McMaster University

Course Project Comparing Classifiers

SFWRTECH 4DA3 - Data Analytics and Big Data

Instructor:
Jeff Fortuna

Students:
Aiden WangYang Li – 400374502
Guang Ouyang – 400353742
Xiran Dong - 400346507

12-10-2022

Introduction

In this project, we applied Fisher's Linear Discriminant and Support Vector Machine classifiers to accelerometer data for classification tasks. Then we compared the results generated from those two classifiers. The computational times for both the training and testing processes are recorded. We also computed the corresponding confusion matrix from the results of each classifier.

Code Execution

Our program is written in Python programming language version 3.10. To execute the code, please open it with any compiler available to run pythons such as Visual Studio Code or PyCharm. Here are the steps that can guide you to the results calculated by the program.

1. Open the main.py file by using any compiler available to run pythons
2. Run the main.py without debug mode
3. The result will be displayed on the terminal screen

Accelerometer Dataset

The accelerometer dataset was generated from the vibrations of a cooler fan with weights on its blades. It can be used for predictions, classification and other tasks that require vibration analysis, especially in engines. (UCI Machine Learning Repository: Accelerometer Data Set, n.d.) [1] The dataset has 153000 instances. It has 3 classes and 4 attributes to categorize data. In this project, we focus on classes 1 and 2, and the attributes of x and y values. Class 1 and Class 2 have 51000 instances on each dataset, we took 38250 (75%) instances for training and 12750 (25%) instances for testing.

Fisher's Linear Discriminant Classification

Basic Algorithm

At first, we tried to determine the threshold by trial and error on the training set, and we got the threshold value of -0.00000119 has the minimum errors. Here are the results generated by the code:

```
threshold = -0.00000115     num of errors = 37130   Correct = 39370
threshold = -0.00000116     num of errors = 37127   Correct = 39373
threshold = -0.00000117     num of errors = 37116   Correct = 39384
threshold = -0.00000118     num of errors = 37124   Correct = 39376
threshold = -0.00000119     num of errors = 37109   Correct = 39391
threshold = -0.00000120     num of errors = 37110   Correct = 39390
threshold = -0.00000121     num of errors = 37127   Correct = 39373
threshold = -0.00000122     num of errors = 37134   Correct = 39366
threshold = -0.00000123     num of errors = 37150   Correct = 39350
```

Figure FLD-1 – Threshold Trial and Error Record

[1] UCI Machine Learning Repository: Accelerometer Data Set. (n.d.). http://archive.ics.uci.edu/ml/datasets/Accelerometer

We plot the training set on a graph, and we can see that those two groups of data almost overlap each other. It is a little bit hard to classify the data. We tried to use the best-case threshold value to draw the W line and the discriminant line on the graph, the result is as follows:

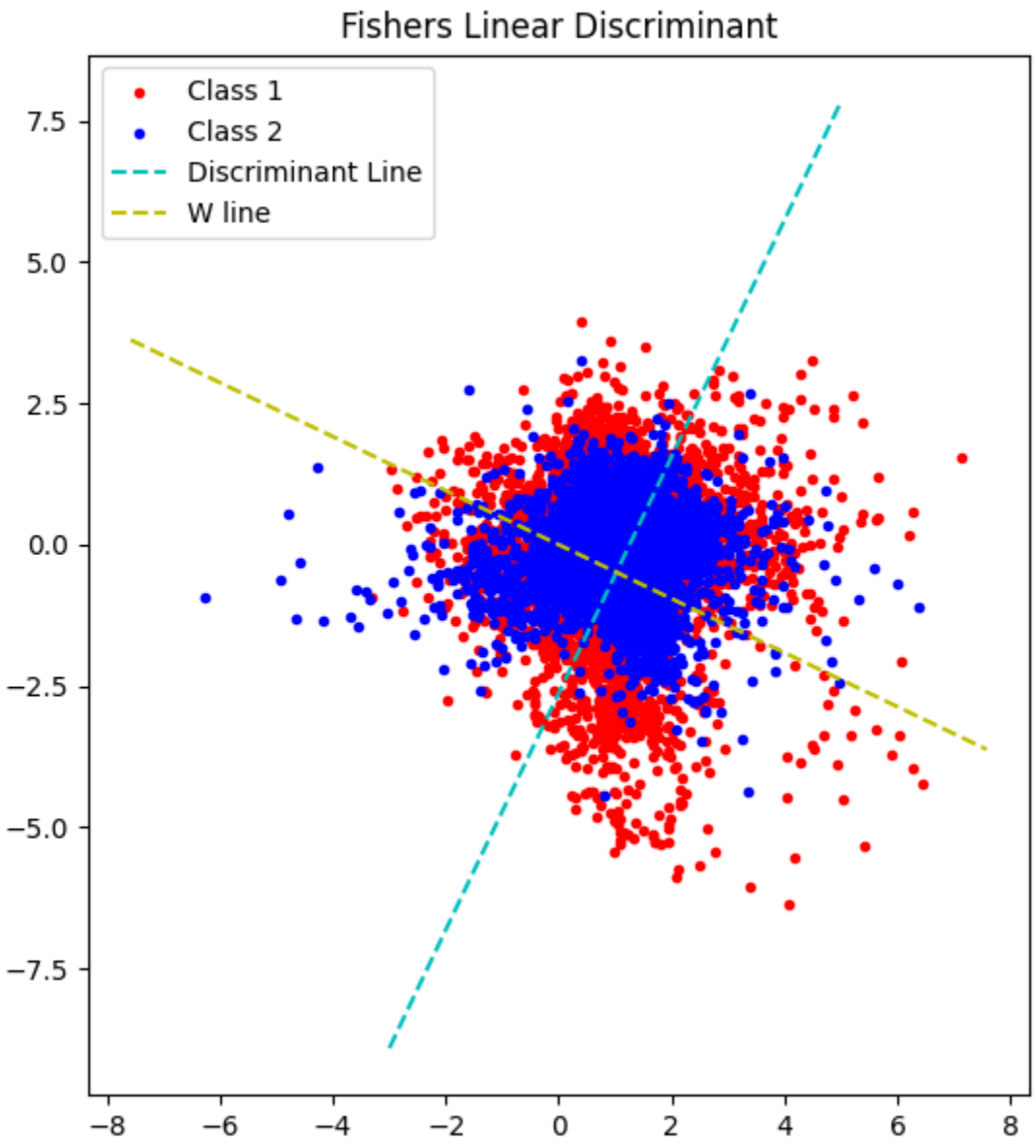

Figure FLD-2 – Fishers Linear Discriminant Classification

As we can see in the graph, the data of Class 1 in red are almost lying on the right side of the discriminant line, and the data of Class 2 in blue are almost lying on the left side of the line.

To make the classification more obvious, we have changed the data that lies on the wrong side to green. From the graph we can see, almost half of the data are mislabeled to the wrong category.

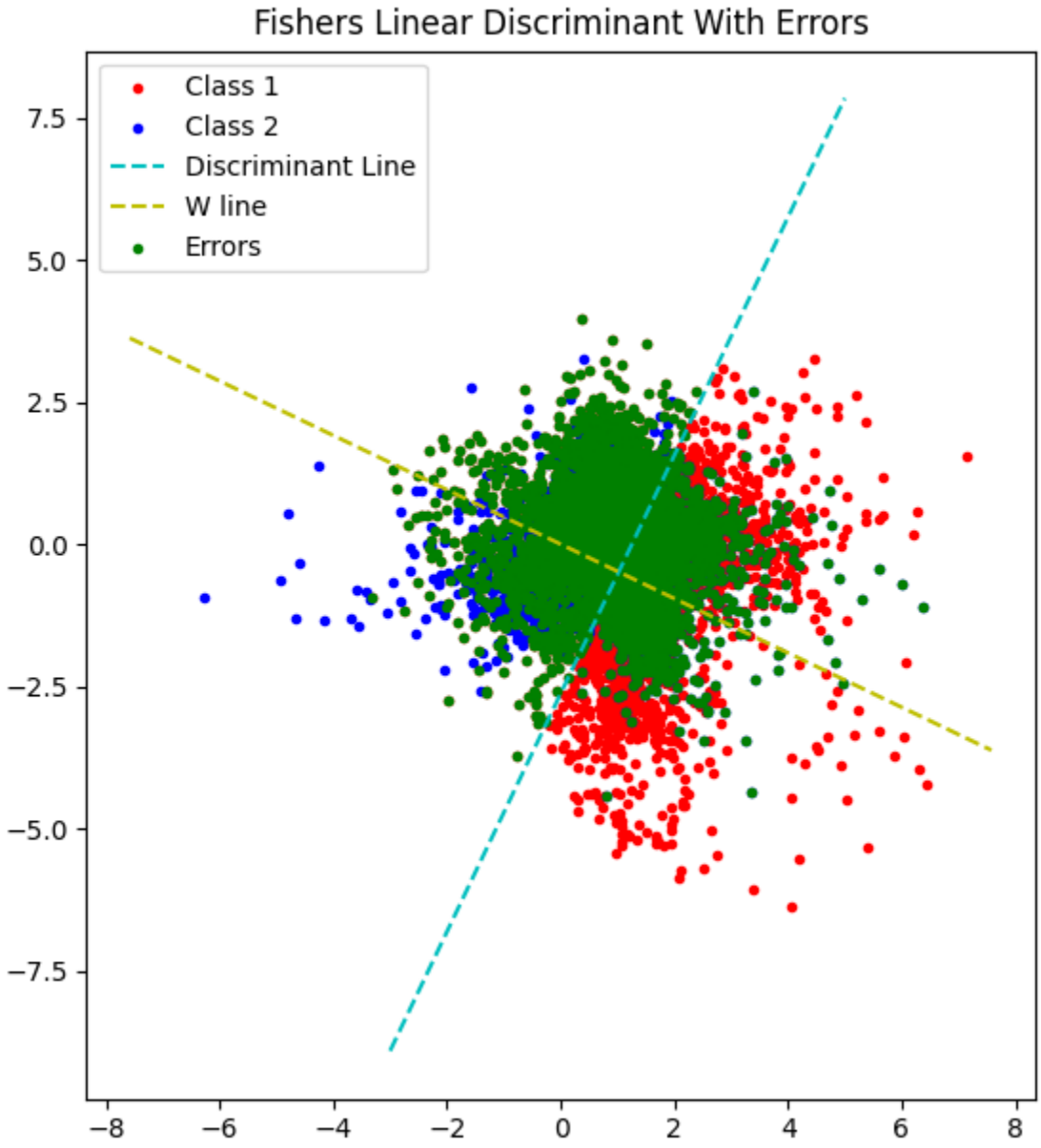

Figure FLD-3 Fishers Linear Discriminant Classification with Error Points

Results Generated from code

```
> Applying Fishers Linear Discriminant Classification
> Computational times for training data is 1.99 milliseconds
> Computational times for testing data is 1.00 milliseconds
True Positive:   5592
False Negative:  7158
True Negative:   7732
False Positive:  5018

> Applying Fishers Linear Discriminant Classification By Using Sklearn
> Computational Times for training data is 25.93 milliseconds
> Computational Times for Testing data is 1.00 milliseconds
True Positive:   3913
False Negative:  8837
True Negative:   10133
False Positive:  2617
```

Figure FLD-4 Results Generated from Code

Computational Times

From the results of the code, we recorded the computation time for the basic FLD algorithm as follows:

- Computational times for training data is 1.99 milliseconds
- Computational times for testing data is 1.00 milliseconds

We have also recorded the computation time by using Sklearn's FLD algorithm, the computational time is a little bit higher than the basic algorithm. We think the loading time from Sklearn's library may take some extra time.

- Computational times for training data is 25.93 milliseconds
- Computational times for testing data is 1.00 milliseconds

Confusion Matrix

Based on the results generated from the testing dataset using the basic FLD algorithm, we can plot the confusion matrix as follows:

Class B misclassified as Class A (**False positive**) **5018 (39.36%)**	Class A correctly classified (**True positive**) **5592 (43.86%)**
Class B correctly classified (**True negative**) **7732 (60.64%)**	Class A misclassified as Class B (**False negative**) **7158 (56.14%)**

We can see that the predictions are very similar to what we saw from the graph earlier. Nearly half of the data was mislabeled. Most of class A (56.14%) data are misclassified, and a small part of class B (39.36%) data are misclassified. The proportion of correctly classified data is (7732 + 5592) / (12750*2) * 100% = 52.25%.

Here is the confusion matrix generated from the testing dataset using Sklearn's Library.

Class B misclassified as Class A (**False positive**) **2617 (20.53%)**	Class A correctly classified (**True positive**) **3913 (30.69%)**
Class B correctly classified (**True negative**) **10133 (79.47%)**	Class A misclassified as Class B (**False negative**) **8837 (69.31%)**

In Sklearn's confusion matrix results, we can see that the result is more biased towards correctly classifying Class B. The proportion of correctly classified data is (10133 + 3913) / (12750*2) * 100% = 55.08%. The result is slightly better than our basic FLD algorithm. But we cannot say this method is better. We still need more testing datasets to judge and measure our computational methods.

Linear Support Vector Machine

Sklearn's Linear Support Vector Machine Algorithm

In this section, we applied Sklearn's Support Vector Machine algorithm directly to the datasets for the purpose of convenience. The linear SVM and BSF SVM are used to compare the results.

As we know, the support vector machine algorithm is difficult to classify data that has classes that overlap with each other. Because it uses the margin of a linear classifier between the two datasets. But when the data sets have almost the same distribution trend, it is almost impossible to find the margin.

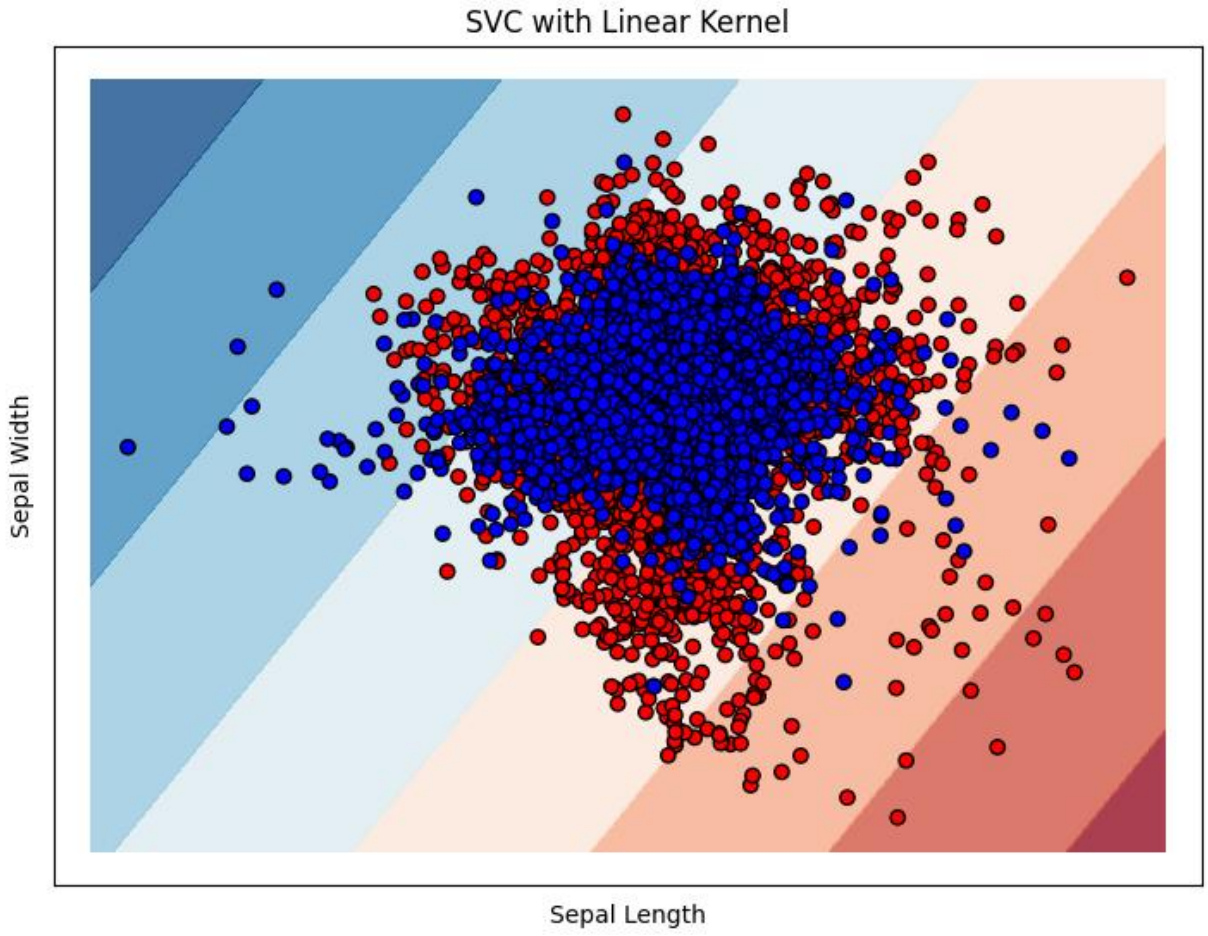

Figure SVM-1 - Linear Support Vector Machine Classification

But Sklearn finally provided us with a relatively feasible result through a lot of calculations. The calculation time for the graph plotting is almost 10 minutes.

```
> Computational times for plotting the graph is 488885.50 milliseconds
```

Non-linear (RBF) Support Vector Machine

We also tried to use the radial basis function (RBF) kernel for the classification of the support vector machine algorithm. It provides a better calculation by placing a radial basis function centred at each point, then performing linear manipulations to map points to higher-dimensional spaces that are easier to separate. (Ye, 2022)[2] We can get better results by using the quadratic lines to classify the datasets. The image below shows what RBF classification looks like:

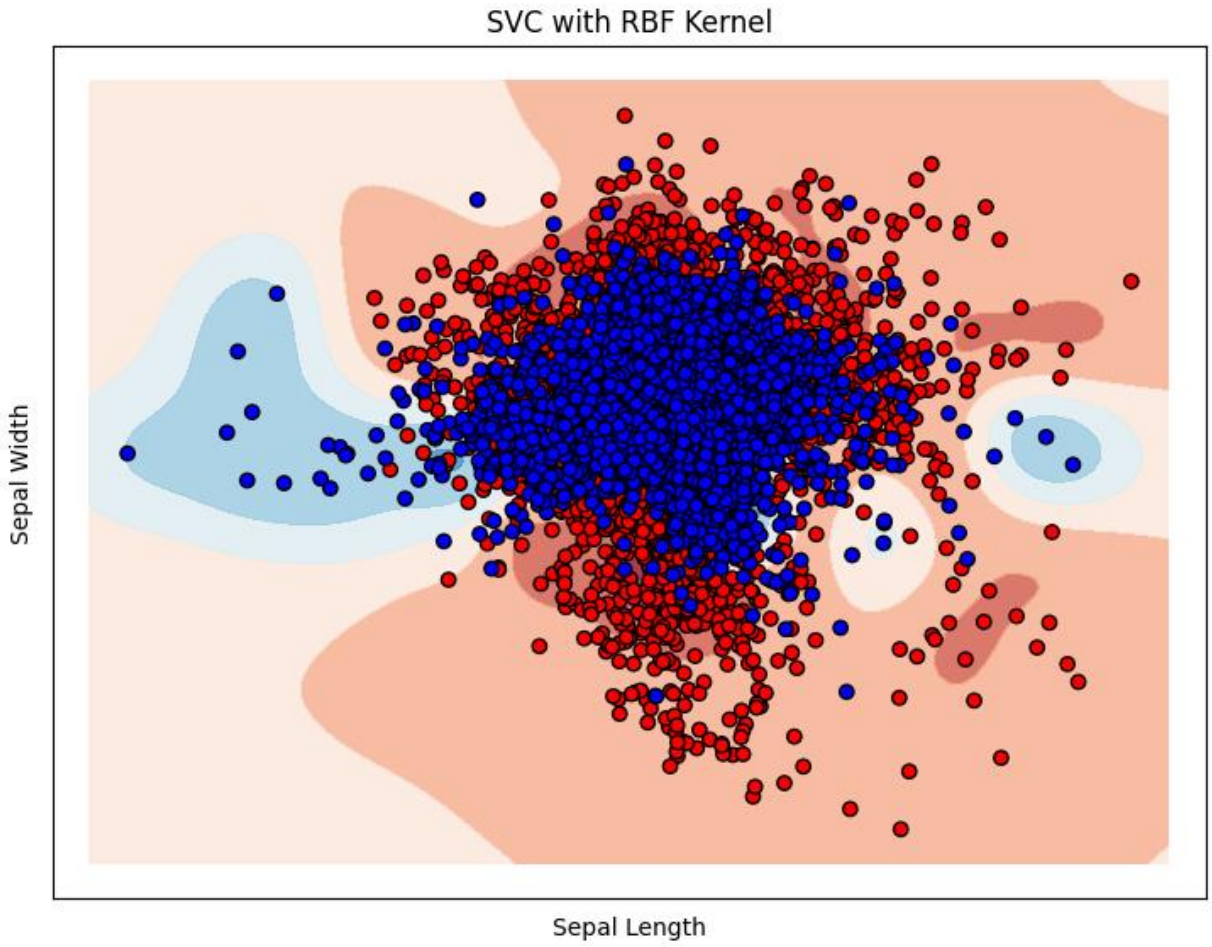

Figure SVM-2 - RBF Support Vector Machine Classification

```
> Computational times for plotting the graph is 527802.68 milliseconds
```

This RBF support vector machine algorithm indeed takes more time to do the calculation.

[2] Ye, A. (2022, January 5). Radial Basis Functions, RBF Kernels, & RBF Networks Explained Simply. Medium. https://medium.com/dataseries/radial-basis-functions-rbf-kernels-rbf-networks-explained-simply-35b246c4b76c

Results Generated from code

```
> Applying Linear Support Vector Machines By Using Sklearn
> Computational Times for training data is 88239.11 milliseconds
True Positive:   4328
False Negative:  8422
True Negative:   9596
False Positive:  3154
> Computational Times for Testing data is 23518.11 milliseconds

> Applying RBF Support Vector Machines By Using Sklearn
> Computational Times for training data is 172186.68 milliseconds
True Positive:   9726
False Negative:  3024
True Negative:   4644
False Positive:  8106
> Computational Times for Testing data is 124272.65 milliseconds
```

Computational Times

From the results of the code, we recorded the computation time for the Linear SVM algorithm by using Sklearn as follows:

- Computational times for training data is 88239.11 milliseconds
- Computational times for testing data is 23518.11 milliseconds

The computation time for the RBF SVM algorithm by using Sklearn:

- Computational times for training data is 172186.68 milliseconds
- Computational times for testing data is 124272.65 milliseconds

Confusion Matrix

The confusion matrix for the linear support vector machine algorithm:

Class B misclassified as Class A (**False positive**) **3154 (24.74%)**	Class A correctly classified (**True positive**) **4328 (33.95%)**
Class B correctly classified (**True negative**) **9596 (75.26%)**	Class A misclassified as Class B (**False negative**) **8422 (66.05%)**

The result is more biased towards correctly classifying Class B. The proportion of correctly classified data is (4328 + 9596) / (12750*2) * 100% = 54.60%.

The confusion matrix for the RBF support vector machine algorithm:

Class B misclassified as Class A (**False positive**) **8106 (63.58%)**	Class A correctly classified (**True positive**) **9726 (76.28%)**
Class B correctly classified (**True negative**) **4644 (36.42%)**	Class A misclassified as Class B (**False negative**) **3024 (23.71%)**

The result is more biased towards correctly classifying Class A. The proportion of correctly classified data is (9726 + 4644) / (12750*2) * 100% = 56.35%.

Comparison of Two Classifiers

By the properties of the classifiers, Fisher's linear discriminant algorithm focuses on all data points to do the classification. Whereas the support vector machine algorithm focuses on the points that are difficult to classify. Since the distribution laws of the two datasets are superimposed, SVM needs more effort to classify than FLD.

From the time perspective, SVM needs to spend a lot of time calculating and processing these overlapping data from different classes. We recorded almost 10 minutes of calculation time for the SVM classifier, but FLD only needs a few milliseconds to perform the calculation. SVMs fail miserably in this regard.

From the accuracy perspective, the results calculated by SVM are slightly better than FLD. The results were correctly classified by FLD as 52.25% and 55.08%. On the other hand, we have the results correctly classified by SVM as 54.60% and 56.35%. Therefore, SVM's classifier is slightly better in classification results.

Conclusion

From the datasets we have trained and tested, FLD takes significantly less time to do the computation but SVM's classifier yields a slightly better result in accuracy. The difference in the computational time is because there is a large number of data points from the 2 classes that are overlapping,

FLD focuses on all data points, and the results can be easily calculated by the algorithm which is to maximize the distance of the projected means and to minimize the projected within-class variance. It is a more universally applicable algorithm with a slightly lower accuracy compared to other algorithms.

SVM focuses on the data that are hard to classify. When the datasets have many overlapping data points in our case, it may need to calculate and compare a tremendous number of support vectors. Which could take a very long time to do the training process. This algorithm is better to be used in a high-denominational space with more separated data points.

Overall, to get better accuracy on the training result, we may need to use more complex data training algorithms like SVM. As we always say, to achieve better results, machine learning takes a long time to train.