INF 552: Assignment #2

Mohammad Reza Rajati

Muxin Liang

Contents

Exercise # 1

Time Series Classification: In this problem, we will classify the activities of humans based on time series obtained by a Wireless Sensor Network.

(a)

Download the AReM data from the web page and put them into folders.
Since there are some flaws in those files, I modified them for easy manipulation in further operations.

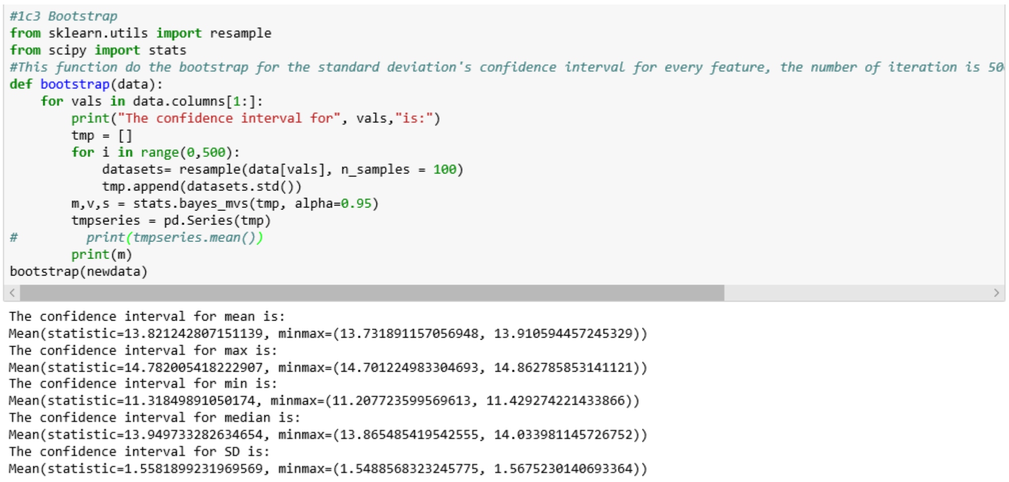
(b)

Keep datasets 1 and 2 in folders bending1 and bending 2, as well as datasets 1, 2, and 3 in other folders as test data and other datasets as train data.
This process has been done in the Jupyter Notebook.

(c)

Feature Extraction:

1. Research on types of time-domain features:
   Minimum, maximum, median, 75 percentile, 25 percentile, mean, variance, standard deviation, etc.
2. Extract the time-domain features for all of the 6 time series in each instance.
   I used them directly, process can be seen from the Notebook. And the feature I selected were: mean, max, median, min, standard deviation. I store them in *Features_records.csv*.
3. Estimate the standard deviation, Bootstrap, and Confidence Interval Interval:
   I used 100 as the size for each sample and 500 as the iteration time, and the result below shows my confidence interval for those variables.



```
#1c3 Bootstrap
from sklearn.utils import resample
from scipy import stats
#This function do the bootstrap for the standard deviation's confidence interval for every feature, the number of iteration is 50
def bootstrap(data):
    for vals in data.columns[1:]:
        print("The confidence interval for", vals,"is:")
        tmp = []
        for i in range(0,500):
            datasets= resample(data[vals], n_samples = 100)
            tmp.append(datasets.std())
        m,v,s = stats.bayes_mvs(tmp, alpha=0.95)
        tmpseries = pd.Series(tmp)
#         print(tmpseries.mean())
        print(m)
bootstrap(newdata)
```

```
The confidence interval for mean is:
Mean(statistic=13.821242807151139, minmax=(13.731891157056948, 13.910594457245329))
The confidence interval for max is:
Mean(statistic=14.782005418222907, minmax=(14.701224983304693, 14.862785853141121))
The confidence interval for min is:
Mean(statistic=11.31849891050174, minmax=(11.207723599569613, 11.429274221433866))
The confidence interval for median is:
Mean(statistic=13.949733282634654, minmax=(13.865485419542555, 14.033981145726752))
The confidence interval for SD is:
Mean(statistic=1.5581899231969569, minmax=(1.5488568323245775, 1.5675230140693364))
```

Figure 1: result for bootstrapped confidence interval for time domain variables

4. Use your judgement to select the three most important time-domain features:
   My selection: Mean, Median, Standard Deviation. Because these are the most common features I encountered in statistic learning to grasp the feature of data.

(d)

Binary Classification Using Logistic Regression

1. Scatter plot for bend and other data.
   For easier manipulation, I represented *time_series* 1, 2,6's mean, median, std into number 1-9:
   $1 = timeseries1\_mean$
   $2 = timeseries2\_mean$
   $3 = timeseries6\_mean$
   $4 = timeseries1\_median$
   $5 = timeseries2\_median$
   $6 = timeseries6\_median$
   $7 = timeseries1\_std$
   $8 = timeseries2\_std$
   $9 = timeseries6\_std$
   And the points for bend data is in blue and for other data is in red.

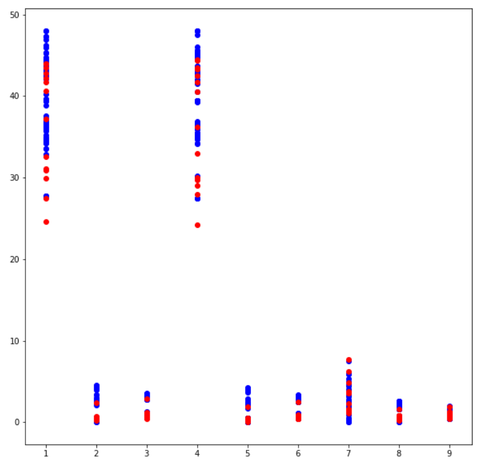


Figure 2: Scatter plot for bend data and other data

2. Scatter plot for equal split data, the representation is the same as previous one. From the observation, we can find that bend data is different from others because the points are more discrete in figure 2 comparing to figure 3.

3. CV, Feature Selection:
   There are two ways in doing cross validation, the right and wrong. The right way is to do validation before feature selection and after feature selection. The wrong way is to do validation only after feature selection. I will use the right way to solve this problem.

   Firstly, I used 5-fold validation to select the best l; To find the best l, I run 5-fold for 50 times and store the best mean accuracy count in a dictionary for every value of l from 1 to 10; and it turns out that the best accuracy happens at $l = 2$. Check note book for the codes and the result in dictionary.

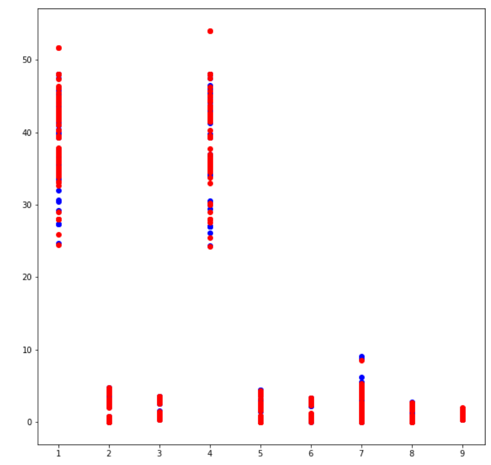


Figure 3: Scatter plot for equal-splited data

Then, I do the feature selection using both p-value and feature selection tools: After generated my train and test data using l = 2 the accuracy was about 92.75%. I used fpr to generate the p-values and the scores for all the 18 features(combination of 6 types of time series data and the 3 time domain features I chose: mean, median, std). I selected those predictors whose p-value was less than 0.01 and put them into feature selection tools. To select the best features, I only choose those predictor who survive from both p-value test and feature selection(detailed codes on Notebook), and the remained features are(SD for standard deviation):

['var_rss12_mean', 'var_rss12_median', 'var_rss12_SD', 'var_rss13_median', 'avg_rss23_mean', 'avg_rss23_median', 'var_rss23_mean', 'var_rss23_SD']

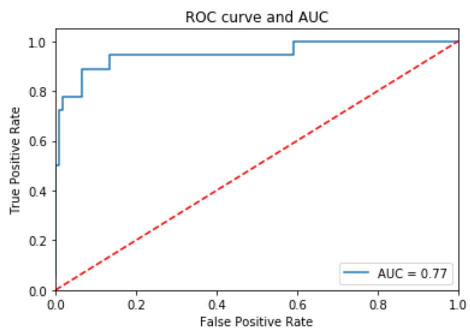
Then I put those features into Logistic Regression and do fit and predict, the result will be shown in the following sections.

4. Confusion matrix, ROC and AUC:
   I put both train set and test set into prediction to get an overview of the classifier. Here I mark bending as 1 and other as 0.

   The following graphics contains those information, from top to bottom: ROC and AUC information, coefficient for every selected predictors(β_i), classification report, and confusion metrics.

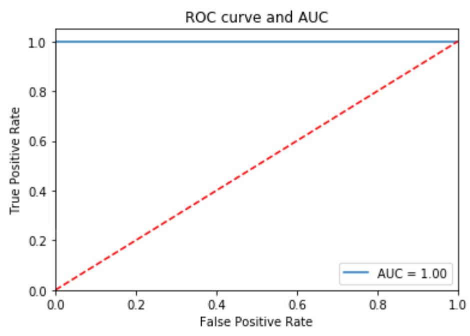
   The test data performs a better classification result from observation, an amazing 100%. A possible reason is the size of data in test set is small because we used $l = 2$ which will only give 38 pieces of data in test set.

   By observing confusion matrix, we can conclude that there are 8 "bending" cases was recognized as "other" (False negative) and 1 "other" recognized as "bending"(False positive) when using train data set for classification.



```
[[-0.41349966 -0.42992182  0.33662812 -0.92185539  0.88522457 -0.68553069
  -1.04748672 -0.3686197 ]]
             precision    recall  f1-score   support

        0.0       0.94      0.99      0.96       120
        1.0       0.91      0.56      0.69        18

avg / total       0.93      0.93      0.93       138

[[119   1]
 [  8  10]]
```

Figure 4: Classification result using train data



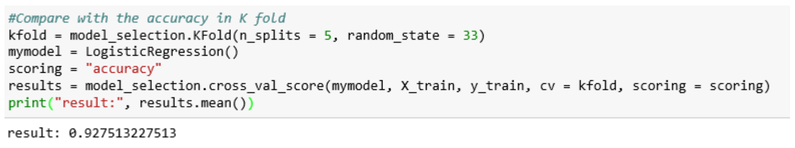
```
[[-0.41349966 -0.42992182  0.33662812 -0.92185539  0.88522457 -0.68553069
  -1.04748672 -0.3686197 ]]
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00        30
        1.0       1.00      1.00      1.00         8

avg / total       1.00      1.00      1.00        38

[[30  0]
 [ 0  8]]
```

Figure 5: Classification result using test data

5. Test set accuracy and CV accuracy:
   We have obtained the accuracy for test set, which is 100%, and for Cross-Validation, which is 92.75%. This shows that feature selection improved the accuracy of logistic regression.

6. Whether the classes are well separated enough to cause instability?
   To check this, I plotted pairwise scatter plot against all selected features of bending data and other data respectively.

   By checking every subplot in scatter matrix, I find that the features for bending data have some



```
#Compare with the accuracy in K fold
kfold = model_selection.KFold(n_splits = 5, random_state = 33)
mymodel = LogisticRegression()
scoring = "accuracy"
results = model_selection.cross_val_score(mymodel, X_train, y_train, cv = kfold, scoring = scoring)
print("result:", results.mean())
```

```
result: 0.927513227513
```

Figure 6: Accuracy for CV

significant patterns comparing to those for other data. I have to say that the classes bend and other separated from each other so there may be some instability in calculating logistic regression parameters.

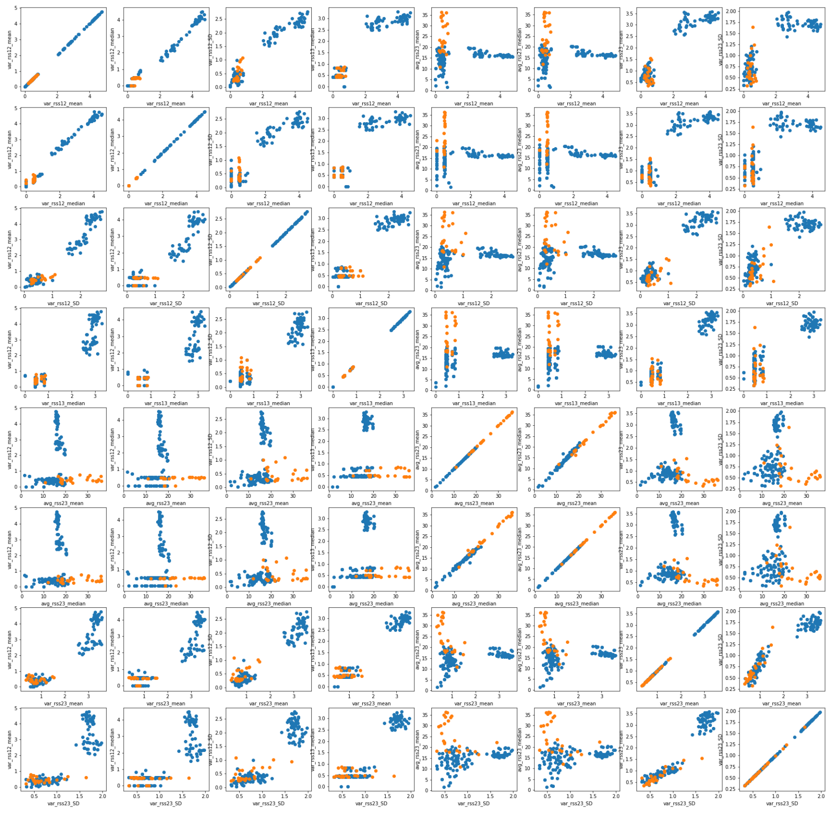


Figure 7: Scatter plot for classes separation

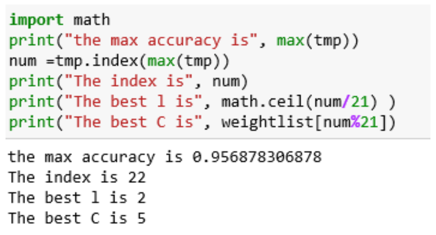
7. Since the test accuracy reached 100%, there is no imbalanced class. Using case-control to get a better accuracy for train data may cause overfit.

(e)

1. L_1-penalized logistic regression:
   I used 5-fold cross validation to get the best value of l and c, value of l was chosen from 1 to 10 and value of c was chosen from the list:

$$[1/\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}, 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50]$$

   the detailed code is in the notebook. The best result shows l = 2 and c = 5. Then I put the parameters into the training.



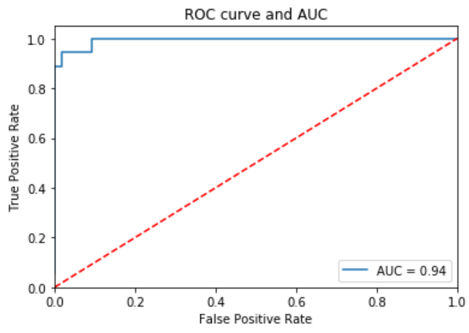
```
import math
print("the max accuracy is", max(tmp))
num =tmp.index(max(tmp))
print("The index is", num)
print("The best l is", math.ceil(num/21) )
print("The best C is", weightlist[num%21])
```

```
the max accuracy is 0.956878306878
The index is 22
The best l is 2
The best C is 5
```

Figure 8: Choosing l and c for best accuracy in CV

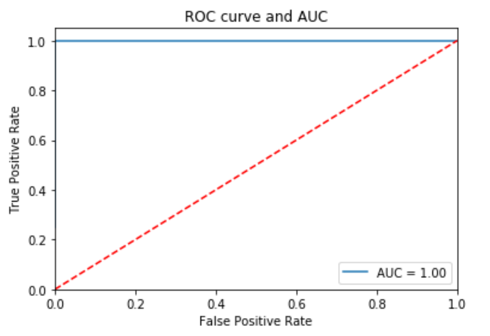
2. Fitting, predicting and comparing:
   I also ran train set and test set for testing and generated ROC curve, classification scores and confusion matrix for L_1-penalized logistic regression.

   From observation, the accuracy of test set classification also reached 100%, and there is also an increment in accuracy in train set classification (0.98 vs 0.93). Therefore we conclude L_1-penalty regression is better, and it is easier to implement. Detailed code on Notebook.



```
Confusion Metris:
             precision    recall  f1-score   support

        0.0       0.98      0.99      0.99       120
        1.0       0.94      0.89      0.91        18

avg / total       0.98      0.98      0.98       138

[[119   1]
 [  2  16]]
```

Figure 9: Classification result using train data(L_1)



```
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00        30
        1.0       1.00      1.00      1.00         8

avg / total       1.00      1.00      1.00        38

[[30  0]
 [ 0  8]]
```

Figure 10: Classification result using test data(L_1)

(f)

Multi-class Classification:

1. I used l = 2, c = 5 to build my multinomial regression model. And found an alternative way to plot ROC curve. For easier manipulation, I represent [”bending”, ”cycling”, ”lying”, ”sitting”, ”standing”, ”walking”] as [1,2,3,4,5,6] respectively. And I generated confusion matrix and classification report for test error. The error rate is 8%.

```
             precision    recall  f1-score   support

        1.0       1.00      1.00      1.00         8
        2.0       1.00      1.00      1.00         6
        3.0       1.00      1.00      1.00         6
        4.0       0.80      0.67      0.73         6
        5.0       0.71      0.83      0.77         6
        6.0       1.00      1.00      1.00         6

avg / total       0.92      0.92      0.92        38

[[8 0 0 0 0 0]
 [0 6 0 0 0 0]
 [0 0 6 0 0 0]
 [0 0 0 4 2 0]
 [0 0 0 1 5 0]
 [0 0 0 0 0 6]]
```

Figure 11: Classification result using test data for multiple classes(L_1)

I also generated the ROC curve for every class. We can see most of the errors happens when classifying standing.

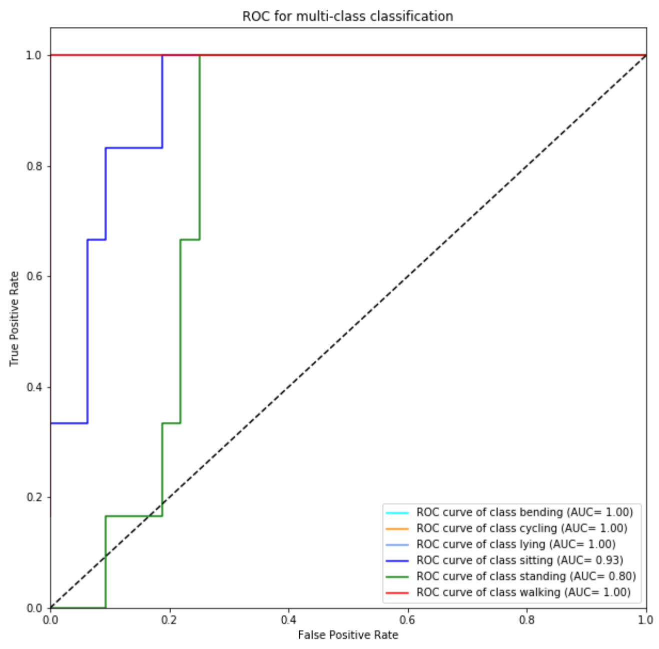
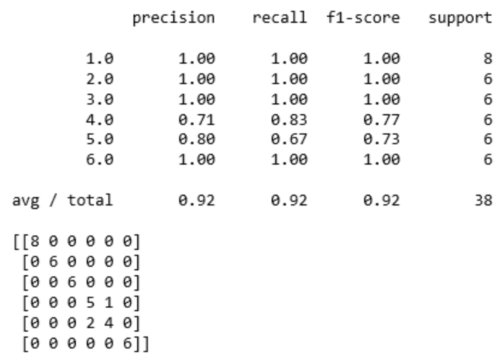


Figure 12: ROC for multiple classes(L_1)

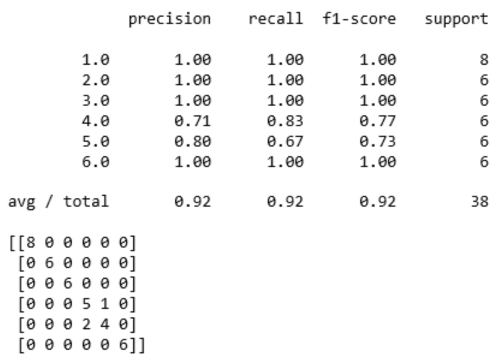
2. Then we apply Gaussian and Multinomial naive Bayes Classifier. Detailed code in Notebook.

   It happens that for test set, all the classifier reaches exactly the same result in prediction. I guess it is a coincidence. Because of the result, I find no difference between these classifiers.



```
             precision    recall  f1-score   support

        1.0       1.00      1.00      1.00         8
        2.0       1.00      1.00      1.00         6
        3.0       1.00      1.00      1.00         6
        4.0       0.71      0.83      0.77         6
        5.0       0.80      0.67      0.73         6
        6.0       1.00      1.00      1.00         6

avg / total       0.92      0.92      0.92        38

[[8 0 0 0 0 0]
 [0 6 0 0 0 0]
 [0 0 6 0 0 0]
 [0 0 0 5 1 0]
 [0 0 0 2 4 0]
 [0 0 0 0 0 6]]
```

Figure 13: Classification result using test data for multiple classes(Gaussian)



```
             precision    recall  f1-score   support

        1.0       1.00      1.00      1.00         8
        2.0       1.00      1.00      1.00         6
        3.0       1.00      1.00      1.00         6
        4.0       0.71      0.83      0.77         6
        5.0       0.80      0.67      0.73         6
        6.0       1.00      1.00      1.00         6

avg / total       0.92      0.92      0.92        38

[[8 0 0 0 0 0]
 [0 6 0 0 0 0]
 [0 0 6 0 0 0]
 [0 0 0 5 1 0]
 [0 0 0 2 4 0]
 [0 0 0 0 0 6]]
```

Figure 14: Classification result using test data for multiple classes(Multinomial)

Exercise # 2

Exercise # 2: (a)

RSS for linear and cubic regressions in training data (result is linear):
Although the true relationship is linear, there is no information that tells the result of RSS. It is possible for a overfitted cubic regression model get lower RSS since the train data is not very large (100 samples) and cubic regression has better flexibility. It is also possible for the linear regression gives a lower RSS because it is the correct fitting model and those points maybe just fit on the curve.

Exercise # 2: (b)

RSS for linear and cubic regressions in testing data (result is linear):
Given big enough testing data, the RSS for linear regression will be lower than the cubic regression because cubic regression will provide more error because of overfitting from train data.

Exercise # 2: (c)

RSS for linear and cubic regressions in training data (result is unknown):
The cubic one will give lower RSS because it has better flexibility comparing to the linear one because the true relationship is not known.

Exercise # 2: (d)

RSS for linear and cubic regressions in testing data (result is unknown):
No information can be told. We do not know how close the true model is to linear or cubic regressions. According to bias-variance tradeoff: there is never answers about what level of flexibility will perform a better fit for unknown data.

Exercise # 3

ISLR, 4.7.3: QDA model:
Based on the density function 4.11, since we do not assure that every variance are the same, we denote the variance of kth class as σ_k, then we have the posterior probability of x, $p_k(x)$:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}$$

And to prove the Quadratic relationship, we need to find $\delta_k(x)$, so we take log of the previous equation:

$$\log(p_k(x)) = \frac{\log(\pi_k)+\log(\frac{1}{\sqrt{2\pi}\sigma_k}) - \frac{1}{2\sigma_k^2}(x-\mu_k)^2}{\log(\sum_{l=1}^{k} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} \exp(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2))}$$

And we modify the equation to calculate $\delta_k(x)$:

$$\delta_k(x) = \log(p_k(x)) \log(\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} \exp(-\frac{1}{2\sigma_l^2}(x-\mu_l)^2)) = \log(\pi_k) + \log(\frac{1}{\sqrt{2\pi}\sigma_k}) - \frac{1}{2\sigma_k^2}(x-\mu_k)^2$$

From observation, we can conclude that $\delta_k(x)$ has a quadratic relationship with x, hence the classifier is quadratic.

Exercise # 4

Use the formula derived for bayes theorem under normal distribution:

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma_k} exp(-\frac{1}{2\sigma^2}(x-\mu_k)^2)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma_l} exp(-\frac{1}{2\sigma^2}(x-\mu_l)^2)}$$

And we have two classes: 1 (YES), and 0 (NO):

$$p_1(x) = \frac{\pi_1 \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_1)^2)}{\pi_1 \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_1)^2) + \pi_0 \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(x-\mu_0)^2)}$$

And we are given $\sigma = 6$, $\mu_1 = 10$, $\mu_2 = 0$, $\pi_1 = 0.8$, $\pi_0 = 0.2$ and x = 4:

$$p_1(4) = \frac{0.8 \frac{1}{\sqrt{2\pi}6} \exp(-\frac{1}{2*36}(4-10)^2)}{0.8 \frac{1}{\sqrt{2\pi}6} \exp(-\frac{1}{2*36}(4-10)^2) + 0.2 \frac{1}{\sqrt{2\pi}6} \exp(-\frac{1}{2*36}(4-0)^2)} = 75\%$$