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 [3]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\C4_framingham.csv")
df
```

Out[3]:

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

4238 rows × 16 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMed
s',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [5]:

```
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 [6]:

```
df2=df.fillna("1")
df2
```

Out[6]:

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

4238 rows × 16 columns

In [7]:

```
feature_matrix=df[['male', 'age', 'currentSmoker','prevalentStroke','prevalentHyp','diabe
target_vector = df[['currentSmoker']]
```

In [8]:

```
feature_matrix.shape
```

Out[8]:

```
(4238, 7)
```

In [9]:

```
target_vector.shape
```

Out[9]:

```
(4238, 1)
```

In [10]:

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

In [11]:

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

In [12]:

```
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[12]:

```
LogisticRegression()
```

In [13]:

```
observation=[[1.4,2,3,4,5,6,9.5]]
# Take Random Values
```

In [14]:

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

```
[1]
```

In [15]:

```
logr.classes_
```

Out[15]:

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

In [16]:

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

Out[16]:

```
5.18655207670804e-09
```

In [17]:

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

Out[17]:

```
0.9999999948134479
```

Logistic Regression-2

In [18]:

```
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
```

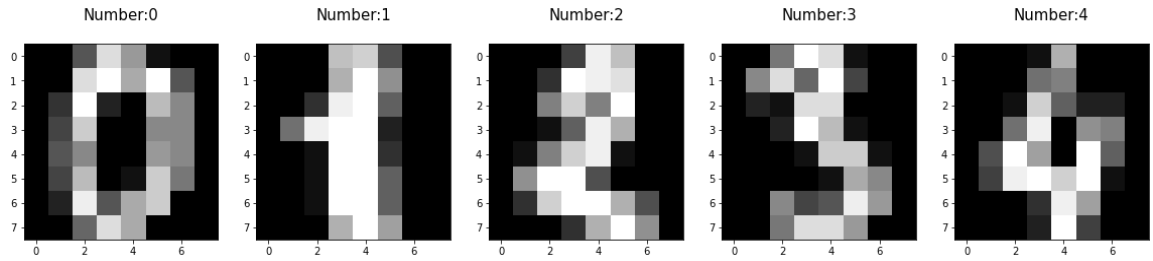
In [19]:

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

```
        ...,
        [ 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 dig
its dataset\n--------------------------------------------------\n\n**Dat
a Set Characteristics:**\n\n    :Number of Instances: 1797\n    :Number
of Attributes: 64\n    :Attribute Information: 8x8 image of integer pixe
ls in the range 0..16.\n    :Missing Attribute Values: None\n    :Creato
r: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis
is a copy of the test set of the UCI ML hand-written digits datasets\nht
tps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten
+Digits\n\nThe data set contains images of hand-written digits: 10 class
es where\neach class refers to a digit.\n\nPreprocessing programs made a
vailable by NIST were used to extract\nnormalized bitmaps of handwritten
digits from a preprinted form. From a\ntotal of 43 people, 30 contribute
d to the training set and different 13\nto the test set. 32x32 bitmaps a
re divided into nonoverlapping blocks of\n4x4 and the number of on pixel
s are counted in each block. This generates\nan input matrix of 8x8 wher
e each element is an integer in the range\n0..16. This reduces dimension
```

In [20]:

```
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 [21]:

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

In [22]:

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

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

In [23]:

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

Out[23]:

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

In [24]:

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

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

In [25]:

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

```
0.9574074074074074
```