In [6]:

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

Logistic Regression

In [7]:

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

In [8]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\C5_health care diabetes.csv")
df
```

Out[8]:

	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

In [9]:

```
df.columns
```

Out[9]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [10]:

```
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 [11]:

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

Out[11]:

	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

In [12]:

```
feature_matrix=df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
target_vector = df[['Outcome']]
```

In [13]:

```
feature_matrix.shape
```

Out[13]:

```
(768, 7)
```

In [14]:

```
target_vector.shape
```

Out[14]:

```
(768, 1)
```

In [15]:

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

In [16]:

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

In [17]:

```
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[17]:

```
LogisticRegression()
```

In [18]:

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

In [19]:

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

```
[1]
```

In [20]:

```
logr.classes_
```

Out[20]:

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

In [21]:

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

Out[21]:

```
0.0
```

In [22]:

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

Out[22]:

```
1.0
```

Logistic Regression-2

In [23]:

```
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
```

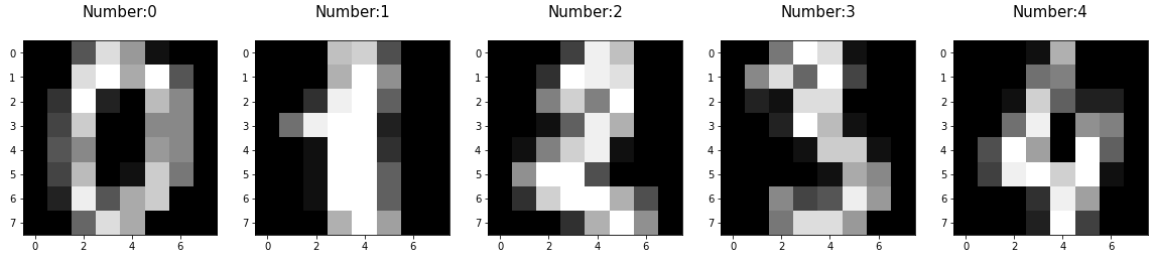
In [24]:

```
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 [25]:

```
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 [26]:

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

In [27]:

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

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

In [28]:

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

Out[28]:

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

In [29]:

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

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

In [30]:

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

```
0.9685185185185186
```