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 [51]:

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

Out[51]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

Logistic Regression

In [52]:

```
feature_matrix=d.iloc[:,0:8]
target_vector=d.iloc[:,-1]
```

In [53]:

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

In [54]:

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

In [56]:

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

Out[56]:

```
LogisticRegression()
```

In [59]:

```
observation=[[1,2,3,4,5,6,7,8
             ]]
```

In [60]:

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

```
[1]
```

In [61]:

```
logr.classes_
```

Out[61]:

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

In [62]:

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

Out[62]:

```
0.00029236948687560993
```

Logistic Regression-2

In [63]:

```
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 [64]:

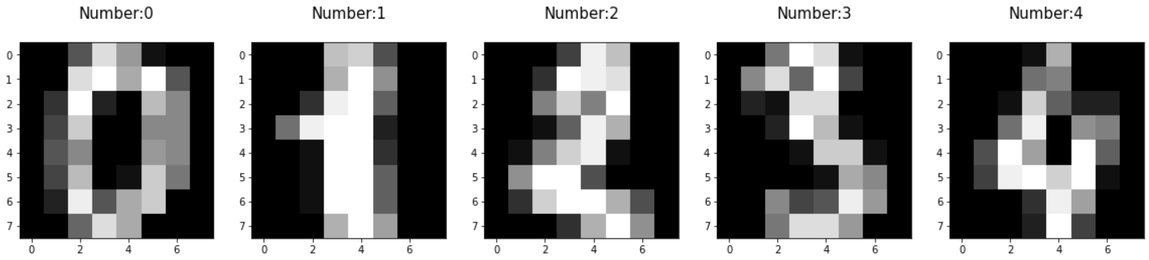
```
digits=load_digits()
digits
```

Out[64]:

```
{'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 [65]:

```
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 [66]:

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

In [67]:

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

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

In [68]:

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

In [69]:

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

Out[69]:

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

In [70]:

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

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

In [71]:

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

```
0.9722222222222222
```

In [ ]:

```

```