In [2]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
df=pd.read_csv(r"C:\Users\91756\Downloads\ionosphere_data (1).csv")
df
```

Out[4]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	...	column_z	column_aa	column_ab	columr
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	...	-0.51171	0.41078	-0.46168	0.2
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0.18401	-0.1
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0.22145	0.4
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1.00000	1.0
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0.53206	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
346	True	False	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0.00123	1.0
347	True	False	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0.04925	0.9
348	True	False	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0.02542	0.9
349	True	False	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0.07760	0.8
350	True	False	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0.04822	0.7

351 rows × 35 columns

In [5]:
```
pd.set_option('display.max_rows',100000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [6]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

```
This DataFrame has 351 Rows and 35 columns
```

In [7]:
```
df.head()
```

Out[7]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_k	column_l	column_m	column_n	col
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	-0.17755	0.59755	-0.44945	0
1	True	False	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432	-0.69707	-0
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443	0.00827	0
3	True	False	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	-1
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409	-0.00712	0

In [8]:
```
features_matrix = df.iloc[:,0:34]
```

In [10]:
```
target_vector = df.iloc[:,-1]
```

In [11]:
```
print('The Features Matrix Has %d Rows And %d Column(S)'%(features_matrix.shape))
print('The Target Matrix Has %d Rows and %d Column(s)'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Features Matrix Has 351 Rows And 34 Column(S)
The Target Matrix Has 351 Rows and 1 Column(s)
```

In [12]:
```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [13]:
```
sticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,class_weight=None,random_state=None)
```

In [14]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [15]:
```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999999,
 -0.37708,1.0,0.0376,0.8524299999999999,-0.17755,0.59755,-0.44945,
 0.60536,-0.38223,0.8435600000000001,-0.38542,0.58212,-0.32192,0.56971
 ,-0.29674,0.36946,-0.47357,0.56811,-0.51171,0.41078000000000003,
 -0.46168000000000003,0.21266,-0.3409,0.42267,-0.54487,0.18641,
 -0.453]]
```

In [16]:
```
predictions = Logistic_Regression_Model.predict(observation)
print('The Model predicted The observation To Belong To class %s'%(predictions))
```

```
The Model predicted The observation To Belong To class ['g']
```

In [17]:
```
print('The Algorithm Was Trained To predict one of the two classes:%s'%(algorithm.classes_))
```

```
The Algorithm Was Trained To predict one of the two classes:['b' 'g']
```

In [18]:
```
print("""The Model says The probability of the observation We passed Belonging To class['g'] Is %s"""%(algorithm.predict_proba(ob
print()
print("""The Model says The probability of the observation We passed Belonging To class['b'] Is %s"""%(algorithm.predict_proba(ob
```

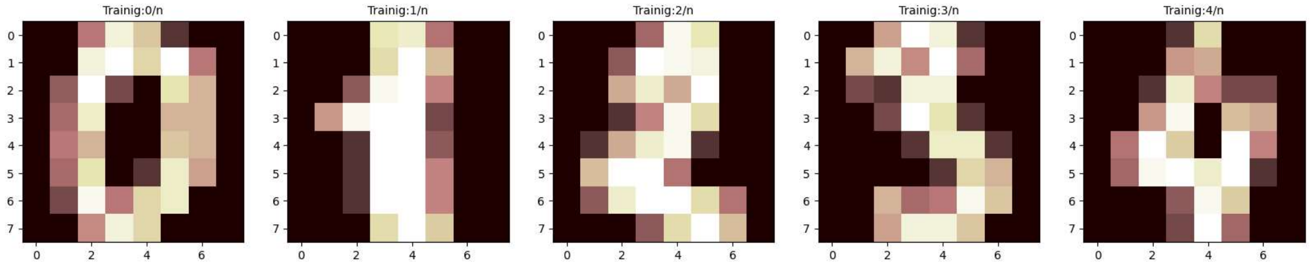
```
The Model says The probability of the observation We passed Belonging To class['g'] Is 0.007773931600137 84

The Model says The probability of the observation We passed Belonging To class['b'] Is 0.00777393160013784
```

In [5]:
```
import re
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
%matplotlib inline
digits=load_digits()
```

In [6]:
```
print("Image Data Shape",digits.data.shape)
print("Label Data Shape",digits.target.shape)
```

```
Image Data Shape (1797, 64)
Label Data Shape (1797,)
```

In [15]:
```
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.pink)
    plt.title('Trainig:%i/n'%label,fontsize=10)
```



In [18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(digits.data,digits.target,test_size=0.30,random_state=2)
```

In [19]:
```
print(x_train.shape)
```

```
(1257, 64)
```

In [20]:
```
print(y_train.shape)
```

```
(1257,)
```