In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Logistic Regression

In [2]:

```
from sklearn.linear_model import LogisticRegression
```

In [3]:

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

Out[3]:

	User ID	Username	Tweet	Retweet Count	Mention Count	Follower Count	Verified	Bot Label	
0	132131	flong	Station activity person against natural majori...	85	1	2353	False	1	
1	289683	hinesstephanie	Authority research natural life material staff...	55	5	9617	True	0	S
2	779715	roberttran	Manage whose quickly especially foot none to g...	6	2	4363	True	0	H
3	696168	pmason	Just cover eight opportunity strong policy which.	54	5	2242	True	1	Ma
4	704441	noah87	Animal sign six data good or.	26	3	8438	False	1	Ca
...	...	...	...	...	...	...	...	...	
49995	491196	uberg	Want but put card direction know miss former h...	64	0	9911	True	1	Kiml
49996	739297	jessicamunoz	Provide whole maybe agree church respond most ...	18	5	9900	False	1	(
49997	674475	lynncunningham	Bring different everyone international capital...	43	3	6313	True	1	D
49998	167081	richardthompson	Than about single generation itself seek sell ...	45	1	6343	False	0	St
49999	311204	daniel29	Here morning class various room human true bec...	91	4	4006	False	0	N

50000 rows × 11 columns

In [4]:

```
df.columns
```

Out[4]:

```
Index(['User ID', 'Username', 'Tweet', 'Retweet Count', 'Mention Count',
       'Follower Count', 'Verified', 'Bot Label', 'Location', 'Created A
t',
       'Hashtags'],
      dtype='object')
```

In [5]:

```
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 [6]:

```
feature_matrix=df[['Verified','User ID','Retweet Count','Mention Count','Follower Count',
target_vector = df[['Verified']]
```

In [7]:

```
feature_matrix.shape
```

Out[7]:

```
(50000, 6)
```

In [8]:

```
target_vector.shape
```

Out[8]:

```
(50000, 1)
```

In [9]:

```
from sklearn.preprocessing import StandardScaler
```

In [10]:

```
fs = StandardScaler().fit_transform(feature_matrix)
```

In [11]:

```
logr = LogisticRegression()
logr.fit(fs,target_vector)
```

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

Out[11]:

```
LogisticRegression()
```

In [12]:

```
observation=[[1.4,2,3,4,5,6]]
# Take Random Values
```

In [13]:

```
prediction = logr.predict(observation)
print(prediction)
```

```
[ True]
```

In [14]:

```
logr.classes_
```

Out[14]:

```
array([False,  True])
```

In [15]:

```
logr.predict_proba(observation)[0][0]
```

Out[15]:

```
5.222045858858642e-06
```

In [16]:

```
logr.predict_proba(observation)[0][1]
```

Out[16]:

```
0.9999947779954141
```

Logistic Regression-2

In [17]:

```
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
```

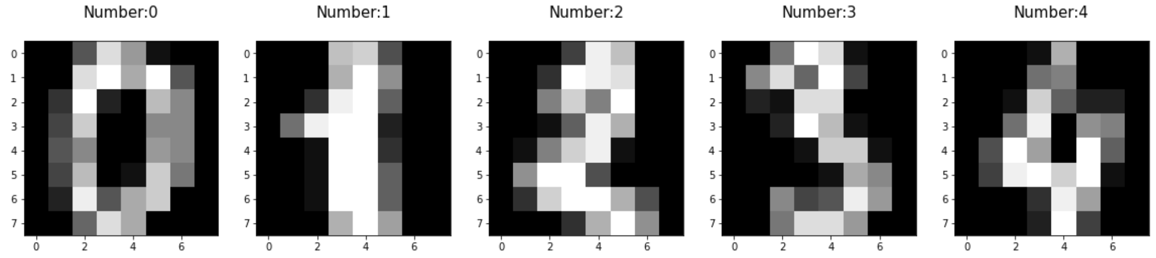
In [18]:

```
digits = load_digits()
digits
```

```
         ...,
         [ 0.,  4., 16., ..., 16.,  6.,  0.],
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
         [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten dig
its dataset\n--------------------------------------------------\n\n**Dat
a Set Characteristics:**\n\n    :Number of Instances: 1797\n    :Number 
of Attributes: 64\n    :Attribute Information: 8x8 image of integer pixe
ls in the range 0..16.\n    :Missing Attribute Values: None\n    :Creato
r: 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\nht
tps://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten
+Digits\n\nThe data set contains images of hand-written digits: 10 class
es where\neach class refers to a digit.\n\nPreprocessing programs made a
vailable by NIST were used to extract\nnormalized bitmaps of handwritten
digits from a preprinted form. From a\ntotal of 43 people, 30 contribute
d to the training set and different 13\nto the test set. 32x32 bitmaps a
re divided into nonoverlapping blocks of\n4x4 and the number of on pixel
s are counted in each block. This generates\nan input matrix of 8x8 wher
e each element is an integer in the range\n0..16. This reduces dimension
```

In [19]:

```
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 [20]:

```
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.30
```

In [21]:

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [22]:

```
logre=LogisticRegression(max_iter=10000)
logre.fit(x_train,y_train)
```

Out[22]:

```
LogisticRegression(max_iter=10000)
```

In [23]:

```
print(logre.predict(x_test))
```

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

In [24]:

```
print(logre.score(x_test,y_test))
```

```
0.9685185185185186
```