Assignment 2
ECE 657A Winter 2022
Group 1

Jehanzeb Mirza
21115879

Mehmet Ali Pelit
21113622

2024-03-25

Introduction

Our objective for this assignment is to understand and apply methods discussed in our lectures for classification. We are provided three datasets *Dataset A*, *Dataset B*, and *Dataset C* to analyze and process. There are three specific goals for this lab covered in the following sections.

1. Nonlinear Dimensionality Reduction
2. Binary Classification
3. Multi-Class Classification

Our invesitgation makes use of code written in Python and the source files, generated figures, and datasets are provided as supplementary files in addition to this report. We extensively use Numpy [1] and Pandas libraries for data management and Sci-kit learn's [2] implementations of algorithms.

1 Nonlinear Dimensionality Reduction

For this section we investigate *Dataset C* provided to us as a comma-seperated-value file `DataC.csv`. The file contains 2066 labeled samples representing hand-written digits of the numbers 0-4. There are 784 features with each feature corresponding to the gray-scale 8 bit value of a pixel on a 28x28 image of the digit. Our goal is to apply nonlinear dimensionality reduction techniques to reduce the number of features down to 4.

For our K nearest neighbours in the following we choose K=5.

1.1 Locally Linear Embedding

We begin by investigating Locally Linear Embedding (LLE). LLE is an unsupervised dimensionality reduction technique that transforms data based on a locally linear reconstruction.

In words, the technique starts by finding the best linear approximations for each point based on its neighbouring K nearest points and then finding the best set of projected points to minimize the distances between a projected point and its reconstruction from the same linear approximation in the original space.

After filtering our dataset to only the digits labelled as "3" and applying LLE we investigate the first two projected dimensions to understand any characteristics.

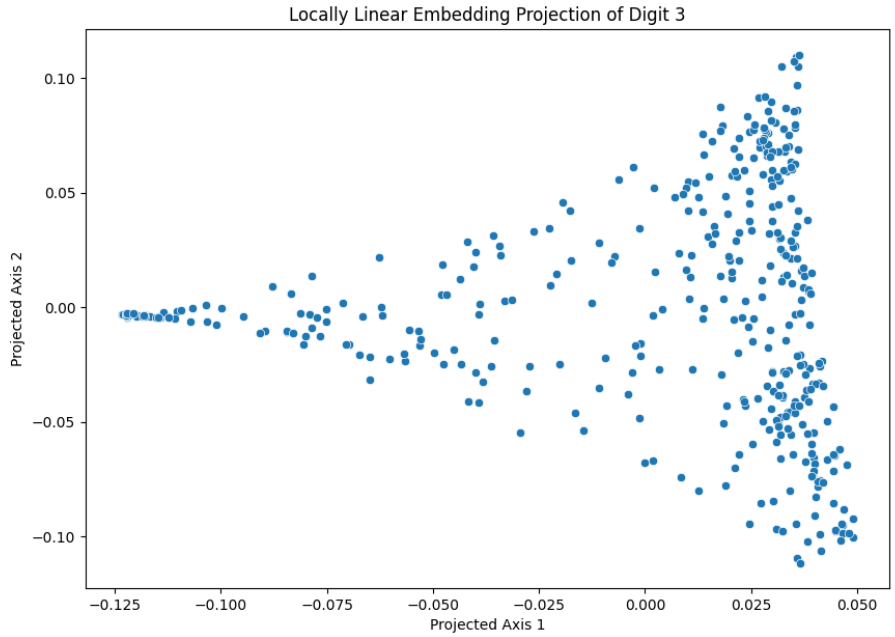


Figure 1: Scatterplot of Locally Linear Embedding of "3"

As we can see the points form an interesting triangular shape in Figure 1 which suggests that there may be some triangular structure present in the 784 dimensional space.

We then take many of these points and in the location they are present on the scatterplot paste the image of the original digit, shown in Figure 2. The idea is to see if visually each of the projected dimensions has some meaningful interpertation.

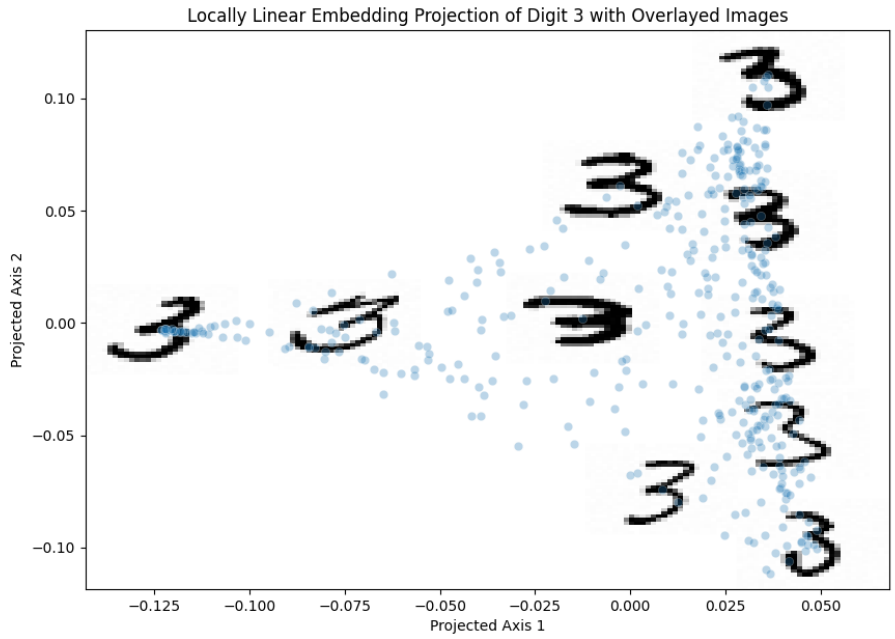


Figure 2: Scatterplot of Locally Linear Embedding of "3" with Superimposed Images

Unforunately, as it turns out there isn't really a very clear pattern with respect to what either of the projected dimensions represents. However it can vaguely be determined that perhaps the first projected dimension corresponds to how tilted the 3 is and up and down determined how spread out / large the 3 is.

1.2 ISOMAP

In this part we repeat the same analysis as performed above however instead we use the ISOMAP technique.

ISOMAP is a multidimensional scaling (MDS) technique that starts by connecting each point to its K nearest neigbours to form a graph. We then find the geodesic distance between all pairs of points as the shortest connected path between each pair.

Our projected set is the set of points that attempts to preseve a similar geodesic distance.

The results of the same analysis as above are the following Figures 3, 4.

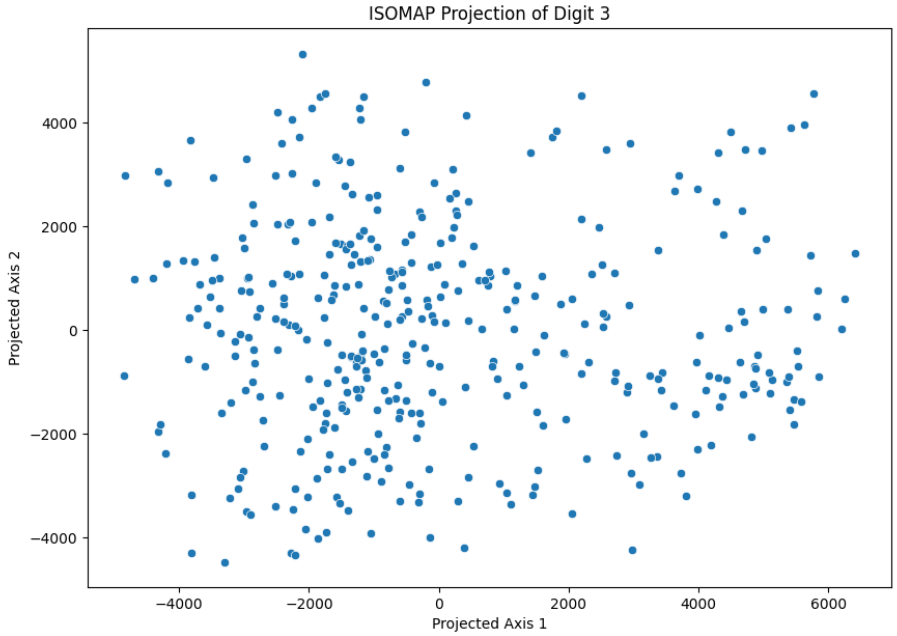


Figure 3: Scatterplot of ISOMAP of "3"

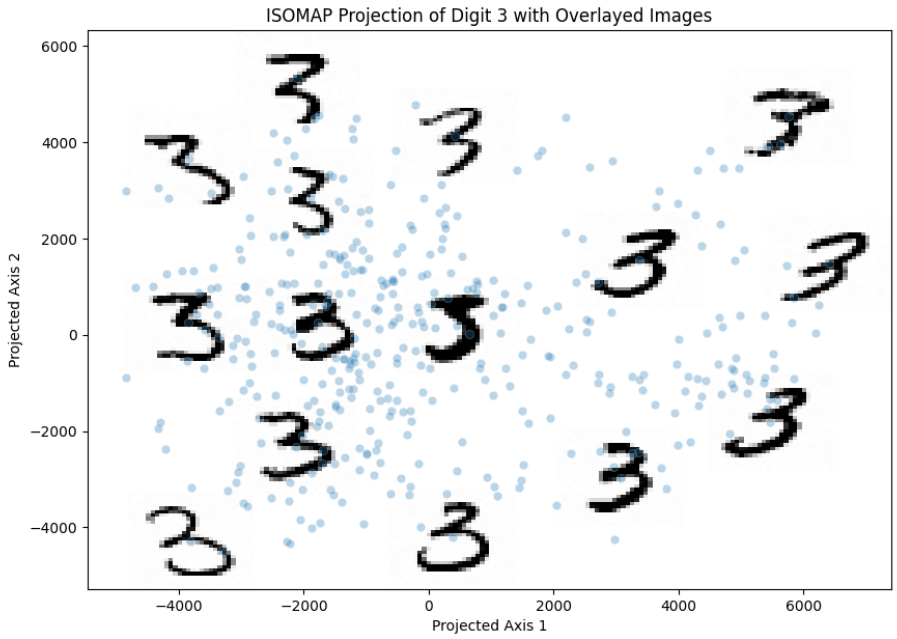


Figure 4: Scatterplot of ISOMAP of "3" with Superimposed Images

ISOMAP produces a lot less interesting of a shape where the points are instead spread out into a uninteresting circular blob pattern.

This time there is a significantly clearer qualitative comparision in that the ISOMAP result shows how the 3s become more skewed with an increase in projected axis 1.

Additionally, the 3's near the bottom have more full curly bottoms while the 3s at the top have no bottom at all.

ISOMAP is a more global technique as it attempts to preserve distances in a more global scale pairwise between every pair instead of just reconstruction error in the local neighbourhood.

1.3 Naive Bayes on Projected Data

We finally conclude by investigating how well the dimensionality reduction techniques seperate the various classes by training a Gaussian Naive bayes classifier on the projected features on 70% of the samples and testing it on

the remaining 30%. We primarily compare average accuracy to determine which is best.

The dimensionality reduction we compare in this part are two linear methods : Principle Component Analysis (PCA) and Linear Discriminant Analysis (LDA), as well as the two non-linear ones of ISOMAP and LLE from above.

We repeat the experiment 30 times and take the average accuracy and present the results below. We chose 30 trials because it was a good balance between modest runtime for the code and how much variance impacts our results. We compute the standard deviation in accuracy to ensure that enough trials have been run that the differences between the means is reasonably outside the standard deviation between methods.

Projection Method	LLE	ISOMAP	LDA	PCA
Accuracy (%)	90.8 ± 2.1	87.8 ± 1.7	86.1 ± 1.4	83.8 ± 1.1

Emprically we find LLE to be the best, with ISOMAP and LDA being comparable, while PCA was the least effective.

2 Binary Classification

In this part of the report, the Dataset A is classified using four classifiers: k-NN, SVM with RBF kernel, Naive Bayes, and Decision Tree. The effect of parameters in classifiers is experimented. By looping through each possible parameter, we try to find the best accuracy for the associated classifier. After finding the most convenient parameters, the performance of the classifiers is compared according to some of metrics like *Accuracy*, *Precision*, *Recall* and $F1 - Score$.

The Dataset A has been given us in a *csv* file. The data defines DNA sequences with 2200 samples. There are 57 different features and the labels are either +1 or -1. Hence, the classification is considered as binary classification.

2.1 Preprocess of Data

We randomly split the data into two parts for training and testing. The testing data consists of 30 percent of the samples. For the partition, seed of 42 is used as the random state.

2.2 Parameter Selection for k-NN

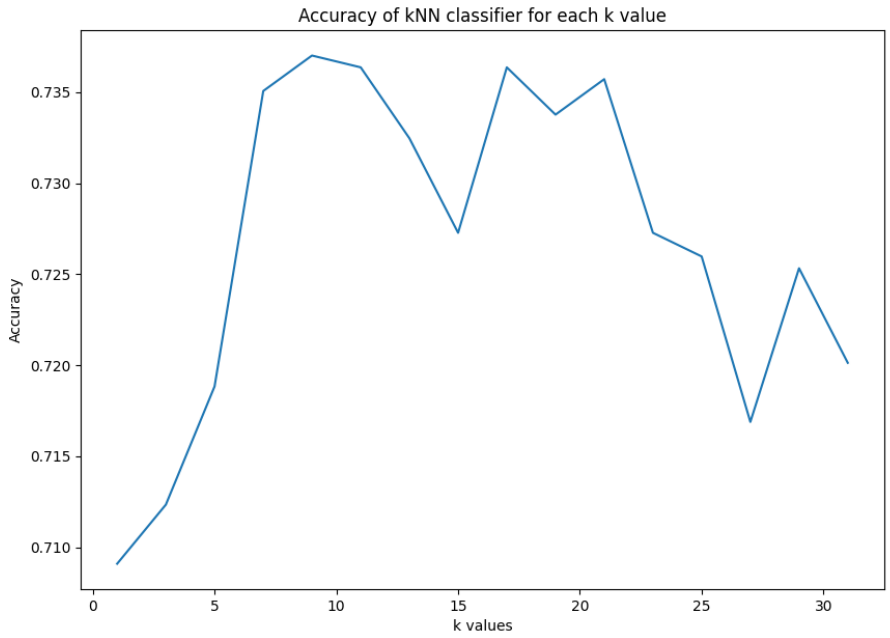


Figure 5: Accuracy of each parameter value of K for the Classifier k-NN

We applied cross-validations with 5-fold on k-NN using varying parameter k. It is selected from the set [1, 3, 5, 7, ..., 31]. In Figure 5, accuracy of each assoctiated classifier is depicted. The best k resulting the most accurate classification is found as 9.

Max accuracy	**K**
0.7370	9

2.3 Parameter Selection for RBF Kernel SVM

In SVM with RBF kernel, there are some parameters that affect the boundry's shape. The soft margin penalty term, c, and kernel width, $gamma$, are selected from respective sets [0.1, 0.5, 1, 2, 5, 10, 20, 50], and [0.01, 0.05, 0.1,

0.5, 1, 2, 5, 10]. Accuracy of the SVM model tested for every c and *gamma* values. The best resulting c and *gamma* are below.

Max accuracy	**C**	**gamma**
0.9052	10	0.01

2.4 Comparision of Classifiers

After picking the best k for k-NN and *c-gamma* for SVM RBF, we went to the comparision stage where the performance of four classifiers are tested. Other two classifiers are Naive Bayes classifier and Decision Tree with default setups. Models are trained and tested with new random sets for 20 times. The average and standard deviations of the metrics *Accuracy*, *Precision*, *Recall* and $F1Score$ are below.

	Accuracy		**Precision**		**Recall**		**F1-Score**	
	Avg	*Std*	*Avg*	*Std*	*Avg*	*Std*	*Avg*	*Std*
k-NN	0.7519	0.0188	0.8120	0.0116	0.7519	0.0188	0.7415	0.0209
SVM RBF	0.9040	0.0134	0.9055	0.0125	0.9040	0.0134	0.9040	0.0134
Naive Bayes	0.8690	0.0101	0.8695	0.0102	0.8690	0.0101	0.8690	0.0102
Decision Tree	0.9289	0.0088	0.9293	0.0086	0.9289	0.0088	0.9289	0.0088

Table 1: The performance of the classifiers according to 4 metrics

2.5 Comments

According to numbers on Table 1, trained Decision Tree model had the best performance among all. Decision Trees are good at handling both complex data as in DNA sequences. Likewise, RBF kernel enables SVM to be effective for non-linear data. That's why, it makes SVM-RBF a good classifier after Decision Trees. The next good performer is the Naive Bayes classifier. k-th Nearest Neighbors model becomes the worst classifier of this performance test. It may be because of the complexity of the dataset since DNA sequences are considered complex structures.

3 Multi-Class Classification

For this section we investigate *Dataset B* provided to us as a comma-seperated-value file `DataB1.csv`. The dataset contains 150 samples of iris data where it is labeled by one of 3 iris types (Setosa, Versicolour, and Virginica) with 4 different features of different lengths: Sepal length, Sepal width, Petal length, Petal width.

We investigate classification techniques that can classify between multiple classes.

3.1 One-vs-all vs One-vs-One

In this section we investigate two modes of using support vector machines (SVM) in order to perform multi-class classification.

- One-vs-all
- One-vs-one

Classically a support vector machine only supports binary labels corresponding to ± 1.

In one-vs-one mode, we produce $\binom{N}{2}$ different different SVM classifiers where each is trained only to seperate between two classes. Then when running inference, we use voting where the class that is most voted by each classifier is chosen as the result class.

In one-vs-all mode, we produce N different SVM classifiers where one class is labeled as +1 and all others are -1. We then choose whichever classifier is most confident as the result class during inference. (Has the most positive result / biggest marigin).

3.2 Comparing OvA and OvO SVM

Once again we use a number of random 70-30 training test splits of our data and test SVM classifiers in both OvO mode and OvA. We run the experiment 30 times which still runs quite quickly because the model is fast.

We compute Accuracy, Precision, Recall, F1-Score, and the summed confusion matrix (sum over all trials).

Table 2: Confusion Matrix OvO

True Label	**setosa**	**versicolor**	**virginica**
setosa	429	0	0
versicolor	0	435	29
virginica	0	8	449

Table 3: Confusion Matrix OvA

True Label	**setosa**	**versicolor**	**virginica**
setosa	429	0	0
versicolor	0	435	29
virginica	0	8	449

SVM Method	Accuracy	Precision	Recall	F1-Score
OvO	97.3 ± 2.3	97.4 ± 2.1	97.3 ± 2.3	97.3 ± 2.3
OvA	97.3 ± 2.3	97.4 ± 2.1	97.3 ± 2.3	97.3 ± 2.3

Something we find is that both methods produce exactly the same results. We sort of expected this since the difference between the two for only 3 categories should not be substantial, especially when using simple linear kernels.

3.3 Decision Tree

In comparision to the two SVM techniques, Decision trees can natively perform classification over multiple classes as information gain and gini impurity are already definied for multiple classes without any voting or tiebreaking or ensemble techniques required. In order to decide on what feature to split we

Table 4: Confusion Matrix Decision Tree

True Label	**setosa**	**versicolor**	**virginica**
setosa	429	0	0
versicolor	0	421	43
virginica	0	34	423

simply choose whichever maximizes the particular objective we choose.

We use a decision tree built with default parameters to classify the same Dataset B and it yields the following results:

Method	Accuracy	Precision	Recall	F1-Score
Decision Tree	94.3 ± 3.0	94.5 ± 3.0	94.3 ± 3.0	94.3 ± 3.0

The decision tree performs a bit worse than our SVM techniques, but still very well. This could be the result of simple overfitting which as we did not tune the parameters at all is likely to be an issue.

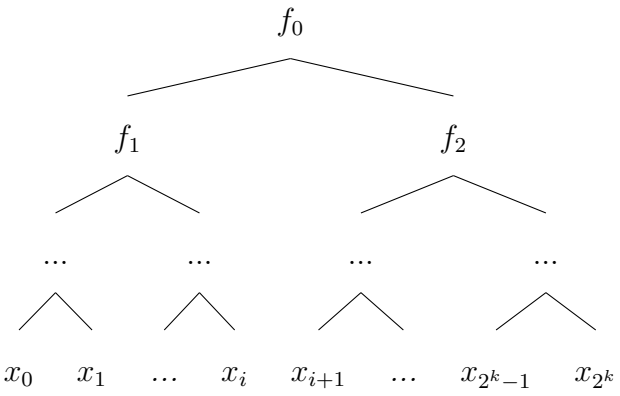
4 Theoretical Questions

4.1 Decision Tree

Suppose we have a data set with k features and N samples and $N \gg k$ then:

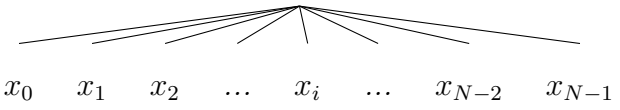
Part a. ***In the event of discrete features (2 category only), what is the highest number of leaves in decision tree?***

If the samples do not overlap each other such that causing filling every binary branch of the splits, then the resulting decision tree would be a perfectly complete binary tree. Since the number of samples, N, is significantly larger than the number of features, k, we expect 2^k many leaves as the highest number. (Assuming $N \gg 2^k$ as well)



Part b. ***In the event of continuous features, what is the highest number of leaves in decision tree?***

For continous features, theoretically, the data can be split uniquely for every sample. Thus, the maximum number of leaves would be N, where each leaf corresponds to a single sample and leading to an overfit model.



4.2 Spam Emails

You are classifying emails as legitimate or spam and in doing so you would like to estimate the probability that a new email e containing the keywords (w1 , w2 , ..., wn) is spam by taking all the emails in the training set with those keywords. In order words the probability is calculated as:

$$P(spam|e) = \frac{Number\ of\ spam\ emails\ with\ keywords(w_1, w_2, ..., w_n)}{Number\ of\ total\ emails\ with\ keywords(w_1, w_2, ..., w_n)} \quad (1)$$

Part a. ***Explain why the plan may not work.***

The reason that this plan may not work is because when you have even just a modest number of keywords that you are selecting, (suppose just 30) then there are many possible combinations of keywords appearing/not appearing

(2^{30}) and its quite likely that a new email will not exactly match with the same keywords present as in the training set.

The function will likely turn into a 0/0 where the keywords don't exactly match anything in the training set.

Part b. ***Describe the data sets for which this plan might work.***

The plan might work as long as the number of keywords that you are choosing is small, for example 5 to 10. This way there should be sufficient training samples of emails with spam/not spam that match all the keywords in a new email and thus you can accuractly calculate a probability.

Part c. ***Using Naive Bayes assumption, explain how to get the probability of an email being spam? Show it mathematically.***

The probability that an email e is spam given its keywords can be expressed as:

$$P(spam|e) = P(spam)\frac{\prod_{i=1}^{n} P(w_i|spam)}{\prod_{i=1}^{n} P(w_i)} \tag{2}$$

Part d. ***How does Naive Bayes assumption caters the problem with your plan?***

The naive bayes assumption fixes the problem with the plan because it allows you to consider each keyword independently so you no longer have the requirement that each possible combination of keywords needs to be represented but instead only that each keyword is.

4.3 SVM with Quadratic Kernel

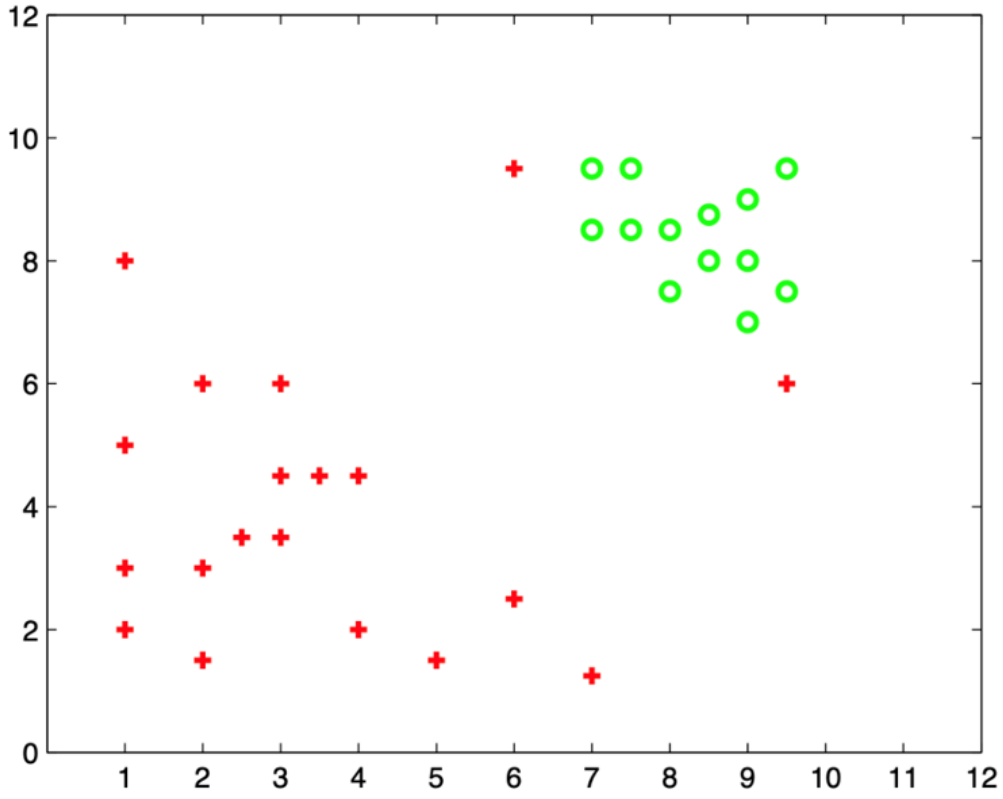


Figure 6: Plot

For the following questions, consider that we are using SVM with quadratic kernel and Figure 6.

Part a. ***When slack penalty (C) is large, draw the decision boundary. (Consider $C \to \infty$)***

Part b. ***When slack penalty (C) is small, draw the decision boundary. (Consider $C \approx 0$)***

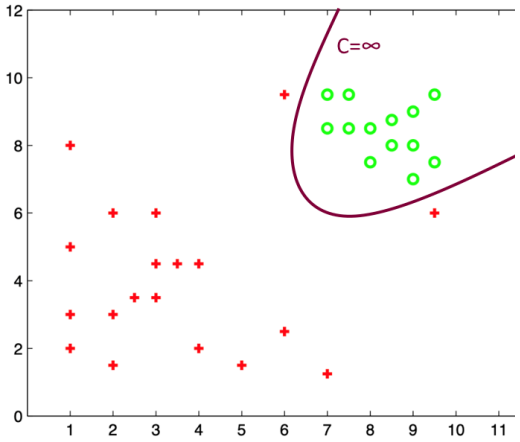


Figure 7: Boundary when C $\to \infty$

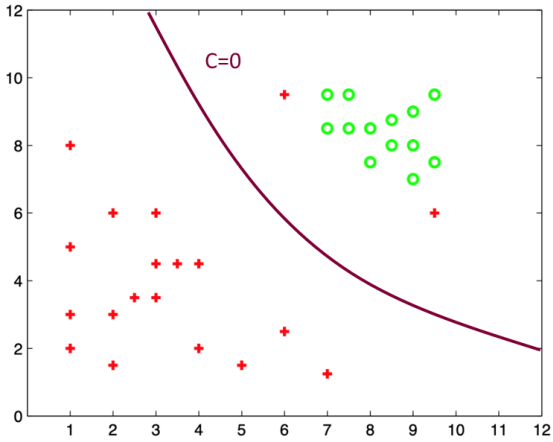


Figure 8: Boundary when C ≈ 0

Part c. ***Which one of the above will give better results in terms of classification.***

Larger penalty can cause overfitting which may not generalize well to new data.

Part d. ***When C is large, draw a data point which will not alter the learnt decision boundary. Give your reasons.***

Part e. ***When C is large, draw a data point which will alter the decision boundary learnt drastically. Give your reasons.***

The new point in Figure 9 is blended in the previous data points. Hence it won't cause any change on the decision boundary since it is not a "support vector". However, the point in Figure 10 has a distance from the cluster of prior data. Due to large penalty C, the boundary will be forced to include the new point as a support vector. It should be realized that new sample

hasn't blended with the other category so that the boundary doesn't neglect it.

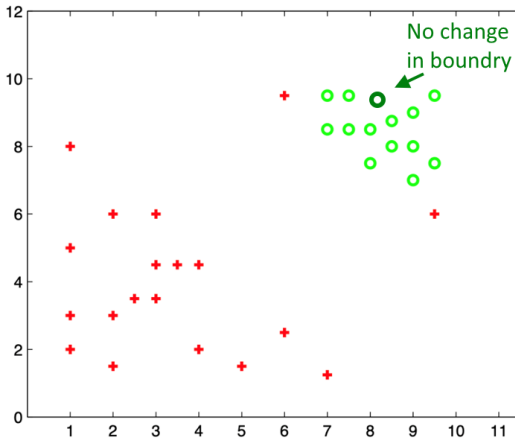


Figure 9: New ineffective data point

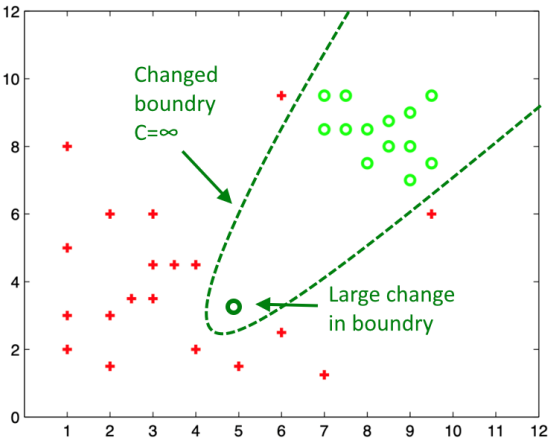


Figure 10: New effective data point

Conclusion

In conclusion we were successfully able to meet our objectives for this lab and developed experience applying various classification algorithms.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duch-

esnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.