LogisticRegression1

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:

```
df=pd.read_csv(r"C:\Users\user\Downloads\C3_bot_detection_data - C3_bot_detectior
df
```

Out[2]:

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	Loca
0	132131	flong	Station activity person against natural majori...	85	1	2353	False	1	Adkin
1	289683	hinesstephanie	Authority research natural life material staff...	55	5	9617	True	0	Sander
2	779715	roberttran	Manage whose quickly especially foot none to g...	6	2	4363	True	0	Harriso
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Martinezl
4	704441	noah87	Animal sign six data good or.	26	3	8438	False	1	Camachc
...	...	...	...	...	...	...	...	...	
49995	491196	uberg	Want but put card direction know miss former h...	64	0	9911	True	1	L Kimberlyb
49996	739297	jessicamunoz	Provide whole maybe agree church respond most ...	18	5	9900	False	1	Greenl
49997	674475	lynncunningham	Bring different everyone international capital...	43	3	6313	True	1	Deboral
49998	167081	richardthompson	Than about single generation itself seek sell ...	45	1	6343	False	0	Stephen

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	Loca
49999	311204	daniel29	Here morning class various room human true bec...	91	4	4006	False	0	Novakl

50000 rows × 11 columns

In [3]:

```
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 [5]:
```
df1=df.fillna(0)
df1
```

Out[5]:

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	Loca
0	132131	flong	Station activity person against natural majori...	85	1	2353	False	1	Adkin
1	289683	hinesstephanie	Authority research natural life material staff...	55	5	9617	True	0	Sander
2	779715	roberttran	Manage whose quickly especially foot none to g...	6	2	4363	True	0	Harriso
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Martinezl
4	704441	noah87	Animal sign six data good or.	26	3	8438	False	1	Camachc
...	...	...	...	...	...	...	...	...	
49995	491196	uberg	Want but put card direction know miss former h...	64	0	9911	True	1	L Kimberlyb
49996	739297	jessicamunoz	Provide whole maybe agree church respond most ...	18	5	9900	False	1	Greenl
49997	674475	lynncunningham	Bring different everyone international capital...	43	3	6313	True	1	Deboral
49998	167081	richardthompson	Than about single generation itself seek sell ...	45	1	6343	False	0	Stephen

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	Loca
49999	311204	daniel29	Here morning class various room human true bec...	91	4	4006	False	0	Novakl

50000 rows × 11 columns

In [6]:
```
df1.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        50000 non-null  object
dtypes: bool(1), int64(5), object(5)
memory usage: 3.9+ MB
```

In [16]:
```
data=df1[['User ID','Retweet Count','Mention Count','Follower Count','Bot Label']
```

In [17]:
```
from sklearn.linear_model import LogisticRegression
```

In [18]:
```
feature_matrix = data.iloc[:,0:5]
target_vector = data.iloc[:,1]
```

In [19]:
```
feature_matrix.shape
```

Out[19]:
```
(50000, 5)
```

In [20]:
```
target_vector.shape
```

Out[20]:
```
(50000,)
```

In [21]:
```
from sklearn.preprocessing import StandardScaler
```

In [22]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [24]:
```
logr=LogisticRegression()
logr.fit(fs,target_vector)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[24]:
```
LogisticRegression()
```

In [27]:
```
observation=[[5,7,9,5,6]]
```

In [28]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[100]
```

In [29]:
```
logr.classes_
```

Out[29]:
```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100], dtype=int64)
```

In [30]:
```
logr.predict_proba(observation)[0][0]
```

Out[30]:
```
7.962997325787375e-227
```

In [31]:
```
logr.predict_proba(observation)[0][0]
```

Out[31]:
```
7.962997325787375e-227
```

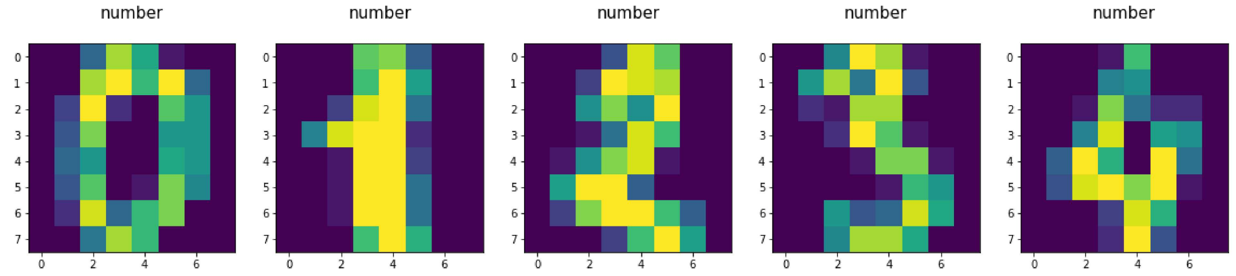
LogisticRegression2

In [32]:
```
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 [33]:
```
digits = load_digits()
digits
```

```
of hand written digits. 10 classes where\neach class refers to a digit.\n\nPr
eprocessing programs made available by NIST were used to extract\nnormalized
bitmaps of handwritten digits from a preprinted form. From a\ntotal of 43 peo
ple, 30 contributed to the training set and different 13\nto the test set. 32
x32 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 w
here each element is an integer in the range\n0..16. This reduces dimensional
ity and gives invariance to small\ndistortions.\n\nFor info on NIST preproces
sing 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 Hand
print Recognition System, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  -
C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their\n    App
lications to Handwritten Digit Recognition, MSc Thesis, Institute of\n    Gra
duate Studies in Science and Engineering, Bogazici University.\n  - E. Alpayd
in, C. Kaynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken Tang and Po
nnuthurai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityr
eduction using relevance weighted LDA. School of\n    Electrical and Electron
ic Engineering Nanyang Technological University.\n    2005.\n  - Claudio Gent
ile. A New Approximate Maximal Margin Classification\n    Algorithm. NIPS. 20
00.\n"}
```

In [37]:
```
plt.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:8])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)))
    plt.title("number\n"%label,fontsize=15)
```



In [38]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_siz
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [39]:
```
logre=LogisticRegression()
logre.fit(x_train,y_train)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:76
3: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  n_iter_i = _check_optimize_result(
```

Out[39]:
```
LogisticRegression()
```

In [40]:
```
print(logre.predict(x_test))
```

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

In [41]:
```
print(logre.score(x_test,y_test))
```

```
0.9685185185185186
```