Logistic Rgression Using Excel

In [29]:
```
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
```

In [30]:
```
df=pd.read_csv(r"C:\Users\yoshitha lakshmi\OneDrive\Desktop\python\ionosphere_data.csv")
df
```

Out[30]:

	column_a	column_b	column_c	column_d	column_e	column_f	column_g	column_h	column_i	column_j	column_k	column_l	colu
0	True	False	0.99539	-0.05889	0.85243	0.02306	0.83398	-0.37708	1.00000	0.03760	0.85243	-0.17755	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
2	True	False	1.00000	-0.03365	1.00000	0.00485	1.00000	-0.12062	0.88965	0.01198	0.73082	0.05346	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
4	True	False	1.00000	-0.02401	0.94140	0.06531	0.92106	-0.23255	0.77152	-0.16399	0.52798	-0.20275	0
5	True	False	0.02337	-0.00592	-0.09924	-0.11949	-0.00763	-0.11824	0.14706	0.06637	0.03786	-0.06302	0
6	True	False	0.97588	-0.10602	0.94601	-0.20800	0.92806	-0.28350	0.85996	-0.27342	0.79766	-0.47929	0
7	False	False	0.00000	0.00000	0.00000	0.00000	1.00000	-1.00000	0.00000	0.00000	-1.00000	-1.00000	0
8	True	False	0.96355	-0.07198	1.00000	-0.14333	1.00000	-0.21313	1.00000	-0.36174	0.92570	-0.43569	0
9	True	False	-0.01864	-0.08459	0.00000	0.00000	0.00000	0.00000	0.11470	-0.26810	-0.45663	-0.38172	0
10	True	False	1.00000	0.06655	1.00000	0.18388	1.00000	0.27320	1.00000	0.43107	1.00000	0.41349	0

In [31]:
```
pd.set_option('display.max_rows',100000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)

```

In [32]:
```
print('This DataFrame has %d Rows and %d columns'%(df.shape))
```

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

In [33]:
```
df.head()
```

Out[33]:

	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
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
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
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
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
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

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

In [35]:
```
target_vector = df.iloc[:,-1]
```

In [36]:
```
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 [37]:
```
features_matrix_standardized = StandardScaler().fit_transform(features_matrix)
```

In [38]:
```
algorithm = LogisticRegression(penalty='l2',dual=False,tol=1e-4,C=1.0,fit_intercept=True,intercept_scaling=1,cla
```

In [47]:
```
Logistic_Regression_Model=algorithm.fit(features_matrix_standardized,target_vector)
```

In [48]:
```
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 [49]:
```
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 [51]:
```
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 [52]:
```
print("""The Model says The probability of the observation We passed Belonging To class['b'] Is %s"""%(algorithm
print()
print("""The Model says The probability of the observation We passed Belonging To class['g'] Is %s"""%(algorithm
```

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

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

Logistic Regression

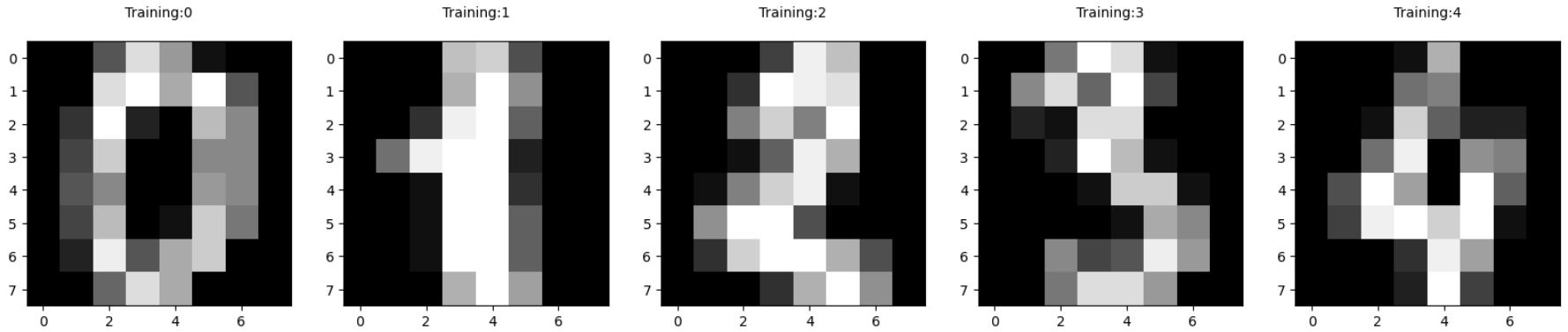
(with out using Excel for visualisation)

In [56]:
```
import re #(re=regular expression)
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 [57]:
```
print("Image Data Shape",digits.data.shape)
print("Label Data Shape",digits.target.shape)
```

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

In [66]:
```
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 [68]:
```
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 [69]:
```
print(x_train.shape)
```

```
(1257, 64)
```

In [71]:
```
print(y_train.shape)
```

```
(1257,)
```

In [72]:
```
print(x_test.shape)
```

```
(540, 64)
```

In [73]:
```
print(y_test.shape)
```

```
(540,)
```

In [74]:
```
from sklearn.linear_model import LogisticRegression
logisticRegr=LogisticRegression(max_iter=10000)
logisticRegr.fit(x_train,y_train)
```

Out[74]:
```
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 [75]:
```
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 [85]:
```
score=logisticRegr.score(x_test,y_test)
print(score)
```

```
0.9537037037037037
```

In [ ]:
```
1
```