In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [16]:
```
df=pd.read_csv("C5_health care diabetes - C5_health care diabetes.csv")
df
```

Out[16]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Ou
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	
...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	
764	2	122	70	27	0	36.8	0.340	27	
765	5	121	72	23	112	26.2	0.245	30	
766	1	126	60	0	0	30.1	0.349	47	
767	1	93	70	31	0	30.4	0.315	23	

768 rows × 9 columns

In [17]:
```
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 [18]:
```
df1=df.fillna(value=0)
df1
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Ou
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	
...	...	...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.171	63	
764	2	122	70	27	0	36.8	0.340	27	
765	5	121	72	23	112	26.2	0.245	30	
766	1	126	60	0	0	30.1	0.349	47	
767	1	93	70	31	0	30.4	0.315	23	

768 rows × 9 columns

In [32]:
```
feature_matrix=df1.iloc[:,0:8]
target_vector=df1.iloc[:,-1]
```

In [33]:
```
feature_matrix.shape
```

Out[33]:
```
(768, 8)
```

In [34]:
```
target_vector.shape
```

Out[34]:
```
(768,)
```

In [35]:
```
from sklearn.preprocessing import StandardScaler
```

In [36]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [37]:
```
logr =LogisticRegression()
logr.fit(fs,target_vector)
```

Out[37]:
```
LogisticRegression()
```

In [41]:
```
observation=[[1.4,2.3,5.0,11,12,13,14,15]]
```

In [42]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[1]
```

In [43]:
```
logr.classes_
```

Out[43]:
```
array([0, 1], dtype=int64)
```

In [44]:
```
logr.predict_proba(observation)[0][0]
```

Out[44]:
```
1.764793530201203e-07
```

In [45]:
```
logr.predict_proba(observation)[0][1]
```

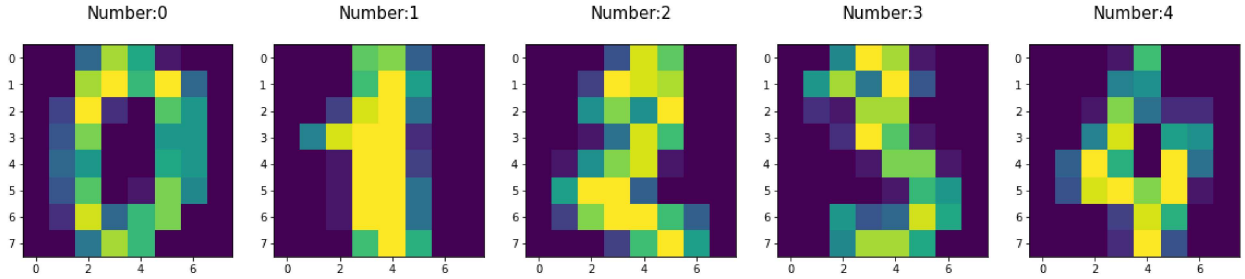
Out[45]:
```
0.999999823520647
```

In [46]:
```
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 [47]:
```
digits = load_digits()
digits
```

```
  'pixel_4_0',
  'pixel_4_1',
  'pixel_4_2',
  'pixel_4_3',
  'pixel_4_4',
  'pixel_4_5',
  'pixel_4_6',
  'pixel_4_7',
  'pixel_5_0',
  'pixel_5_1',
  'pixel_5_2',
  'pixel_5_3',
  'pixel_5_4',
  'pixel_5_5',
  'pixel_5_6',
  'pixel_5_7',
  'pixel_6_0',
  'pixel_6_1',
  'pixel_6_2',
  'pixel 6 3',
```

In [48]:
```
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)))
    plt.title('Number:%i\n' %label,fontsize=15)
```



In [50]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30
```

In [51]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [54]:
```
logre=LogisticRegression(max_iter=10000) # if error comes declare max_iter=10000
logre.fit(x_train,y_train)
```

Out[54]:
```
LogisticRegression(max_iter=10000)
```

In [55]:
```
print(logre.predict(x_test))
```

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

In [56]:
```
print(logre.score(x_test,y_test))
```

```
0.9518518518518518
```

In [ ]:
```
```