In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LogisticRegression
```

In [16]:
```
df=pd.read_csv("C3_bot_detection_data - C3_bot_detection_data.csv")
df
```

Out[16]:

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

50000 rows × 11 columns

In [17]:
```
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 [18]:
```
df1=df.fillna(value=0)
df1
```

Out[18]:

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

50000 rows × 11 columns

In [25]:
```
df1.columns
```

Out[25]:
```
Index(['User ID', 'Username', 'Tweet', 'Retweet Count', 'Mention Count',
       'Follower Count', 'Verified', 'Bot Label', 'Location', 'Created At',
       'Hashtags'],
      dtype='object')
```

In [26]:
```
df1=df1[['User ID','Retweet Count', 'Mention Count',
       'Follower Count', 'Verified', 'Bot Label']]
df1
```

Out[26]:

	User ID	Retweet Count	Mention Count	Follower Count	Verified	Bot Label
0	132131	85	1	2353	False	1
1	289683	55	5	9617	True	0
2	779715	6	2	4363	True	0
3	696168	54	5	2242	True	1
4	704441	26	3	8438	False	1
...	...	...	...	...	...	...
49995	491196	64	0	9911	True	1
49996	739297	18	5	9900	False	1
49997	674475	43	3	6313	True	1
49998	167081	45	1	6343	False	0
49999	311204	91	4	4006	False	0

50000 rows × 6 columns

In [29]:
```
feature_matrix=df1.iloc[:,0:10]
target_vector=df1.iloc[:,-1]
```

In [30]:
```
feature_matrix.shape
```

Out[30]:
```
(50000, 6)
```

In [31]:
```
target_vector.shape
```

Out[31]:
```
(50000,)
```

In [32]:
```
from sklearn.preprocessing import StandardScaler
```

In [33]:
```
fs=StandardScaler().fit_transform(feature_matrix)
```

In [34]:
```
logr =LogisticRegression()
logr.fit(fs,target_vector)
```

Out[34]:
```
LogisticRegression()
```

In [37]:
```
observation=[[1.4,2.3,5.0,11,12,13]]
```

In [38]:
```
prediction=logr.predict(observation)
print(prediction)
```

```
[1]
```

In [39]:
```
logr.classes_
```

Out[39]:
```
array([0, 1], dtype=int64)
```

In [40]:
```
logr.predict_proba(observation)[0][0]
```

Out[40]:
```
0.0
```

In [41]:
```
logr.predict_proba(observation)[0][1]
```

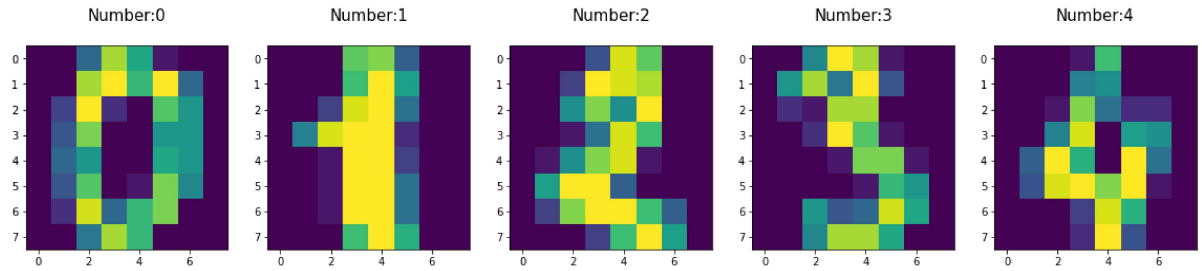
Out[41]:
```
1.0
```

In [42]:
```
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 [43]:
```
digits = load_digits()
digits
```

Out[43]:
```
{'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 [44]:
```
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)))
    plt.title('Number:%i\n' %label,fontsize=15)
```



In [47]:
```
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_s
```

In [48]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

In [49]:
```
logre=LogisticRegression(max_iter=10000) # if error comes declare max_iter=1000
logre.fit(x_train,y_train)
```

Out[49]:
```
LogisticRegression(max_iter=10000)
```

In [50]:
```
print(logre.predict(x_test))
```

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

In [51]:
```
print(logre.score(x_test,y_test))
```

```
0.9629629629629629
```

In [ ]: