EE559-Homework#4-Lei Lei

1.(a) Final (augmented) weight vectors:

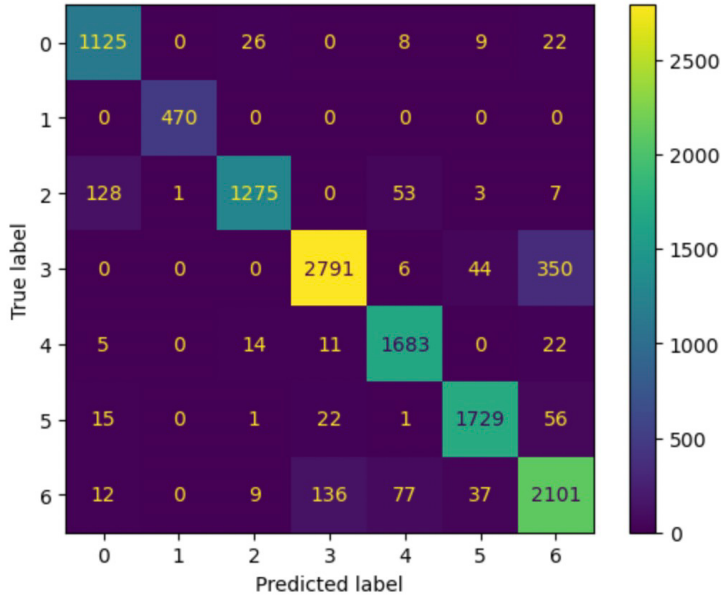
```
Final weights is : [[ 5.00000000e+00  2.12796720e+00  7.60585752e+00  1.34295771e+00
   1.19911622e+01 -6.75631691e+00 -1.00905513e-01  2.29432636e+00
   6.88080450e+00 -1.67181508e+00 -4.51363017e+00 -6.92410801e+00
   6.26528919e+00 -1.32294487e+01 -8.50954425e+00  5.28559780e+00
   1.25934303e+01]
  [-2.10000000e+01  2.28191891e+01  8.81236711e+00  8.42526643e+00
   1.09445976e+01 -2.58056370e+00 -4.59697369e+00  2.29915094e+01
   9.82708714e+00  1.87271995e+00 -1.82682445e+00  9.28829013e+00
   5.56959815e+00  1.04208363e+01  1.12187898e+01  5.86054176e+00
   1.94915034e+00]
  [-3.00000000e+00 -1.32628798e+00  5.43166932e+00  9.50334710e+00
   4.57098025e+00  2.58745515e+00  4.59194022e+00 -1.40322074e+00
   7.59502709e+00  2.37526651e+00  2.08404690e+00  9.99850477e+00
  -2.50000669e+00 -1.73368610e+01 -5.91919045e+00 -2.79498942e+00
  -7.72103551e+00]
  [ 4.00000000e+00 -5.89423748e+00 -8.55906568e+00 -8.07684600e+00
  -9.08011520e+00 -1.99306440e+00  9.20299976e+00 -5.93997408e+00
  -8.66012583e+00  1.45232818e+00 -1.07276886e+00  8.13324313e+00
   6.11932068e-01  1.72754049e+01  6.33213989e+00 -7.84860728e-01
   1.54121928e-01]
  [-8.00000000e+00 -2.90267097e+00 -7.10516327e-01  2.09963204e+00
  -6.88504690e+00  1.81612589e+01  2.65651853e+00 -3.01660356e+00
  -3.07532296e+00  3.22282938e+00  7.69690431e+00 -8.27527311e+00
  -9.02104293e+00  1.03250661e+01  2.73124270e-02 -6.77495571e+00
  -1.44022702e+01]
  [ 7.00000000e+00 -7.32537542e-01  1.43799397e+00  4.48120022e-02
   2.12533879e-01 -3.60505421e-01 -1.06760657e+01 -7.61718923e-01
  -6.79343389e-03 -1.73964978e+00  4.19153617e+00 -6.36565341e+00
   5.62315941e+00 -2.54728797e+00  7.05420167e+00  6.86544364e+00
   1.71340154e+01]
  [ 2.30000000e+01 -7.09142231e+00 -7.01830591e+00 -6.33916928e+00
  -4.75411188e+00 -2.05826360e+00  5.92248642e+00 -7.16431843e+00
  -5.56067649e+00  1.48832083e+00  4.40736103e-01  1.14499651e+00
   4.51070807e-01  2.09229039e+00 -3.20370907e+00 -6.56777343e-01
  -2.70741226e+00]]
```

Their magnitudes:

```
Magtitude is : [29.58992822 47.74780091 27.53432469 29.65791159 32.5516521  25.61320525
 28.96927871]
```

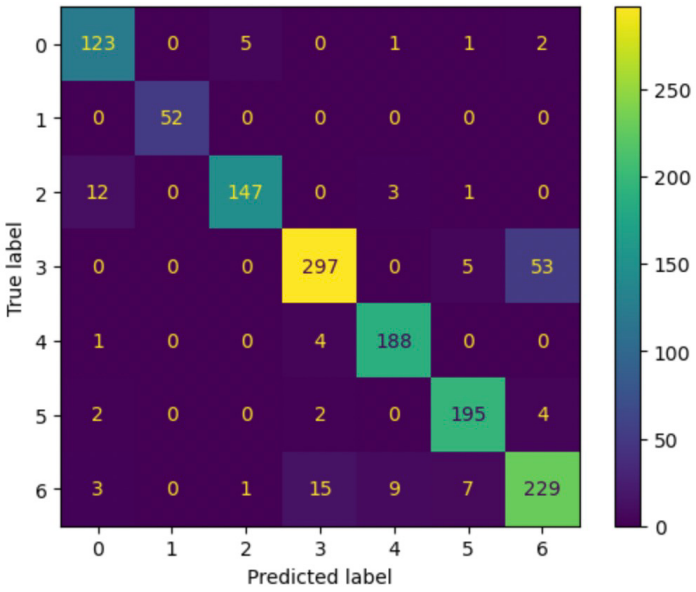
Accuracy and confusion matrix C:

```
The accuracy on the training set is : 91.22377336925463  %
```




```
<Figure size 640x480 with 0 Axes>
```

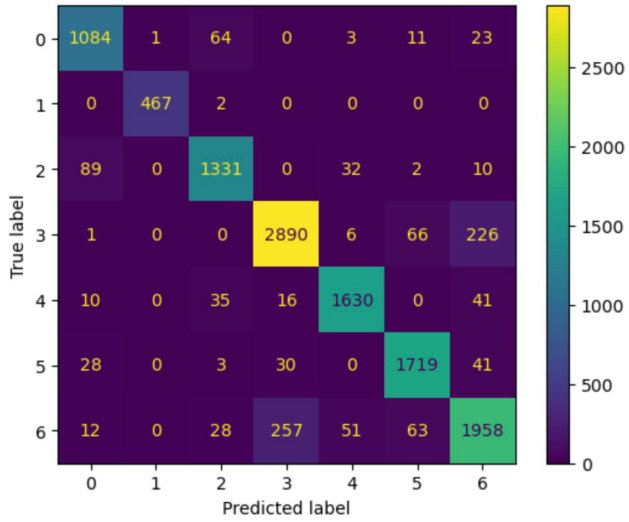
```
The accuracy on the testing    set is : 90.38179148311308  %
```



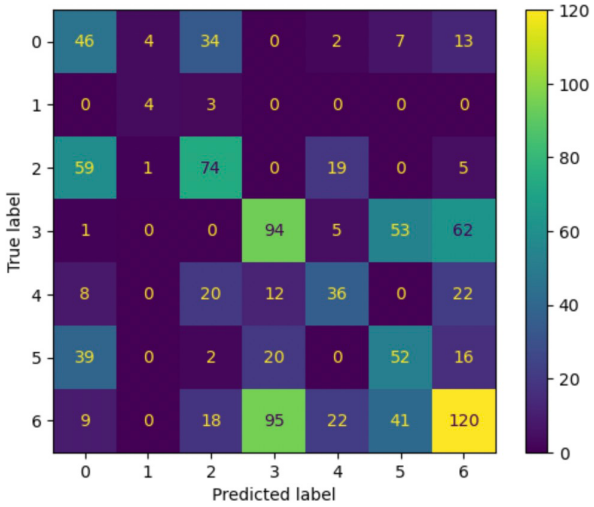

```
<Figure size 640x480 with 0 Axes>
```

(b)

```
The mean for the trainning accuracy is: 0.9047432443464771
The mean for the testing accuracy is: 0.8960352422907489


The std for the trainning accuracy is: 0.0074947205649701755
The std for the testing accuracy is: 0.008137533628485993


The mean for the magnitude is: [29.56702155 45.90045369 30.87991092 29.25864678 32.27456786 25.35509257
 28.82129615]
The std for the magnitude is: [2.32968867 1.71676904 1.60612051 1.78060799 2.88614301 1.19451596
 1.45609997]


The confusion matrix for the mean of training set is:
```

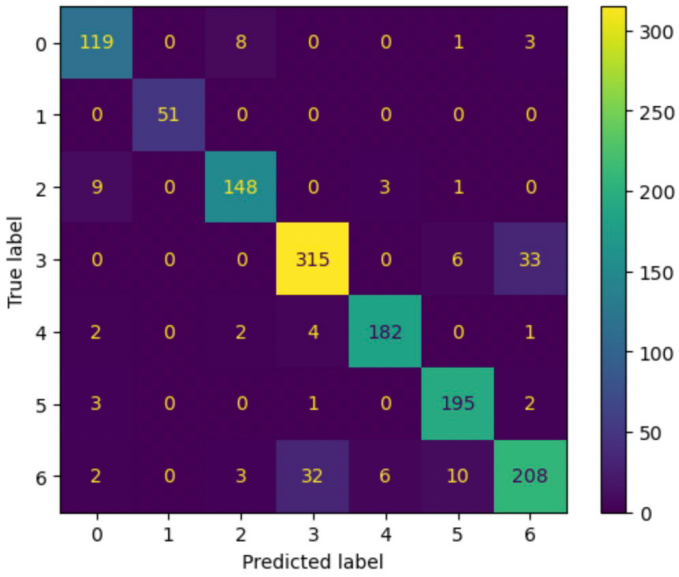



```
<Figure size 640x480 with 0 Axes>

The confusion matrix for the std of training set is:
```



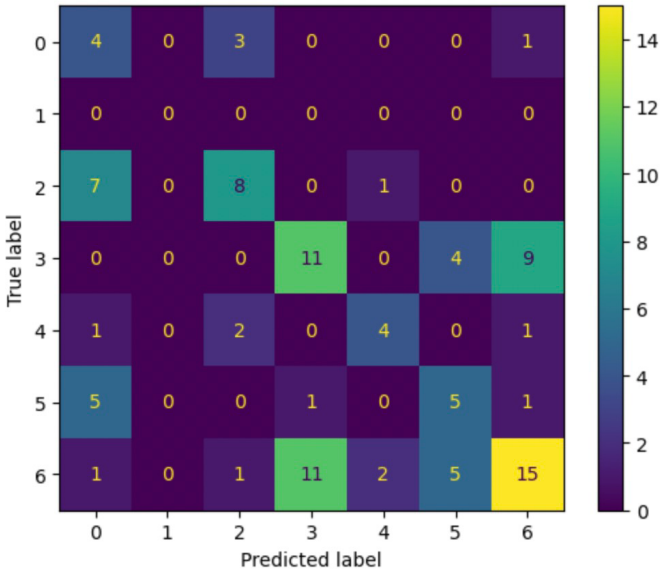

```
<Figure size 640x480 with 0 Axes>
```

The confusion matrix for the mean of testing set is:



<Figure size 640x480 with 0 Axes>

The confusion matrix for the std of testing set is:

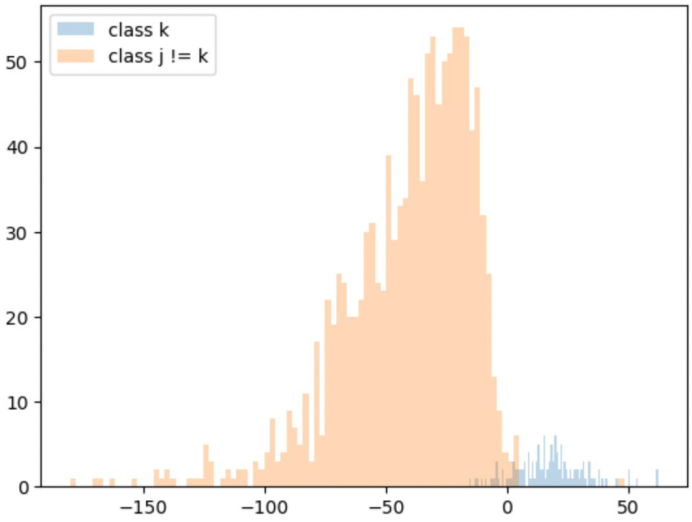


<Figure size 640x480 with 0 Axes>

2.(a) OvR classifier

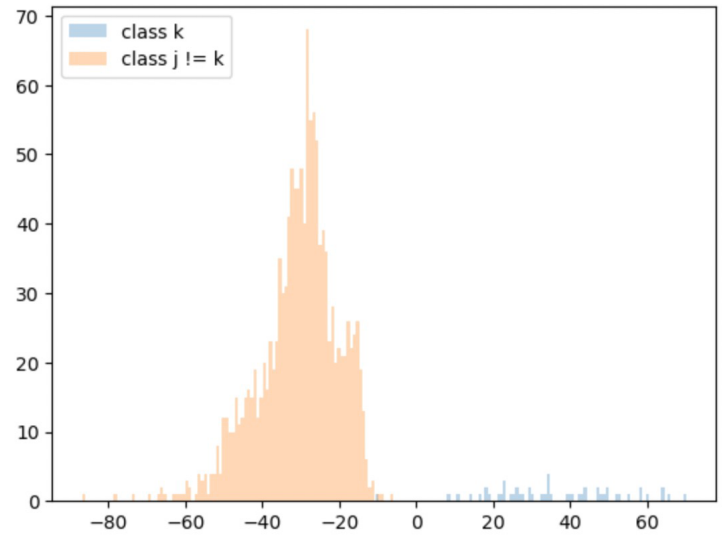
Class 1:

```
i2 reach.
Weight matrix is: [-36.         -28.60087316  54.73154488 -28.1266936  -14.98698367
 -49.62951441 -21.90425284 -15.73967196 -21.68719729   2.1055915
  -4.79083549  -1.15495649  -8.08200612 -21.48593591 -72.43551164
 -22.73420473  13.20265585]
Min J is: 186
Accuracy for training data is: 98.48150869458732 %
Accuracy for testing data is: 98.09104258443465 %
```



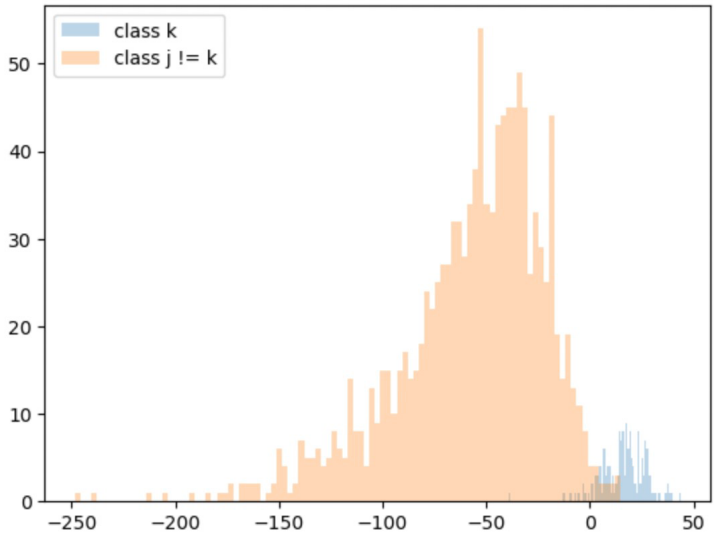
Class 2:

```
i1 reach. Data is linearly separable
Weight matrix is: [-28.          8.44727203   1.66088407   0.81776488   3.3682316
  -2.63667497   2.79285994   8.45797429   2.22162419   1.74867361
   2.05566716   4.02851812   2.87062772   8.63314079   4.27164211
   2.15745894   2.73616084]
Min J is: 0
Accuracy for training data is: 100.0 %
Accuracy for testing data is: 99.92657856093979 %
```



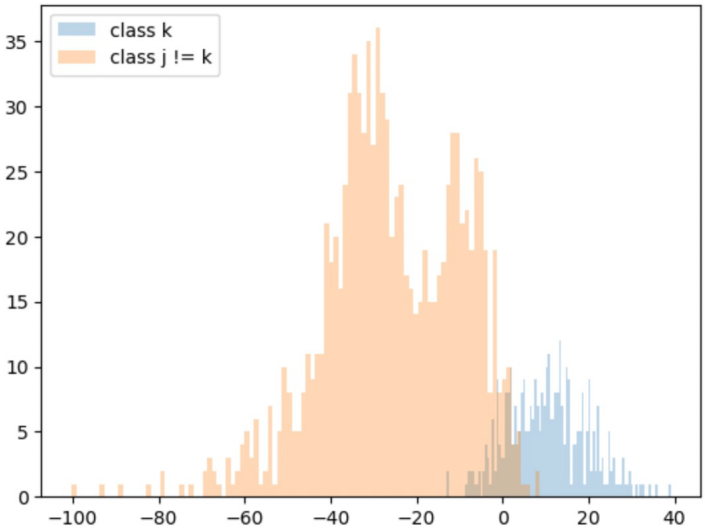
Class 3:

```
i2 reach.
Weight matrix is: [ -49.          -30.65660599  -31.85917368   69.815186    -77.37273476
  -67.3783354    28.80888287   -9.09333362   -4.83854503    0.81856057
   -3.2413633    13.70298145  -35.74025225 -140.06376002   68.47276459
  -65.31643919   -3.88222522]
Min J is: 285
Accuracy for training data is: 97.67327945138379 %
Accuracy for testing data is: 98.16446402349486 %
```



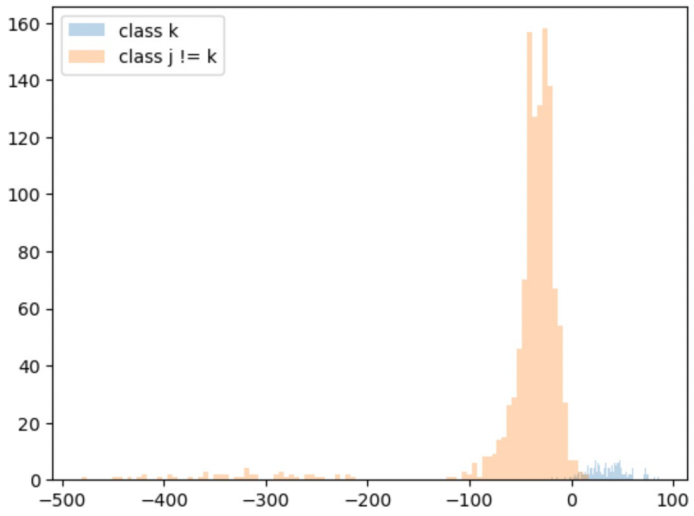
Class 4:

```
i2 reach.
Weight matrix is: [-1.60000000e+01 -4.95161341e+00  1.92317028e+01 -7.08014034e+00
  1.12832912e+01  2.45856703e-02  5.87694159e+01 -1.30612487e+01
  3.73229927e+00 -2.81903386e+00  1.86703210e+00  6.00631650e+00
  2.25374026e+01  1.18678333e+01  2.64575279e+01  1.10847086e+01
  1.31197165e+00]
Min J is: 592
Accuracy for training data is: 95.16695240427791 %
Accuracy for testing data is: 94.27312775330397 %
```



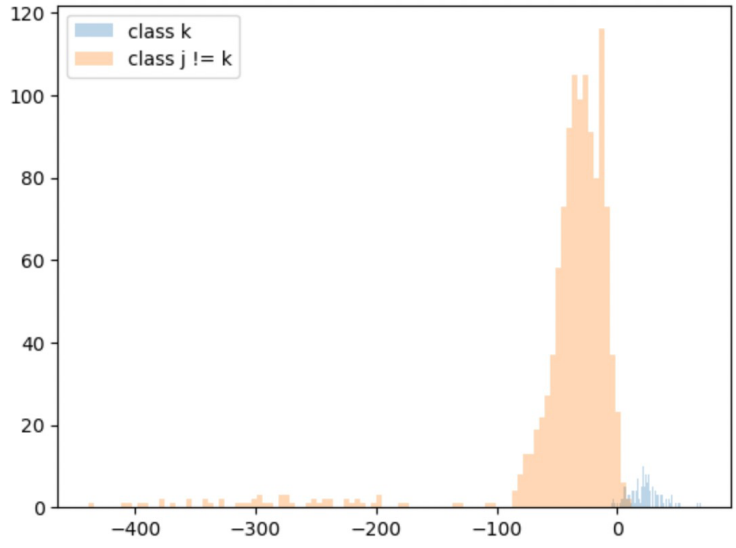
Class 5:

```
i2 reach.
Weight matrix is: [-36.         -45.50515993   9.49695167   1.83255746 -10.75525762
  69.73096833   9.81141473 -47.66518436 -12.75444641   0.64219741
   3.69507221  -3.18176461  11.31827127 -48.10524125   0.85186481
  27.54056961  -6.12319121]
Min J is: 220
Accuracy for training data is: 98.20393501510327 %
Accuracy for testing data is: 98.38472834067548 %
```



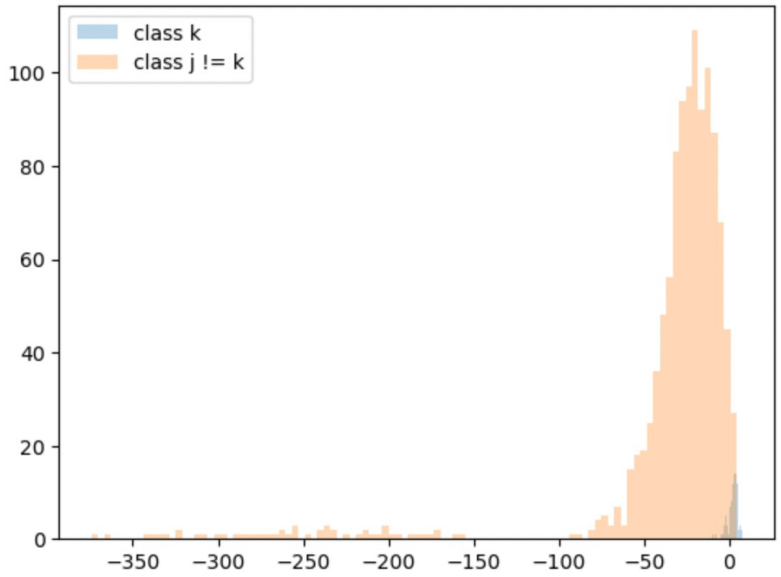
Class 6:

```
i2 reach.
Weight matrix is: [ -34.          -29.33074247  -26.20480651   16.98748079  -49.9632721
   -6.51176292  -22.29610256  -35.63303252   -5.39689601   -2.64479815
   -0.52265602    1.0786598   -26.26811941 -103.93958625   26.43716586
  -27.07839638   -0.18875012]
Min J is: 265
Accuracy for training data is: 97.83655808637441 %
Accuracy for testing data is: 97.79735682819384 %
```



Class 7:

```
i2 reach.
Weight matrix is: [ -28.           11.56025143 -180.07677194   13.84223432   -4.35203167
  -39.61881869  -14.80187212  -15.87783617   27.98203347    1.05487084
    0.46028178  -25.72585778  -23.09950017  -87.93645621  -11.45523474
  -58.81522415    0.2653616 ]
Min J is: 752
Accuracy for training data is: 93.86072332435302 %
Accuracy for testing data is: 93.3186490455213 %
```



(b) Rule 1:

```
Classify the training data using decision rule 1:
The accuracy rate is : 86.71728304351376 %
The error rate is : 5.208588456200506 %
The unclassified rate is : 8.074128500285738 %


Classify the testing data using decision rule 1:
The accuracy rate is : 86.04992657856094 %
The error rate is : 6.093979441997063 %
The unclassified rate is : 7.856093979441997 %
```

(c) Rule 2:

```
Classify the training data using decision rule 2:
The accuracy rate is : 91.26459302800228 %
The error rate is : 8.735406971997714 %
The unclassified rate is : 0.0 %


Classify the testing data using decision rule 2:
The accuracy rate is : 90.30837004405286 %
The error rate is : 9.691629955947137 %
The unclassified rate is : 0.0 %
```

(d) Rule 3:

```
Classify the training data using decision rule 3:
The accuracy rate is : 91.11764225651073 %
The error rate is : 8.882357743489266 %
The unclassified rate is : 0.0 %


Classify the testing data using decision rule 3:
The accuracy rate is : 90.08810572687224 %
The error rate is : 9.911894273127754 %
The unclassified rate is : 0.0 %
```

(e) Compare the results of (b)-(d)

Obviously, rule 2 and rule 3 perform better than rule 1 according to the accuracy rate. But the error rate of rule 1 is much smaller than the other one. But there exist an indeterminate region when using rule 1 so there is unclassified data in rule 1's result as the other one has no indeterminate region.

3.(a) Yes.
g_k(x) / norm||w_k|| is the signed distance form hyperplane H to point x. In the indeterminate region, which means all the g_k(x) < 0, all the distance from H to the x should be negative. Therefore, the maximum of g_k(x) / norm||w_k|| is the minimum abs(g_k(x) / norm||w_k||), which means this class k's decision boundary is the closest to this point x. Therefore, k = argmax(g_k(x) / norm||w_k||) which is the closest class. To conclude, this will assign x_n to the same class whose decision region closest to x_n.

(b)Yes.
We can separate the feature region into two regions. The first one is the determinate region and the second one is the indeterminate region.

Obviously, in the second indeterminate region, the decision rules from Approach (i) and Approach (ii) are the same. Therefore, to figure this out, we only need to focus on the first determinate region.

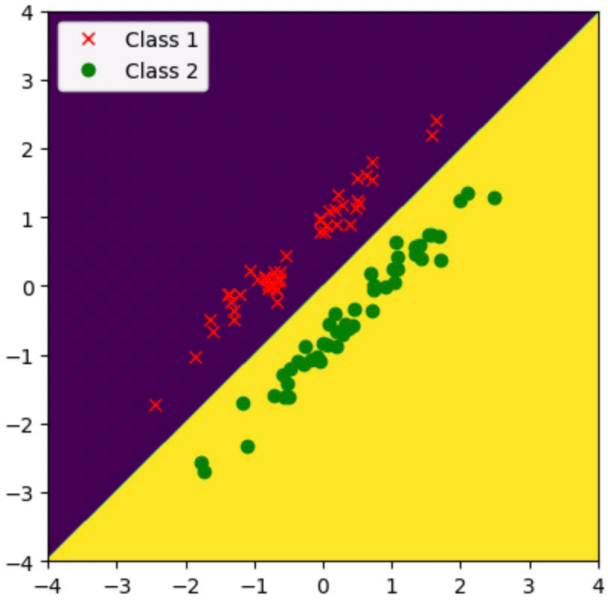
According to the Approach (i), to classify x to class k can only be done by g_k(x) > 0 and g_j(x) < 0 (j != k). Therefore, g_k(x) = max(g(x)) which means k = argmax(g(x)), which is the same as the Approach (ii).

In my opinion, Approach (i) and Approach (ii) are equivalent.

4.(a) Linear classification

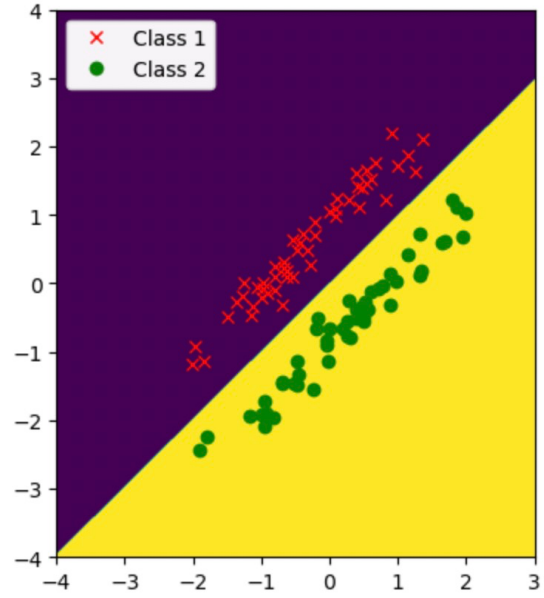
Dataset1_train:

```
i1 reach. Data is linearly separable
Weight matrix is: [ 0.         -2.09552505  2.11386957]
Min J is: 0
```




```
Accuracy for training data 1 is:  100.0 %
```

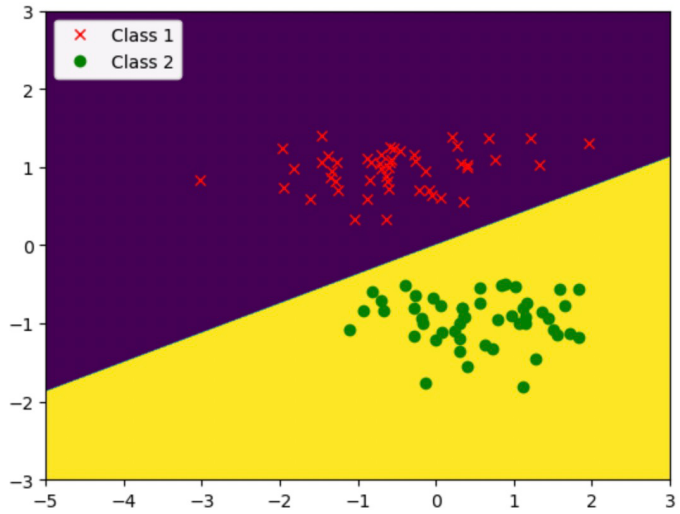
Dataset1_test:




```
Accuracy for testing data 1 is:  100.0 %
```

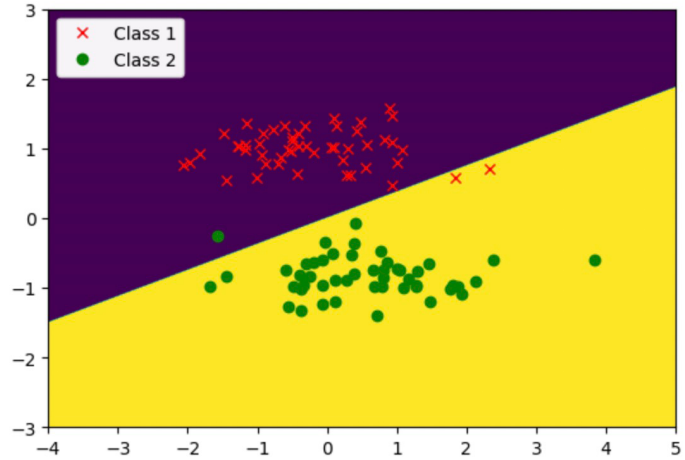
Dataset2_train:

```
i1 reach. Data is linearly separable
Weight matrix is: [ 0.         -0.58388862  1.55803305]
Min J is: 0
```




```
Accuracy for training data 2 is:  100.0 %
```

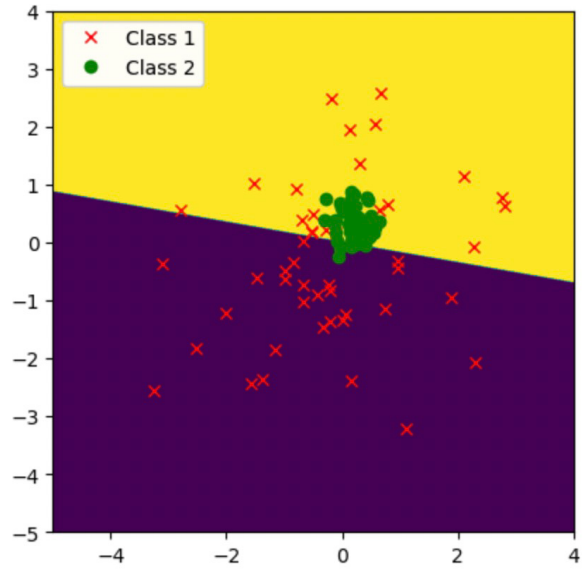
Dataset2_test:




```
Accuracy for testing data 2 is:  96.96969696969697 %
```

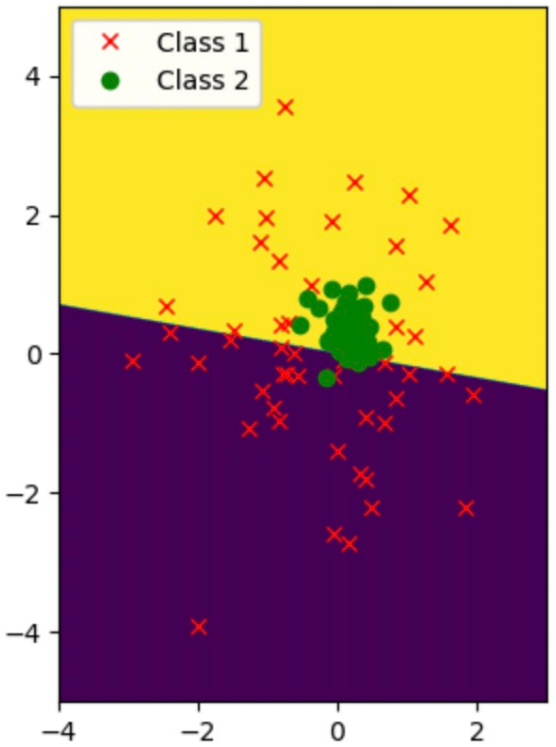
Dataset3_train:

```
i2 reach.
Weight matrix is: [ 0.         -0.00949884 -0.05430716]
Min J is: 1.0980766158415043
```




```
Accuracy for training data 3 is:  74.74747474747475 %
```

Dataset3_test:




```
Accuracy for testing data 3 is:  75.75757575757575 %
```

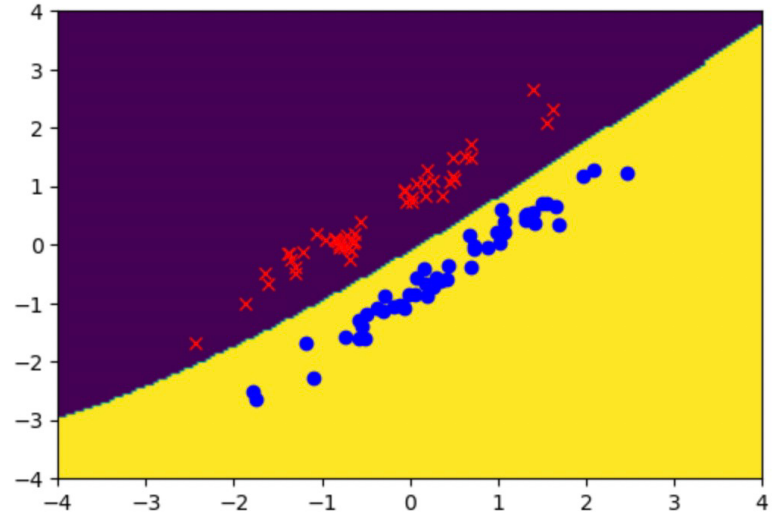
(b) Linear classification 10 runs

```
The mean of accuracy for the training data 1 is : 100.0 %
The mean of accuracy for the testing data 1 is : 99.3939393939394 %
The std of accuracy for the training data 1 is : 0.0
The std of accuracy for the testing data 1 is : 1.0301049522409669


The mean of accuracy for the training data 2 is : 100.0 %
The mean of accuracy for the testing data 2 is : 98.68686868686868 %
The std of accuracy for the training data 2 is : 0.0
The std of accuracy for the testing data 2 is : 1.693237839822244


The mean of accuracy for the training data 3 is : 55.45454545454546 %
The mean of accuracy for the testing data 3 is : 54.34343434343434 %
The std of accuracy for the training data 3 is : 20.28996011281075
The std of accuracy for the testing data 3 is : 20.866838683353016
```

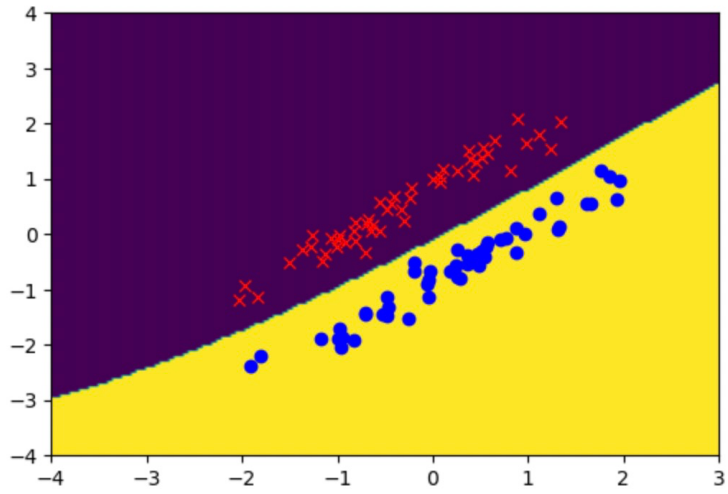
(c) Nonlinear-quadratic classification

Dataset 1:

```
i1 reach. Data is linearly separable
Weight matrix is: [ 1.         -7.66323998  8.7350138  -0.6293219   0.07998281  0.39826552]
Min J is: 0
```



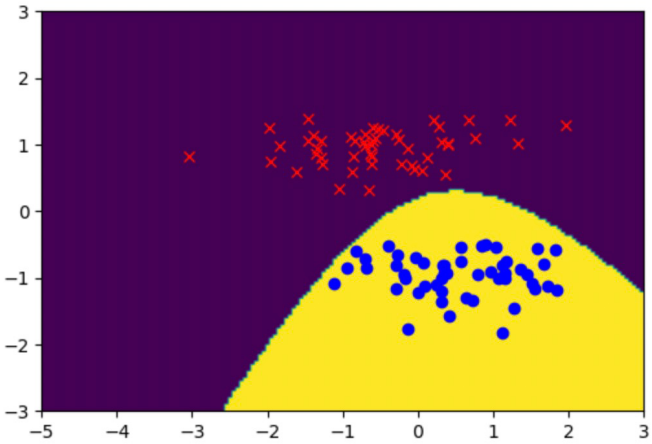

```
Accuracy rate for training data 1 : 100.0 %
```



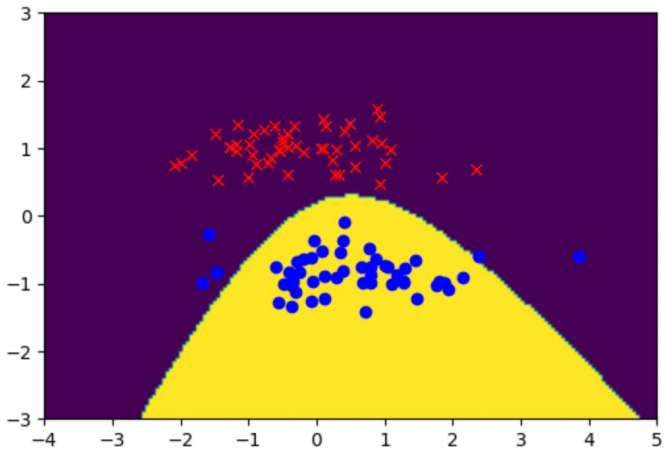

```
Accuracy rate for testing data 1 : 100.0 %
```

Dataset 2:

```
i1 reach. Data is linearly separable
Weight matrix is: [-1.         -2.23120865  5.25456869  1.97562164  0.68693003 -0.86031339]
Min J is: 0
```



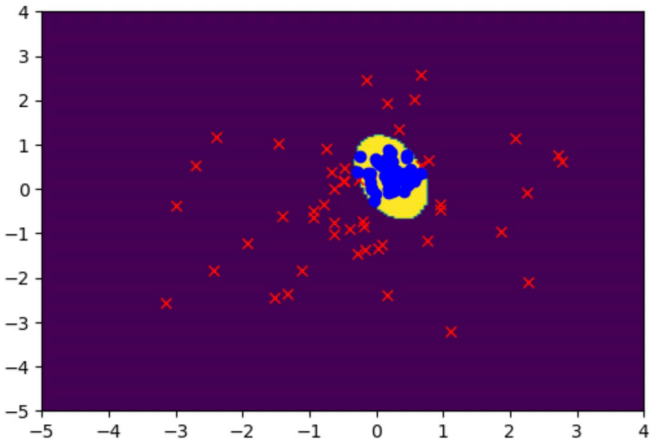

```
Accuracy rate for training data 2 : 100.0 %
```

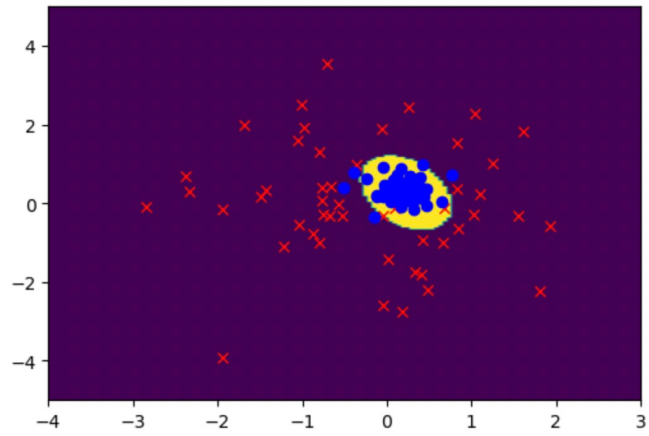


```
Accuracy rate for testing data 2 : 95.0 %
```

Dataset 3:

```
i2 reach.
Weight matrix is: [-1.         -3.86819942 -1.75654489  6.64888088  2.73306438  2.14209377]
Min J is: 2
```





```
Accuracy rate for testing data 2 : 93.0 %
```

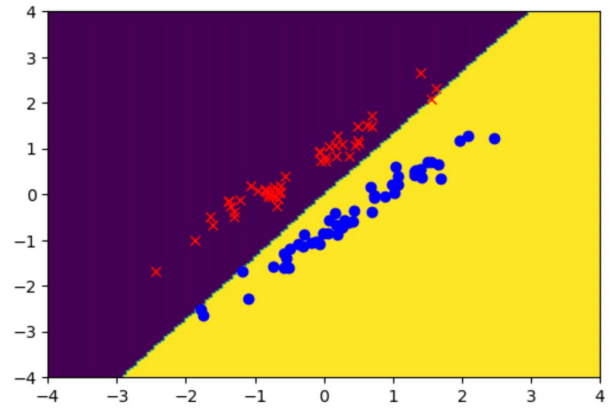
(d) Nonlinear-quadratic classification 10 run

```
The mean of accuracy for the training data 1 is : 100.0 %
The mean of accuracy for the testing data 1 is : 99.7 %
The std of accuracy for the training data 1 is : 0.0
The std of accuracy for the testing data 1 is : 0.6403124237432849


The mean of accuracy for the training data 2 is : 100.0 %
The mean of accuracy for the testing data 2 is : 96.4 %
The std of accuracy for the training data 2 is : 0.0
The std of accuracy for the testing data 2 is : 1.3564659966250536


The mean of accuracy for the training data 3 is : 98.0 %
The mean of accuracy for the testing data 3 is : 92.1 %
The std of accuracy for the training data 3 is : 0.0
The std of accuracy for the testing data 3 is : 0.8306623862918076
```

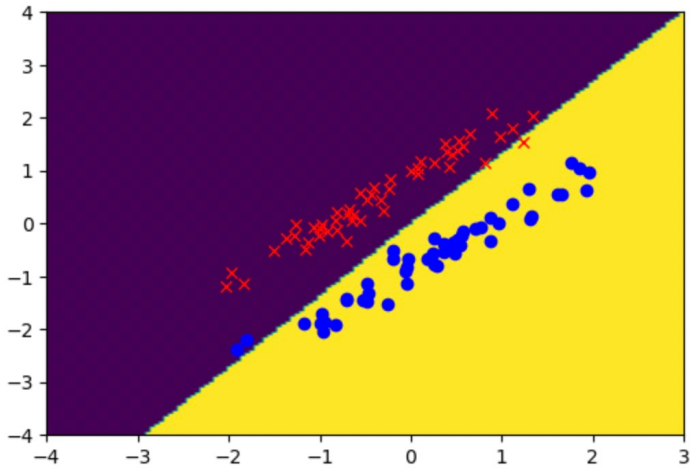
(e) Nonlinear-cubic classification

Dataset 1:

```
i1 reach. Data is linearly separable
Weight matrix is: [ 0.         -1.61543874  1.24146949 -0.28421054  0.65143742 -0.0449748
 -5.97039289 -0.67180192  0.92055614  2.10049763]
Min J is: 0
```



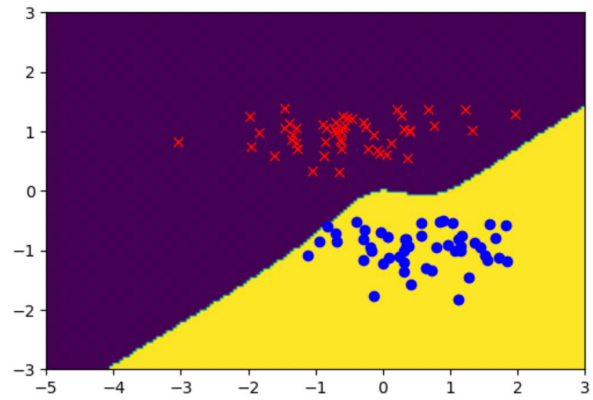

```
Accuracy rate for training data 1 : 100.0 %
```



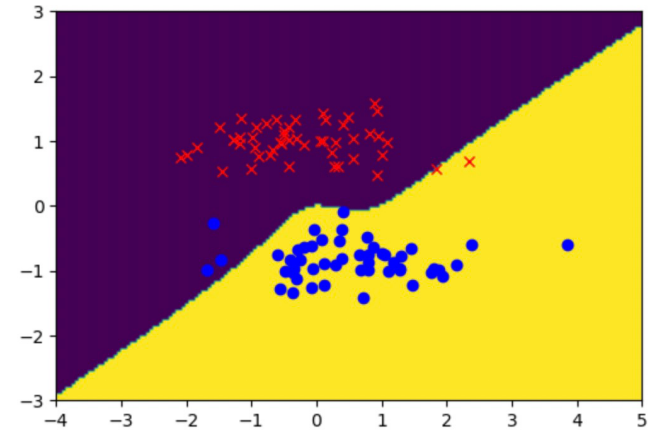
```
Accuracy rate for testing data 1 : 97.0 %
```

Dataset 2:

```
i1 reach. Data is linearly separable
Weight matrix is: [ 0.         -0.21279609  3.29674302  3.02243026 -0.14617657 -0.16473553
 -2.91061976  2.98469536  0.49709636  2.56366838]
Min J is: 0
```



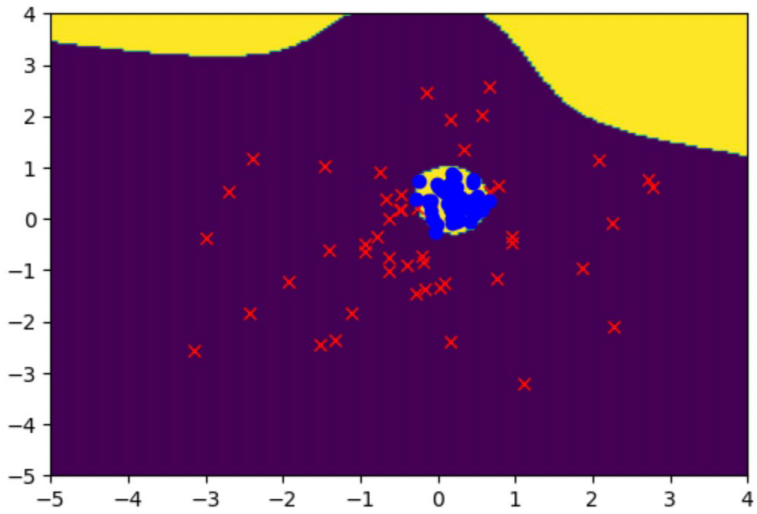
```
Accuracy rate for training data 2 : 100.0 %
```



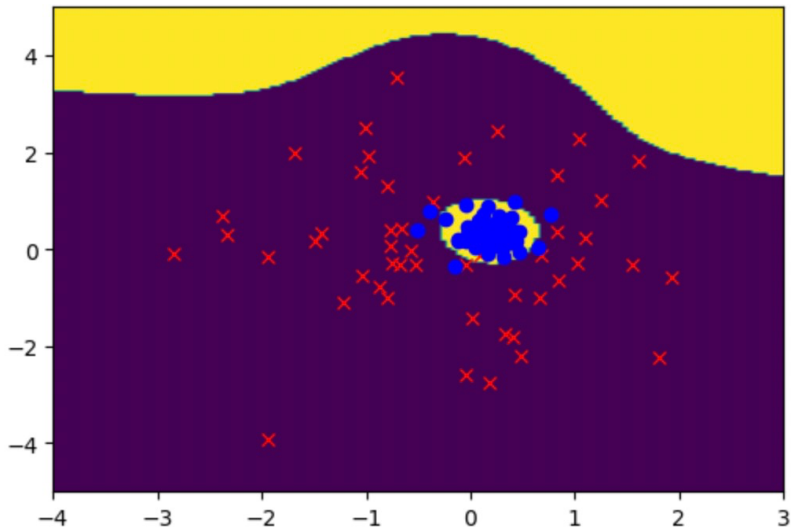
```
Accuracy rate for testing data 2 : 95.0 %
```

Dataset 3:

```
i2 reach.
Weight matrix is: [ -4.         -15.39150474 -13.2885659   40.41485695  13.15847859
  21.44429684  -4.17019256 -18.59555545  -3.22875261  -4.12572051]
Min J is: 3
```




```
Accuracy rate for training data 3 : 97.0 %
```




```
Accuracy rate for testing data 2 : 92.0 %
```

(f) Nonlinear-cubic classification 10 run

```
The mean of accuracy for the training data 1 is : 100.0 %
The mean of accuracy for the testing data 1 is : 99.5 %
The std of accuracy for the training data 1 is : 0.0
The std of accuracy for the testing data 1 is : 0.806225774829855


The mean of accuracy for the training data 2 is : 100.0 %
The mean of accuracy for the testing data 2 is : 96.1 %
The std of accuracy for the training data 2 is : 0.0
The std of accuracy for the testing data 2 is : 1.9209372712298547


The mean of accuracy for the training data 3 is : 97.8 %
The mean of accuracy for the testing data 3 is : 91.4 %
The std of accuracy for the training data 3 is : 0.4
The std of accuracy for the testing data 3 is : 1.3564659966250536
```

(g) Explain using words and mathematical expressions, how the PlotNonlinear.py routine finds the final decision regions as mapped back into the original feature space.

First, the function defines a grid of points across the feature space. By finding the minimum and maximum values for each feature and expand the grid to ensure the whole feature space is covered.
Xrange and yrange define the range of grid and inc was used to define the step size.

```
xrange = (min_x, max_x)
yrange = (min_y, max_y)
inc = 0.05
```

By using the meshgrid() NumPy function, create a grid from np.arange(xrange[0], xrange[1]+inc/100, inc) and np.arange(yrange[0], yrange[1]+inc/100, inc) two vector.

```
(x, y) = np.meshgrid(np.arange(xrange[0], xrange[1]+inc/100, inc), np.arange(yrange[0], yrange[1]+inc/100, inc))
```

Define the first feature x1 is our x-axis of the feature space and x2 as our y-axis of the feature space, then we need one row of x1 values of the grid for each point on the y-axis and one column of x2 values of the grid for each point on the x-axis.

The function stack the vectors, to make (x,y) pairs as a bunch of row vectors and get the transformed matrix to find the predict label.

```
xy = np.hstack( (x.reshape(x.shape[0]*x.shape[1], 1, order='F'), y.reshape(y.shape[0]*y.shape[1], 1, order='F')) )
```

Then feed this into the model and get a prediction for each point in the grid by using the predictor which defined by ourself.

```
pred_label = predictor(xy, weight)
```

What's more, plot the grid of values as a curve. The function use x.shape that we defined earlier and reshape the pred_label from the model to have the same shape.

```
decisionmap = pred_label.reshape(image_size, order='F')
```

Finally, using plt.imshow to plot the decision surface with two different color. And also plot each data points according to their actual label.

(h) Compare your results of (a)-(f) (e.g., compare plots to plots; mean (and std.) accuracies to mean (and std.) accuracies). You can use standard deviations to gauge whether small differences in mean accuracies are statistically significant. Try to explain what you observe.

Comparing plots to plots, we can find that the decision boundary of linear classification is a line with offset while the decision boundary of nonlinear classification is a curve. And comparing the curve of quadratic classification results with cubic classification results, the former curve is like a part of a circle or an oval which is much smoother while the other one has more twists.

Comparing the mean and std. of the accuracies, the linear classification performs best when the data can be linearly separatable but on the other hand, the mean accuracy is low, and the std. is large when the data can't be linearly separatable. The nonlinear classifications perform much more stable in both situations considering the smaller std. The mean accuracy of both quadratic and cubic are good and differ slightly. The std. of cubic is a little bigger than the quadratic's which may be caused by the higher degree of the polynomial.