In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn
```

In [2]:
```
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
df=pd.read_csv(r"C8_loan-test")
```

In [4]:
```
df
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amou
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
3	LP001035	Male	Yes	2	Graduate	No	2340	2546	100.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	
...	...	...	...	...	...	...	...	...	...	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	113.0	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	115.0	
364	LP002980	Male	No	0	Graduate	No	3250	1993	126.0	
365	LP002986	Male	Yes	0	Graduate	No	5000	2393	158.0	
366	LP002989	Male	No	0	Graduate	Yes	9200	0	98.0	

367 rows × 12 columns

In [5]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            367 non-null    object 
 1   Gender             356 non-null    object 
 2   Married            367 non-null    object 
 3   Dependents         357 non-null    object 
 4   Education          367 non-null    object 
 5   Self_Employed      344 non-null    object 
 6   ApplicantIncome    367 non-null    int64  
 7   CoapplicantIncome  367 non-null    int64  
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object 
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

In [6]:
```
1 df.dropna(inplace=True)
```

In [7]:
```
1 df.columns
```

Out[7]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [8]:
```
1 df
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_An
0	LP001015	Male	Yes	0	Graduate	No	5720	0	110.0	
1	LP001022	Male	Yes	1	Graduate	No	3076	1500	126.0	
2	LP001031	Male	Yes	2	Graduate	No	5000	1800	208.0	
4	LP001051	Male	No	0	Not Graduate	No	3276	0	78.0	
5	LP001054	Male	Yes	0	Not Graduate	Yes	2165	3422	152.0	
...	...	...	...	...	...	...	...	...	...	
361	LP002969	Male	Yes	1	Graduate	No	2269	2167	99.0	
362	LP002971	Male	Yes	3+	Not Graduate	Yes	4009	1777	113.0	
363	LP002975	Male	Yes	0	Graduate	No	4158	709	115.0	

In [9]:
```
1 from sklearn.linear_model import LogisticRegression
```

In [10]:
```
1 logr =LogisticRegression()
```

In [11]:
```
1 feature_matrix=df[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
2        'Loan_Amount_Term', 'Credit_History']]
3 target_vector=df['Married']
```

In [14]:
```
1 feature_matrix.shape
```

Out[14]: (289, 5)

In [15]:
```
1 target_vector.shape
```

Out[15]: (289,)

In [16]:
```
from sklearn.preprocessing import StandardScaler
```

In [17]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [18]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

Out[18]:
```
LogisticRegression()
```

In [20]:
```
observation=[[1,2,3,4,5]]
```

In [21]:
```
prediction = logr.predict(observation)
print(prediction)
```

```
['Yes']
```

In [22]:
```
logr.classes_
```

Out[22]:
```
array(['No', 'Yes'], dtype=object)
```

In [23]:
```
logr.predict_proba(observation)[0][1]
```

Out[23]:
```
0.9680271507223828
```

In [24]:
```
logr.predict_proba(observation)[0][0]
```

Out[24]:
```
0.03197284927761723
```

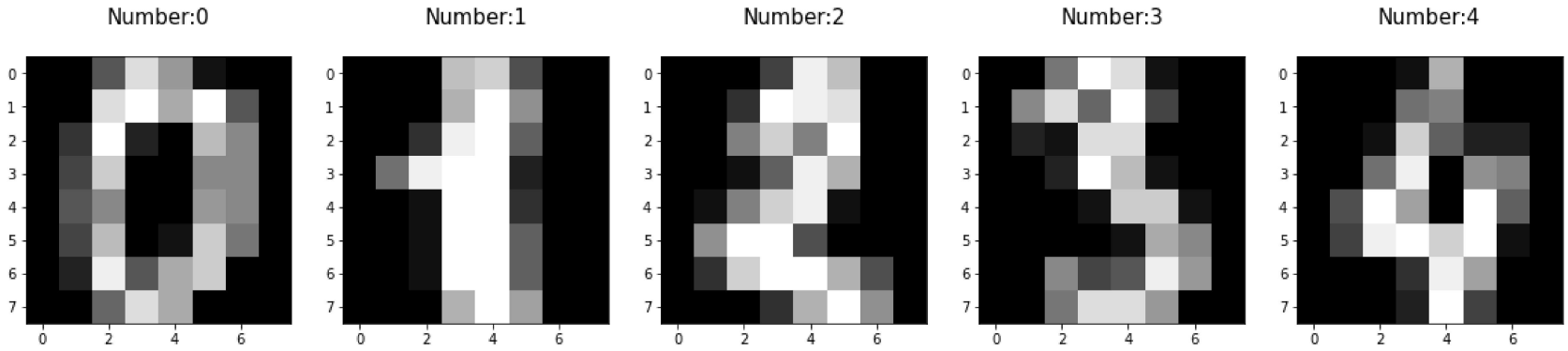
Linear regression 2

In [25]:
```
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 [26]:
```
digits =load_digits()
digits
```

Out[26]:
```
{'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 [27]:
```
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 [28]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30)
```

In [29]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [30]:
```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[30]:
```
LogisticRegression(max_iter=10000)
```

In [31]:
```
print(logre.predict(x_test))
```

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

In [32]:
```
print(logre.score(x_test,y_test))
```

```
0.9685185185185186
```