.predict(x_test))
```

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

In [41]:
```
print(logre.score(x_test,y_test))
```

```
0.9555555555555556
```