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

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

In [3]:
```
df=pd.read_csv(r"C:\Users\user\Downloads\bot.csv")
df
```

2	779715	roberttran	whose quickly especially foot none to g...	6	2	4363	True	0	Harrisonfurt	0(
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Martinezberg	2(
4	704441	noah87	Animal sign six data good or.	26	3	8438	False	1	Camachoville	2(
...	...	...	...	...	...	...	...	...	...	
49995	491196	uberg	Want but put card direction know miss former h	64	0	9911	True	1	Lake Kimberlyburgh	1(

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   User ID         50000 non-null  int64 
 1   Username        50000 non-null  object
 2   Tweet           50000 non-null  object
 3   Retweet Count   50000 non-null  int64 
 4   Mention Count   50000 non-null  int64 
 5   Follower Count  50000 non-null  int64 
 6   Verified        50000 non-null  bool  
 7   Bot Label       50000 non-null  int64 
 8   Location        50000 non-null  object
 9   Created At      50000 non-null  object
 10  Hashtags        41659 non-null  object
dtypes: bool(1), int64(5), object(5)
memory usage: 3.9+ MB
```

In [26]:
```
df1=df[['Retweet Count','Mention Count','Follower Count','Bot Label']][0:50]
```

In [27]:
```
df1.fillna(value=1)
```

Out[27]:

	Retweet Count	Mention Count	Follower Count	Bot Label
0	85	1	2353	1
1	55	5	9617	0
2	6	2	4363	0
3	54	5	2242	1
4	26	3	8438	1
5	41	4	3792	1
6	54	0	10	0
7	64	0	1442	1
8	25	2	836	0
9	67	3	6523	1
10	57	4	8694	1
11	29	1	5986	1
12	60	2	6779	0
13	61	0	6073	0
14	21	2	4846	0
15	78	1	2342	0
16	64	5	6947	1
17	43	4	7945	0
18	39	0	8305	1
19	8	2	1256	0
20	26	4	653	1
21	84	4	5466	0
22	86	1	5131	1
23	55	0	5278	1
24	56	1	3347	1
25	43	2	4851	0
26	49	5	7600	1
27	7	0	706	1
28	75	1	7313	1
29	39	0	337	1
30	77	1	7006	0
31	40	2	697	0
32	25	5	4517	0
33	15	2	1785	0
34	85	0	4057	1
35	13	0	7925	0
36	63	4	2804	1
37	75	4	3544	1

	Retweet Count	Mention Count	Follower Count	Bot Label
38	58	1	2063	0
39	34	5	466	0
40	66	1	2852	1
41	18	0	3782	0
42	0	3	4581	0
43	21	3	6979	0
44	7	3	7523	0
45	39	0	3755	0
46	34	4	6933	0
47	72	0	8386	1
48	24	5	4096	1
49	77	0	7967	1

In [52]:
```
feature_matrix = df1.iloc[:,0:3]
target_vector = df1.iloc[:,-1]
```

In [53]:
```
feature_matrix.shape
```

Out[53]: (50, 3)

In [54]:
```
target_vector.shape
```

Out[54]: (50,)

In [55]:
```
from sklearn.preprocessing import StandardScaler
```

In [56]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [57]:
```
logr=LogisticRegression()
```

In [58]:
```
logr.fit(fs,target_vector)
```

Out[58]: LogisticRegression()

In [59]:
```
observation=[[3,4,4]]
```

In [60]:
```
prediction = logr.predict(observation)
print(prediction)
```

```
[1]
```

In [61]:
```
logr.classes_
```

Out[61]: array([0, 1], dtype=int64)

In [62]:
```
logr.predict_proba(observation)[0][0]
```

Out[62]: 0.08362498004517882

In [63]:
```
logr.predict_proba(observation)[0][1]
```

Out[63]:
```
0.9163750199548212
```

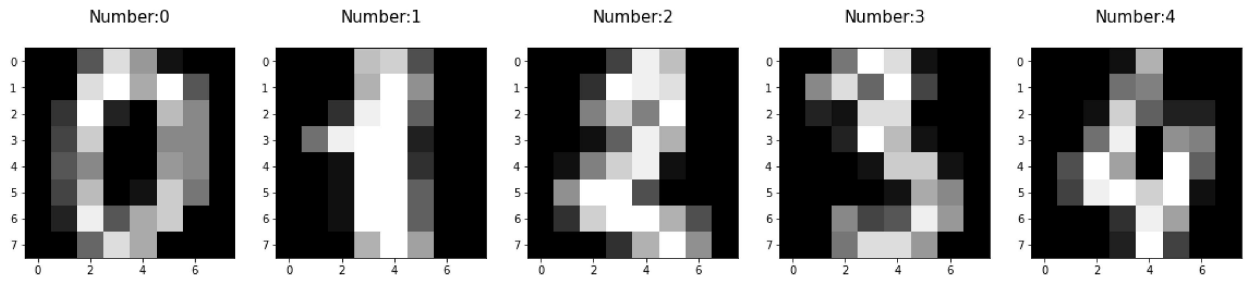
Logistic Regression-2

In [64]:
```
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 [65]:
```
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 Attributes: 64\n    :Attribute I
nformation: 8x8 image of integer pixels in the range 0..16.\n    :Missing Attribute
Values: None\n    :Creator: 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\nh
ttps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\n
The data set contains images of hand-written digits: 10 classes where\neach class re
fers to a digit.\n\nPreprocessing programs made available by NIST were used to extra
ct\nnormalized bitmaps of handwritten digits from a preprinted form. From a\ntotal o
f 43 people, 30 contributed to the training set and different 13\nto the test set. 3
2x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the number of on pix
els are counted in each block. This generates\nan input matrix of 8x8 where each ele
ment is an integer in the range\n0..16. This reduces dimensionality and gives invari
ance to small\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D. Ga
rris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Jane
t, and C.\nL. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,\n19
```

In [66]:
```
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 [67]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30
```

In [68]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [69]:
```
logr=LogisticRegression(max_iter=10000)
```

In [70]:
```
logr.fit(x_train,y_train)
```

Out[70]:
```
LogisticRegression(max_iter=10000)
```

In [71]:
```
print(logr.predict(x_test))
```

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

In [72]:
```
print(logr.score(x_test,y_test))
```

```
0.9685185185185186
```

In [ ]:

In [ ]: