In [1]:

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

In [2]:

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

In [83]:

```
d=pd.read_csv(r"C:\Users\user\Downloads\fra.csv")[0:13]
d
```

Out[83]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHy
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	(
5	0	43	2.0	0	0.0	0.0	0	'
6	0	63	1.0	0	0.0	0.0	0	(
7	0	45	2.0	1	20.0	0.0	0	(
8	1	52	1.0	0	0.0	0.0	0	'
9	1	43	1.0	1	30.0	0.0	0	'
10	0	50	1.0	0	0.0	0.0	0	(
11	0	43	2.0	0	0.0	0.0	0	(
12	1	46	1.0	1	15.0	0.0	0	'

Logistic Regression

In [84]:

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

In [85]:

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

In [86]:

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

In [87]:

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

Out[87]:

```
LogisticRegression()
```

In [91]:

```
observation=[[1,2,3,4,5,6,7,8,9,10,11,12,12,14,15
             ]]
```

In [92]:

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

```
[0]
```

In [93]:

```
logr.classes_
```

Out[93]:

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

In [94]:

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

Out[94]:

```
0.6613205387211075
```

Logistic Regression-2

In [95]:

```
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 [96]:

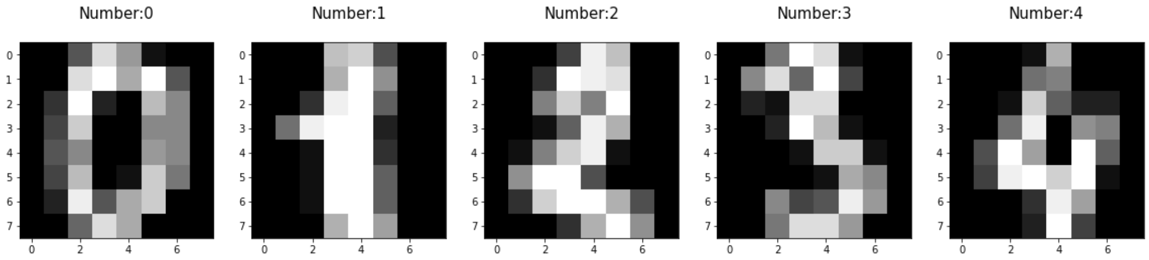
```
digits=load_digits()
digits
```

Out[96]:

```
{'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 [97]:

```
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 [98]:

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

In [99]:

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

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

In [100]:

```
logr=LogisticRegression(max_iter=100000)
```

In [101]:

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

Out[101]:

```
LogisticRegression(max_iter=100000)
```

In [102]:

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

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

In [103]:

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

```
0.9611111111111111
```

In [ ]:

```

```