In [ ]:

```
```

LogisticalRegression

In [2]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
dg=pd.read_csv(r"C:\Users\magam\Downloads\archive\ionosphere.csv")
dg
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-0.51171	0.41078	-0
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-0.26569	-0.20468	-0
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-0.40220	0.58984	-0
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	0.90695	0.51613	1
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-0.65158	0.13290	-0
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-0.01535	-0.03240	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-0.04202	0.83479	0
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	0.01361	0.93522	0
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	0.03193	0.92489	0
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-0.02099	0.89147	-0
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-0.15114	0.81147	-0

350 rows × 35 columns

In [3]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
print('This DataFrame has %d Rows and %d columns'%(dg.shape))
dg.head()
```

```
This DataFrame has 350 Rows and 35 columns
```

Out[3]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.1	-0.17755	0.59755
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.50874	-0.67743	0.34432
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	0.85443
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.00000	0.00000	0.00000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0.56409
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0.00000

In [4]:

```
features_matrix=dg.iloc[:,0:34]
target_vector=dg.iloc[:,-1]
print('The Features matrix has %d rows and %d columns'%(features_matrix.shape))
print('The Target matrix has %d rows and %d columns'%(np.array(target_vector).reshape(-1,1).shape))
```

```
The Features matrix has 350 rows and 34 columns
The Target matrix has 350 rows and 1 columns
```

In [10]:

```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=
Logistic_Regression_model=algorithm.fit(features_matrix_standardized,target_vector)
```

```
C:\Users\magam\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_mo
del\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/s
table/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (http
s://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [12]:

```
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,
-0.38542,0.58219,-0.32192,0.56971,-0.29674,0.36946,-0.47357,
0.56811,-0.51171,0.4107800000000003,-0.46168000000000003,0.21266,
-0.3409,0.42267,-0.54487,0.18641,-0.453]]
```

In [14]:

```
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 [15]:

```
print('The Algorithm Was Trained To predict The One Of The Classes: %s'%(algorithm.classes_))
```

```
The Algorithm Was Trained To predict The One Of The Classes: ['b' 'g']
```

In [18]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Class['b'] is %s""
%(algorithm.predict_proba(observation)[0][0]))
print()
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class['b'] is
5.9702836953001714e-05
```

In [19]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Class['g'] is %s "
%(algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The observation We Passed belonging To The Class['g'] is
0.999940297163047
```

In [ ]:

In [ ]:

In [12]:

```
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 sb
from sklearn import metrics
%matplotlib inline
digits=load_digits()
```

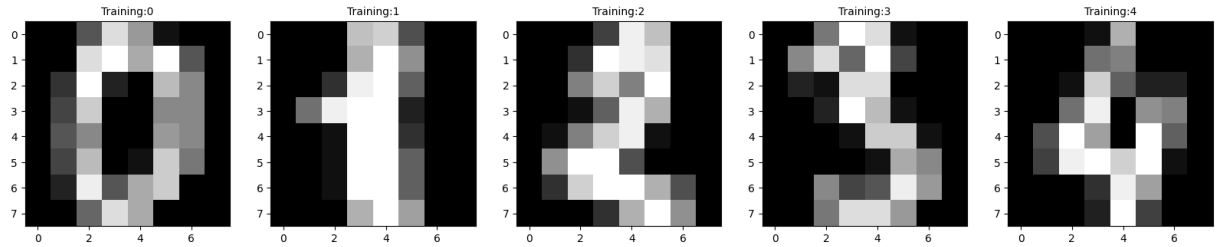
In [13]:

```
print("Image Data Shape",digits.data.shape)
print("Label data shape",digits.target.shape)
```

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

In [17]:

```
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('Training:%i'%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 [22]:

```
print(x_train.shape)
```

```
(1257, 64)
```

In [23]:

```
print(y_train.shape)
```

```
(1257,)
```

In [24]:

```
print(x_test.shape)
```

```
(540, 64)
```

In [25]:

```
print(y_test.shape)
```

```
(540,)
```

In [28]:

```
from sklearn.linear_model import LogisticRegression
LogReg=LogisticRegression(max_iter=10000)
LogReg.fit(x_train,y_train)
print(LogReg.predict(x_test))
```

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

In [27]:

```
score=LogReg.score(x_test,y_test)
print(score)
```

```
0.9537037037037037
```

In [ ]:

In [ ]: