In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
```

In [4]:

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

Out[4]:

	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	I

50000 rows × 11 columns

In [43]:

```
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 [42]:

```
df.columns
```

Out[42]:

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

In [44]:

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

In [45]:

```
feature_matrix.shape
```

Out[45]:

```
(50000, 6)
```

In [46]:

```
target_vector.shape
```

Out[46]:

```
(50000, 1)
```

In [47]:

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

In [48]:

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

In [49]:

```
log = LogisticRegression()
log.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[49]:

```
LogisticRegression()
```

In [53]:

```
observations = [[1,2,3,4,5,6]]
```

In [54]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[ True]
```

In [55]:

```
log.classes_
```

Out[55]:

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

In [56]:

```
log.predict_proba(observations)[0][0]
```

Out[56]:

```
0.0
```

In [57]:

```
log.predict_proba(observations)[0][1]
```

Out[57]:

```
1.0
```

Logic regression 2

In [58]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [59]:

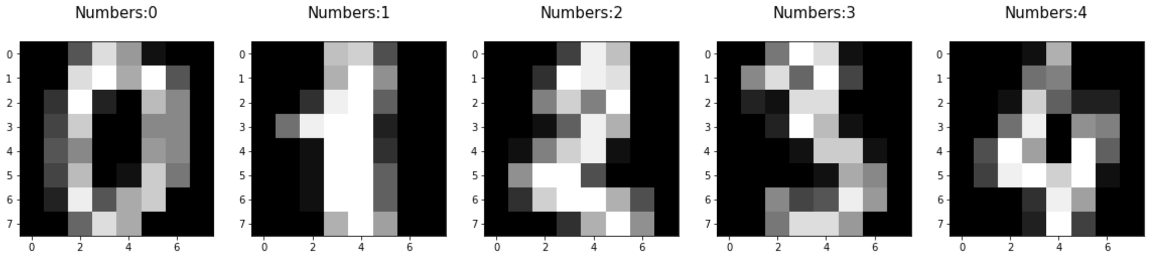
```
digits = load_digits()
digits
```

Out[59]:

```
{'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',
```

In [60]:

```
pp.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    pp.subplot(1,5,index+1)
    pp.imshow(np.reshape(image,(8,8)),cmap = pp.cm.gray)
    pp.title('Numbers:%i\n'%label,fontsize=15)
```



Numbers:0 Numbers:1 Numbers:2 Numbers:3 Numbers:4

In [61]:

```
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.3)
```

In [62]:

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

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

In [63]:

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

Out[63]:

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

In [64]:

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

```
0.9629629629629629
```

In [ ]:

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
```

In [2]:

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

Out[2]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentH
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

4238 rows × 16 columns

In [5]:

```
df.columns
```

Out[5]:

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMed
s',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4238 entries, 0 to 4237
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4238 non-null   int64  
 1   age              4238 non-null   int64  
 2   education        4133 non-null   float64
 3   currentSmoker    4238 non-null   int64  
 4   cigsPerDay       4209 non-null   float64
 5   BPMeds           4185 non-null   float64
 6   prevalentStroke  4238 non-null   int64  
 7   prevalentHyp     4238 non-null   int64  
 8   diabetes         4238 non-null   int64  
 9   totChol          4188 non-null   float64
 10  sysBP            4238 non-null   float64
 11  diaBP            4238 non-null   float64
 12  BMI              4219 non-null   float64
 13  heartRate        4237 non-null   float64
 14  glucose          3850 non-null   float64
 15  TenYearCHD       4238 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 529.9 KB
```

In [15]:

```
feature_matrix = df[['male', 'age', 'currentSmoker',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'TenYearCHD']]
target_vector = df[['currentSmoker']]
```

In [16]:

```
feature_matrix.shape
```

Out[16]:

```
(4238, 7)
```

In [17]:

```
target_vector.shape
```

Out[17]:

```
(4238, 1)
```

In [18]:

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

In [19]:

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

In [20]:

```
log = LogisticRegression()
log.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[20]:

```
LogisticRegression()
```

In [24]:

```
observations = [[1,2,3,4,5,6,7]]
```

In [25]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[1]
```

In [26]:

```
log.classes_
```

Out[26]:

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

In [27]:

```
log.predict_proba(observations)[0][0]
```

Out[27]:

```
6.348847669812585e-09
```

In [28]:

```
log.predict_proba(observations)[0][1]
```

Out[28]:

```
0.9999999936511523
```

Logic regression 2

In [29]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [30]:

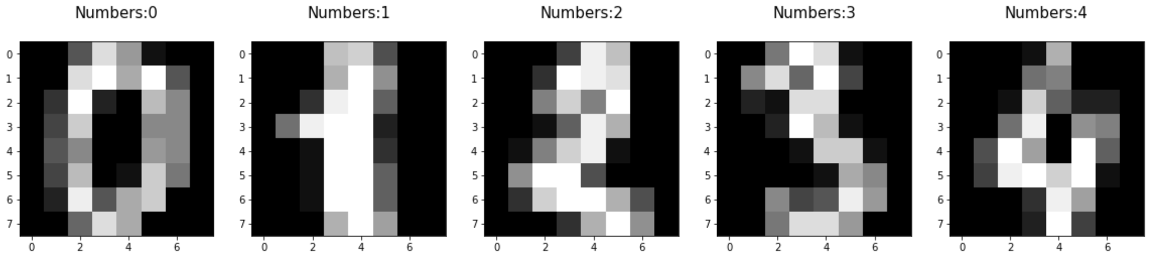
```
digits = load_digits()
digits
```

Out[30]:

```
{'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',
```

In [31]:

```
pp.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    pp.subplot(1,5,index+1)
    pp.imshow(np.reshape(image,(8,8)),cmap = pp.cm.gray)
    pp.title('Numbers:%i\n'%label,fontsize=15)
```



In [32]:

```
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.3)
```

In [33]:

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

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

In [34]:

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

Out[34]:

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

In [35]:

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

```
0.9574074074074074
```

In [ ]:

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as pp
from sklearn.linear_model import LogisticRegression
```

In [3]:

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

Out[3]:

	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 [4]:

```
df.columns
```

Out[4]:

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insuli
n',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [5]:

```
df.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 [6]:

```
feature_matrix = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulir
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']]
target_vector = df[['Age']]
```

In [7]:

```
feature_matrix.shape
```

Out[7]:

```
(768, 9)
```

In [8]:

```
target_vector.shape
```

Out[8]:

```
(768, 1)
```

In [9]:

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

In [10]:

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

In [11]:

```
log = LogisticRegression()
log.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 [13]:

```
observations = [[1,2,3,4,5,6,7,8,9]]
```

In [14]:

```
prediction = log.predict(observations)
print(prediction)
```

```
[52]
```

In [15]:

```
log.classes_
```

Out[15]:

```
array([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, 72,
       81], dtype=int64)
```

In [16]:

```
log.predict_proba(observations)[0][0]
```

Out[16]:

```
1.6536768855247902e-42
```

In [17]:

```
log.predict_proba(observations)[0][1]
```

Out[17]:

```
7.414864970104293e-36
```

Logic regression 2

In [18]:

```
import re
from sklearn.datasets import load_digits
import numpy as np
import pandas as pd
import matplotlib.pyplot as pp
import seaborn as sb
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [19]:

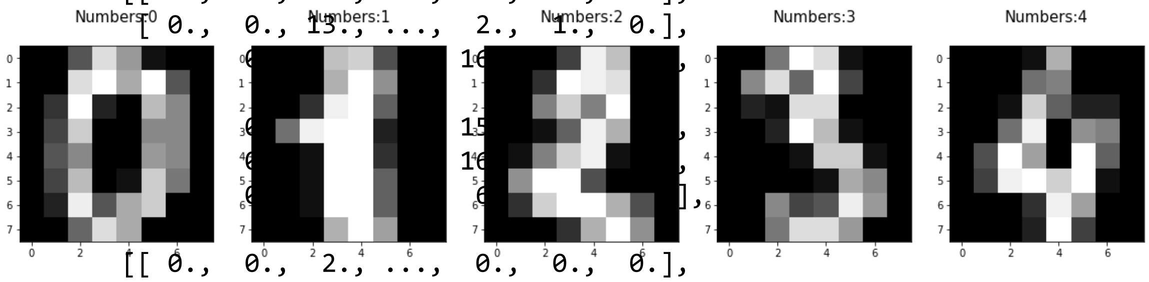
```
digits = load_digits()
digits
```

Out[19]:

```
{'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',
  'pixel_1_2',
  'pixel_1_3',
  'pixel_1_4',
  'pixel_1_5',
  'pixel_1_6',
  'pixel_1_7',
  'pixel_2_0',
  'pixel_2_1',
  'pixel_2_2',
  'pixel_2_3',
  'pixel_2_4',
  'pixel_2_5',
  'pixel_2_6',
  'pixel_2_7',
  'pixel_3_0',
  'pixel_3_1',
  'pixel_3_2',
  'pixel_3_3',
  'pixel_3_4',
  'pixel_3_5',
  'pixel_3_6',
  'pixel_3_7',
  'pixel_4_0',
  'pixel_4_1',
  'pixel_4_2',
  'pixel_4_3',
  'pixel_4_4',
  'pixel_4_5',
  'pixel_4_6',
  'pixel_4_7',
  'pixel_5_0',
  'pixel_5_1',
  'pixel_5_2',
  'pixel_5_3',
  'pixel_5_4',
  'pixel_5_5',
  'pixel_5_6',
  'pixel_5_7',
  'pixel_6_0',
  'pixel_6_1',
  'pixel_6_2',
  'pixel_6_3',
  'pixel_6_4',
  'pixel_6_5',
  'pixel_6_6',
  'pixel_6_7',
  'pixel_7_0',
  'pixel_7_1',
  'pixel_7_2',
  'pixel_7_3',
  'pixel_7_4',
  'pixel_7_5',
  'pixel_7_6',
  'pixel_7_7'],
 'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ..., 15.,  5.,  0.],
         [ 0.,  3., 15., ..., 11.,  8.,  0.],
         ...,
         [ 0.,  4., 11., ..., 12.,  7.,  0.],
         [ 0.,  2., 14., ..., 12.,  0.,  0.],
         [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

        [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
         [ 0.,  0.,  0., ...,  9.,  0.,  0.],
         [ 0.,  0.,  3., ...,  6.,  0.,  0.],
         ...,
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  1., ...,  6.,  0.,  0.],
         [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

        [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
         [ 0.,  0.,  3., ..., 14.,  0.,  0.],
         [ 0.,  0.,  8., ..., 16.,  0.,  0.],
         ...,
         [ 0.,  9., 16., ...,  0.,  0.,  0.],
         [ 0.,  3., 13., ..., 11.,  5.,  0.],
         [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

        [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
         [ 0.,  0., 13., ...,  2.,  1.,  0.],
         [ 0.,  0., [illegible] 16., [illegible],
         ...,
         [ 0.,  0., [illegible] 15., [illegible],
         [ 0.,  0., [illegible] 16., [illegible],
         [ 0.,  0., [illegible]  6., [illegible]]],

        [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
         [ 0.,  0., 14., ..., 15.,  1.,  0.],
         [ 0.,  4., 16., ..., 16.,  7.,  0.],
         ...,
         [ 0.,  0.,  0., ..., 16.,  2.,  0.],
         [ 0.,  0.,  4., ..., 16.,  2.,  0.],
         [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

        [[ 0.,  0., 10., ...,  1.,  0.,  0.],
         [ 0.,  2., 16., ...,  1.,  0.,  0.],
         [ 0.,  0., 15., ..., 15.,  0.,  0.],
         ...,
         [ 0.,  4., 16., ..., 16.,  6.,  0.],
         [ 0.,  8., 16., ..., 16.,  8.,  0.],
```

In [20]:

```
pp.figure(figsize=(20,4))
for index,(image,label) in enumerate(zip(digits.data[0:5],digits.target[0:5])):
    pp.subplot(1,5,index+1)
    pp.imshow(np.reshape(image,(8,8)),cmap = pp.cm.gray)
    pp.title('Numbers:%i\n'%label,fontsize=15)
```



In [21]:

```
x_train,x_test,y_train,y_test = train_test_split(digits.data,digits.target,test_size=0.3)
```

```
       [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
 'DESCR': ".. _digits_dataset:\n\nOptical recognition of handwritten digit
s dataset\n--------------------------------------------------\n\n**Data Se
t Characteristics:**\n\n    :Number of Instances: 1797\n    :Number of Att
ributes: 64\n    :Attribute Information: 8x8 image of integer pixels in th
e range 0..16.\n    :Missing Attribute Values: None\n    :Creator: E. Alpa
ydin (alpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a copy o
f the test set of the UCI ML hand-written digits datasets\nhttps://archiv
e.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits\n\nThe
data set contains images of hand-written digits: 10 classes where\neach cl
ass refers to a digit.\n\nPreprocessing programs made available by NIST we
re used to extract\nnormalized bitmaps of handwritten digits from a prepri
nted form. From a\ntotal of 43 people, 30 contributed to the training set
and different 13\nto the test set. 32x32 bitmaps are divided into nonoverl
apping blocks of\n4x4 and the number of on pixels are counted in each bloc
k. This generates\nan input matrix of 8x8 where each element is an integer
in the range\n0..16. This reduces dimensionality and gives invariance to s
mall\ndistortions.\n\nFor info on NIST preprocessing routines, see M. D. G
arris, J. L. Blue, G.\nT. Candela, D. L. Dimmick, J. Geist, P. J. Grother,
S. A. Janet, and C. L. Wilson, NIST Form-Based Handprint Recognition Syst
em, NISTIR 5469,\n1994.\n\n.. topic:: References\n\n  - C. Kaynak (1995) M
ethods of Combining Multiple Classifiers and Their\n    Applications to Ha
ndwritten Digit Recognition, MSc Thesis, Institute of\n    Graduate Studie
s in Science and Engineering, Bogazici University.\n  - E. Alpaydin, C. Ka
ynak (1998) Cascading Classifiers, Kybernetika.\n  - Ken Tang and Ponnuthu
rai N. Suganthan and Xi Yao and A. Kai Qin.\n    Linear dimensionalityredu
ction using relevance weighted LDA. School of\n    Electrical and Electron
ic Engineering Nanyang Technological University.\n    2005.\n  - Claudio G
entile. A New Approximate Maximal Margin Classification\n    Algorithm. NI
PS. 2000.\n"}
```

In [22]:

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

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

In [23]:

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

Out[23]:

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

In [24]:

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

```
0.9518518518518
```

In [ ]: