LogisticRegression1

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

In [2]:
```
"C:\Users\user\Downloads\C5_health care diabetes - C5_health care diabetes.csv")
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288
...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171
764	2	122	70	27	0	36.8	0.340
765	5	121	72	23	112	26.2	0.245
766	1	126	60	0	0	30.1	0.349
767	1	93	70	31	0	30.4	0.315

768 rows × 9 columns

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [4]:
```
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
feature_matrix = df.iloc[:,0:5]
target_vector = df.iloc[:,1]
```

In [6]:
```
feature_matrix.shape
```

Out[6]:
```
(768, 5)
```

In [7]:
```
target_vector.shape
```

Out[7]:
```
(768,)
```

In [8]:
```
from sklearn.preprocessing import StandardScaler
```

In [9]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [10]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[10]:
```
LogisticRegression()
```

In [11]:
```
observation=[[5,7,9,5,6]]
```

In [12]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[189]
```

In [13]:
```
logr.classes_
```

Out[13]:
```
array([  0,  44,  56,  57,  61,  62,  65,  67,  68,  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, 113,
       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, 161, 162, 163, 164, 165,
       166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
       179, 180, 181, 182, 183, 184, 186, 187, 188, 189, 190, 191, 193,
       194, 195, 196, 197, 198, 199], dtype=int64)
```

In [14]:
```
logr.predict_proba(observation)[0][0]
```

Out[14]:
```
8.192298469933462e-26
```

In [15]:
```
logr.predict_proba(observation)[0][0]
```

Out[15]:
```
8.192298469933462e-26
```

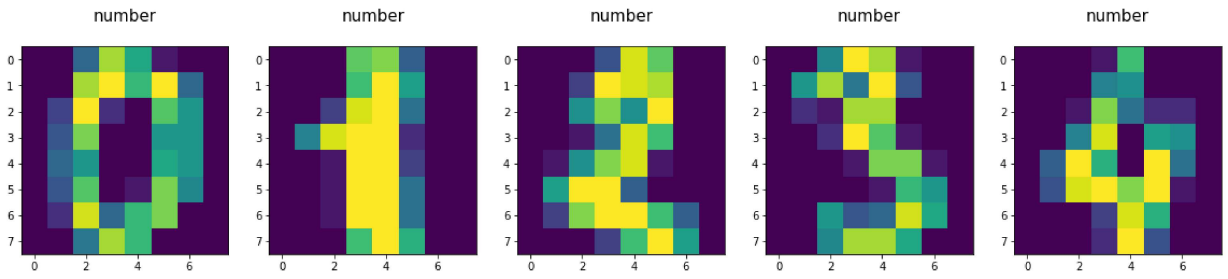
LogisticRegression2

In [16]:
```
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 [17]:
```
digits = load_digits()
digits
```

Out[17]:
```
{'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',
```

In [18]:
```
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 [19]:
```
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 [20]:
```
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[20]:
```
LogisticRegression()
```

In [21]:
```
print(logre.predict(x_test))
```

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

In [22]:
```
print(logre.score(x_test,y_test))
```

```
0.9611111111111111
```