Ionosphere (1-06-23)

In [12]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
df=pd.read_csv(r"C:\Users\Teju\Downloads\ionosphere.csv")
df
```

Out[2]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	...	-(
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	...	-(
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	...	-(
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	...	(
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	...	-(
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	...	-(
...	...	...	...	...	...	...	...	...	...	...	...	
345	1	0	0.83508	0.08298	0.73739	-0.14706	0.84349	-0.05567	0.90441	-0.04622	...	-(
346	1	0	0.95113	0.00419	0.95183	-0.02723	0.93438	-0.01920	0.94590	0.01606	...	(
347	1	0	0.94701	-0.00034	0.93207	-0.03227	0.95177	-0.03431	0.95584	0.02446	...	(
348	1	0	0.90608	-0.01657	0.98122	-0.01989	0.95691	-0.03646	0.85746	0.00110	...	-(
349	1	0	0.84710	0.13533	0.73638	-0.06151	0.87873	0.08260	0.88928	-0.09139	...	-

350 rows × 35 columns

In [5]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [6]:

```
print('This DataFrame hs %d Rows and %d columns'%(df.shape))
```

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

In [7]:

```
df.head()
```

Out[7]:

	1	0	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.1	0.03760	0.85243.
0	1	0	1.00000	-0.18829	0.93035	-0.36156	-0.10868	-0.93597	1.00000	-0.04549	0.5087
1	1	0	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.7308
2	1	0	1.00000	-0.45161	1.00000	1.00000	0.71216	-1.00000	0.00000	0.00000	0.0000
3	1	0	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.5279
4	1	0	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.0378

In [15]:

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

In [16]:

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

In [17]:

```
print('The Features Matrix has %d Rows and %d columns'%(features_matrix.shape))
```

```
The Features Matrix has 350 Rows and 34 columns
```

In [18]:

```
print('The Target Matrix has %d Rows and %d columns'%(np.array(target_vector).reshape(-1
```

```
The Target Matrix has 350 Rows and 1 columns
```

In [19]:

```
features_matrix_standardized=StandardScaler().fit_transform(features_matrix)
```

In [27]:

```
algorithm=LogisticRegression(penalty=None,dual=False,tol=1e-4,C=1.0,fit_intercept=True,i
                             class_weight=None,random_state=None,solver='lbfgs',max_iter
                             multi_class='auto',verbose=0,warm_start=False,n_jobs=None,l
```

In [28]:

```
Logistic_Regression_Model = algorithm.fit(features_matrix_standardized,target_vector)
```

In [29]:

```
observation=[[1,0,0.99539,-0.05889,0.8524299999999999,0.02306,0.8339799999999999,-0.3770
```

In [30]:

```
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 [31]:

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

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

In [32]:

```
print("""The Model Says The Probability Of The observation We Passed belonging To The Cl
 %(algorithm.predict_proba(observation)[0][0]))
print()
print("""The Model Says The Probability Of The observation We Passed belonging To The Cl
 %(algorithm.predict_proba(observation)[0][1]))
```

```
The Model Says The Probability Of The observation We Passed belonging To T
he Class ['b'] is 2.0539100364147522e-05

The Model Says The Probability Of The observation We Passed belonging To T
he Class ['g'] is 0.9999794608996359
```

In [ ]:

```

```

Logistic (2-6-23)

In [36]:

```
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()
```

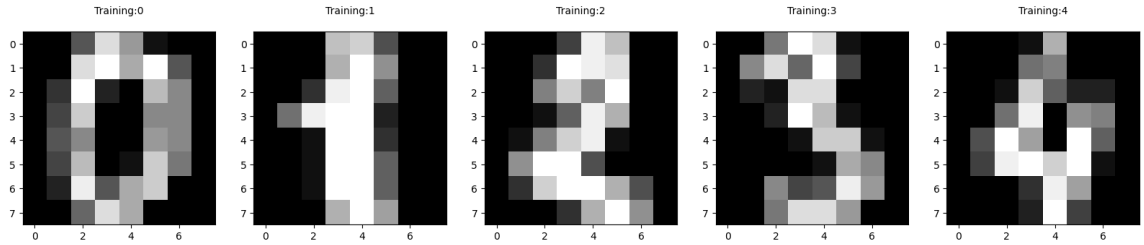
In [37]:

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

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

In [47]:

```
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\n'%label,fontsize=10)
```



In [48]:

```
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,
```

In [49]:

```
print(X_train.shape)
```

```
(1257, 64)
```

In [50]:

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

```
(1257,)
```

In [51]:

```
print(X_test.shape)
```

```
(540, 64)
```

In [52]:

```
from sklearn.linear_model import LogisticRegression
```

In [57]:

```
logisticRegr=LogisticRegression(max_iter=10000)
logisticRegr.fit(X_train,y_train)
```

Out[57]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [54]:

```
print(logisticRegr.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 1 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 [55]:

```
score=logisticRegr.score(X_test,y_test)
score
```

Out[55]:

```
0.9537037037037037
```

In [ ]:

```
`
```