In [1]:

```
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.preprocessing import StandardScaler
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
```

DataSet Bot Detection ¶

In [2]:

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

Out[2]:

	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	Cai
...	...	...	...	...	...	...	...	...	
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 [5]:

```
b=a.dropna(axis=1)
b
```

Out[5]:

	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	Cai
...	...	...	...	...	...	...	...	...	
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 × 10 columns

In [7]:

```
c=b.head(1000)
c
```

Out[7]:

	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
...	...	...	...	...	...	...	...	...	
995	902891	heather27	Rock agency course explain fear star audience ...	64	1	2821	False	1	Port
996	116097	rowetanya	Animal least opportunity green ground bring pe...	81	5	8425	False	1	Travisch
997	375161	ymartin	Cut organization member know last her center.	13	0	5665	False	0	Long
998	623097	perezanthony	Old policy democratic think people society des...	3	4	7631	False	0	Rhond
999	188036	fingram	Take provide message main carry challenge big ...	87	4	1062	True	0	S Tony

1000 rows × 10 columns

In [9]:

```
c.columns
```

Out[9]:

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

In [11]:

```
d=c[['User ID','Retweet Count', 'Mention Count',
       'Follower Count']]
e=c['Verified']
```

In [12]:

```
f=StandardScaler().fit_transform(d)
```

In [13]:

```
logr=LogisticRegression()
logr.fit(f,e)
```

Out[13]:

```
LogisticRegression()
```

In [14]:

```
g=[[23,45,67,89]]
```

In [16]:

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

```
[ True]
```

In [17]:

```
logr.classes_
```

Out[17]:

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

In [18]:

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

Out[18]:

```
0.09131157149119351
```

In [19]:

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

Out[19]:

```
0.9086884285088065
```

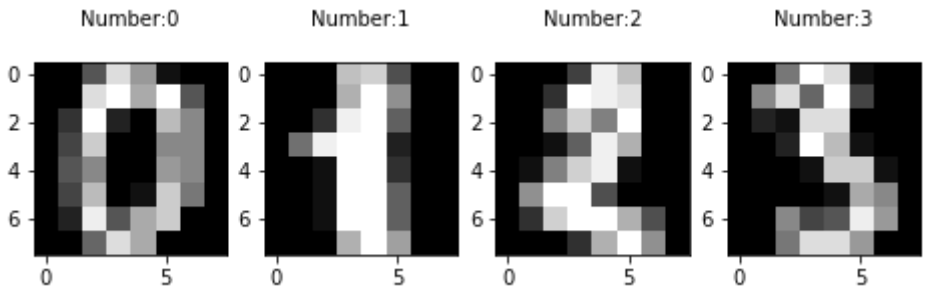
In [20]:

```
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 [21]:

```
plt.figure(figsize=(10,6))
for index,(image,label) in enumerate(zip(digits.data[0:4],digits.target[0:4])):
    plt.subplot(1,5,index+1)
    plt.imshow(np.reshape(image,(8,8)),cmap=plt.cm.gray)
    plt.title('Number:%i\n'%label,fontsize=10)
```



In [22]:

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

In [23]:

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

Out[23]:

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

In [24]:

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

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

In [25]:

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

```
0.9740740740740741
```

DataSet Framing Ham

In [26]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C4_framingham.csv")
a
```

Out[26]:

e	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes
l	39	4.0	0	0.0	0.0	0	0	(
)	46	2.0	0	0.0	0.0	0	0	(
l	48	1.0	1	20.0	0.0	0	0	(
)	61	3.0	1	30.0	0.0	0	1	(
)	46	3.0	1	23.0	0.0	0	0	(
.	...	...	...	...	...	...	...	..
l	50	1.0	1	1.0	0.0	0	1	(
l	51	3.0	1	43.0	0.0	0	0	(
)	48	2.0	1	20.0	NaN	0	0	(
)	44	1.0	1	15.0	0.0	0	0	(
)	52	2.0	0	0.0	0.0	0	0	(

× 16 columns

In [28]:

```
b=a.dropna(axis=1)
b
```

Out[28]:

	male	age	currentSmoker	prevalentStroke	prevalentHyp	diabetes	sysBP	diaBP	TenYearCHD
0	1	39	0	0	0	0	106.0	70.0	0
1	0	46	0	0	0	0	121.0	81.0	0
2	1	48	1	0	0	0	127.5	80.0	0
3	0	61	1	0	1	0	150.0	95.0	1
4	0	46	1	0	0	0	130.0	84.0	0
...	...	...	...	...	...	...	...	...	...
4233	1	50	1	0	1	0	179.0	92.0	1
4234	1	51	1	0	0	0	126.5	80.0	0
4235	0	48	1	0	0	0	131.0	72.0	0
4236	0	44	1	0	0	0	126.5	87.0	0

In [29]:

```
b.columns
```

Out[29]:

```
Index(['male', 'age', 'currentSmoker', 'prevalentStroke', 'prevalentHyp',
       'diabetes', 'sysBP', 'diaBP', 'TenYearCHD'],
      dtype='object')
```

In [30]:

```
c=b.iloc[:,0:8]
d=b.iloc[:,-1]
```

In [31]:

```
e=StandardScaler().fit_transform(c)
```

In [32]:

```
logr=LogisticRegression()
logr.fit(e,d)
```

Out[32]:

```
LogisticRegression()
```

In [33]:

```
f=[[12,34,56,78,90,32,54,57]]
```

In [34]:

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

```
[1]
```

In [35]:

```
logr.classes_
```

Out[35]:

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

In [36]:

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

Out[36]:

```
0.0
```

In [37]:

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

Out[37]:

```
1.0
```

DataSet Health Care

In [38]:

```
a=pd.read_csv(r"C:\Users\user\Downloads\C5_health care diabetes.csv")
a
```

Out[38]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

In [39]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [40]:

```
b=a.iloc[:,0:8]
c=a.iloc[:,-1]
```

In [41]:

```
d=StandardScaler().fit_transform(b)
```

In [42]:

```
logr=LogisticRegression()
logr.fit(d,c)
```

Out[42]:

```
LogisticRegression()
```

In [43]:

```
e=[[10,20,30,40,50,60,70,80]]
```

In [44]:

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

```
[1]
```

In [45]:

```
logr.classes_
```

Out[45]:

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

In [46]:

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

Out[46]:

```
0.0
```

In [47]:

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

Out[47]:

```
1.0
```

In [ ]:

```

```