In [1]:
```
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.preprocessing import StandardScaler
```

In [ ]:

In [2]:
```
df_train=pd.read_csv(r"E:\154\C8_loan-train - C8_loan-train.csv")
df_test=pd.read_csv(r"E:\154\C8_loan-test - C8_loan-test.csv")
```

In [3]:
```
df_train.dropna(inplace=True)
df_test.dropna(inplace=True)
```

In [4]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 480 entries, 1 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            480 non-null    object
 1   Gender             480 non-null    object
 2   Married            480 non-null    object
 3   Dependents         480 non-null    object
 4   Education          480 non-null    object
 5   Self_Employed      480 non-null    object
 6   ApplicantIncome    480 non-null    int64
 7   CoapplicantIncome  480 non-null    float64
 8   LoanAmount         480 non-null    float64
 9   Loan_Amount_Term   480 non-null    float64
 10  Credit_History     480 non-null    float64
 11  Property_Area      480 non-null    object
 12  Loan_Status        480 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 52.5+ KB
```

In [5]:
```
feature_matrix = df_train[['ApplicantIncome','CoapplicantIncome','LoanAmount', 'Loan_Amount_Term','Credit_History
       ]]
target_vector = df_train[['Gender']]
```

In [6]:
```
fs = StandardScaler().fit_transform(feature_matrix)
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:63: DataConversionWarning: A column-vecto
r y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ra
vel().
  return f(*args, **kwargs)
```

Out[6]:
```
LogisticRegression()
```

In [7]:
```
observation = df_test[['ApplicantIncome','CoapplicantIncome','LoanAmount', 'Loan_Amount_Term','Credit_History']]
prediction = logr.predict(observation)
print(prediction)
```

```
['Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male'
 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male' 'Male']
```

In [8]:
```
logr.classes_
```

Out[8]:
```
array(['Female', 'Male'], dtype=object)
```

In [9]:
```
logr.predict_proba(observation)[0][0]
```

Out[9]:
```
0.0
```

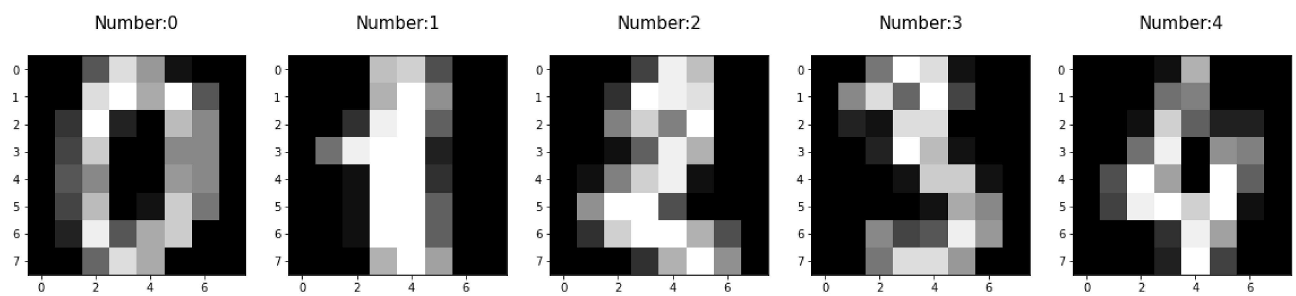
Logistic Regression 2

In [10]:
```
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 [11]:
```
digits =load_digits()
digits
```

```
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
         [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digits dataset\n-------------------------
-------------------------\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 1797\n    :Number of A
ttributes: 64\n    :Attribute Information: 8x8 image of integer pixels in the range 0..16.\n    :Missing Attri
bute Values: None\n    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a co
py of the test set of the UCI ML hand-written digits datasets\nhttps://archive.ics.uci.edu/ml/datasets/Optical
+Recognition+of+Handwritten+Digits\n\nThe data set contains images of hand-written digits: 10 classes where\ne
ach class refers to a digit.\n\nPreprocessing programs made available by NIST were used to extract\nnormalized
bitmaps of handwritten digits from a preprinted form. From a\ntotal of 43 people, 30 contributed to the traini
ng set and different 13\nto the test set. 32x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the
number of on pixels are counted in each block. This generates\nan input matrix of 8x8 where each element is an
integer in the range\n0..16. This reduces dimensionality and gives invariance to small\ndistortions.\n\nFor in
fo on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P.
J. Grother, S. A. Janet, and C.\nL. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,\n199
4.\n\n.. topic:: References\n\n  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n    A
pplications to Handwritten Digit Recognition, MSc Thesis, Institute of\n    Graduate Studies in Science and En
gineering, Bogazici University.\n  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ke
n Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityreduction using relev
```

In [12]:
```
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 [13]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [14]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [15]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[15]:
```
LogisticRegression(max_iter=10000)
```

In [16]:
```
print(logre.predict(x_test))
```

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

In [17]:
```
print(logre.score(x_test,y_test))
```

```
0.9611111111111111
```