In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
df=pd.read_csv("C4_framingham - C4_framingham.csv")
df
```

Out[2]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabe
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
3	0	61	3.0	1	30.0	0.0	0	1	
4	0	46	3.0	1	23.0	0.0	0	0	
...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	
4234	1	51	3.0	1	43.0	0.0	0	0	
4235	0	48	2.0	1	20.0	NaN	0	0	
4236	0	44	1.0	1	15.0	0.0	0	0	
4237	0	52	2.0	0	0.0	0.0	0	0	

4238 rows × 16 columns

In [10]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64  
```

```
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

In [11]:

```
df1=df.fillna(value=0)
df1
```

Out[11]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabe
0	1	39	4.0	0	0.0	0.0	0	0	
1	0	46	2.0	0	0.0	0.0	0	0	
2	1	48	1.0	1	20.0	0.0	0	0	
3	0	61	3.0	1	30.0	0.0	0	1	
4	0	46	3.0	1	23.0	0.0	0	0	
...	...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	1	
4234	1	51	3.0	1	43.0	0.0	0	0	
4235	0	48	2.0	1	20.0	0.0	0	0	
4236	0	44	1.0	1	15.0	0.0	0	0	
4237	0	52	2.0	0	0.0	0.0	0	0	

4238 rows × 16 columns

In [27]:

```
feature_matrix=df1.iloc[:,0:15]
target_vector=df1.iloc[:,-1]
```

In [28]:

```
feature_matrix.shape
```

Out[28]: (4238, 15)

In [29]:

```
target_vector.shape
```

Out[29]: (4238,)

In [30]:

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

In [31]:

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

In [32]:

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

Out[32]: LogisticRegression()

In [35]:
```
observation=[[1.4,2.3,5.0,11,12,13,14,15,3,4,5,7,6,7,13]]
```

In [36]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[1]
```

In [37]:
```
logr.classes_
```

Out[37]:
```
array([0, 1], dtype=int64)
```

In [38]:
```
logr.predict_proba(observation)[0][0]
```

Out[38]:
```
0.00016936367260200758
```

In [39]:
```
logr.predict_proba(observation)[0][1]
```

Out[39]:
```
0.999830636327398
```

In [40]:
```
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 [41]:
```
digits = load_digits()
digits
```

Out[41]:
```
{'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',
```

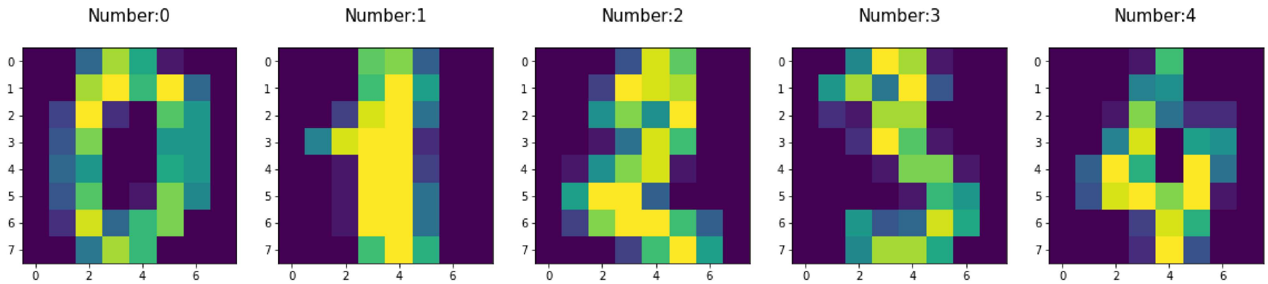
```
  'pixel_1_4',
  'pixel_1_5',
  'pixel_1_6',
  'pixel_1_7',
  'pixel_2_0',
  'pixel_2_1',
  'pixel_2_2',
  'pixel_2_3',
  'pixel_2_4',
  'pixel_2_5',
  'pixel_2_6',
  'pixel_2_7',
  'pixel_3_0',
  'pixel_3_1',
  'pixel_3_2',
  'pixel_3_3',
  'pixel_3_4',
  'pixel_3_5',
  'pixel_3_6',
  'pixel_3_7',
  '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 [42]:

```
plt.figure(figsize=(20,4))
for index,(image,label)in enumerate(zip(digits.data[0:5],digits.target[0:5])):
```

```python
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)))
    plt.title('Number:%i\n' %label,fontsize=15)
```



In [45]:

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

In [46]:

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

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

In [47]:

```python
logre=LogisticRegression(max_iter=10000) # if error comes declare max_iter=10000
logre.fit(x_train,y_train)
```

Out[47]:

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

In [49]:

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

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

In [48]:

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

```
0.9685185185185186
```

In [ ]: