Middle East Technical University Northern Cyprus Campus
Computer Engineering Program

CNG 409 Special Topics: Introduction to Machine Learning
Fall 2021-2022

Homework 3
Decision Trees and Support Vector Machines

2243392 Pınar Dilbaz

Chapter 1

INTRODUCTION

Thanks to this assignment, I had the opportunity to code the decision trees and support vector machines that we learned theoretically in the course. I had difficulties in visualizing decison trees. Although the algorithms are completely understood in theory, the coding part can be a little different and difficult, and this assignment was very useful for me as I had the opportunity to code the algorithms. In the first part, we were expected to fill in the empty functions in dt.py, I had no problems with this and I believe it works correctly, but I had a lot of difficulty in the visualization and accuracy calculation part. In the second part, we were expected to code the support vector machine(SVM) algorithm, there are 4 tasks together in svm.py. We were expected to use scikit-learn in this part, so I could complete my homework faster and easier. I had the chance to examine how hyperparameters(kernel, c value, gamma value) affect accuracy for SVMs. In addition, I also observed how changes in c value, gamma value and kernel change accuracy. As a result, it has been a quite enjoyable and instructive assignment for me.

Chapter 2

DECISION TREES

2.1 Information Gain

2.2 Average Gini Index

2.3 Information Gain with Chi-Squared Prepruning

2.4 Average Gini Index with Chi-squared Preprun-ing

Chapter 3

SUPPORT VECTOR MACHINES

3.1 Task1

Small c allows constraints to be easily ignored, so, causes large margins, in other hands, large c makes constraints hard to ignore, so, causes small margins. As we can see when we examine the figures below, most classifications will be done correctly if we can choose a large enough c value, but if a very small value of c is chosen, most will be misclassified.

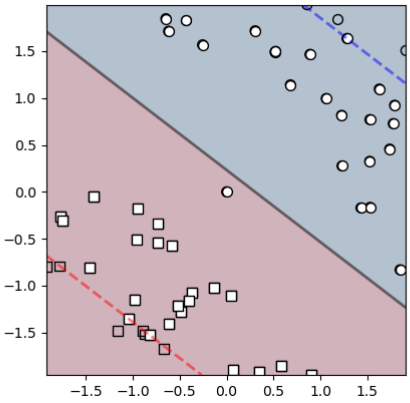

Figure 3.1: C value is 0.01

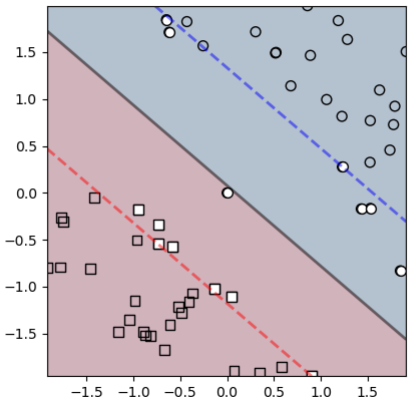

Figure 3.2: C value is 0.1

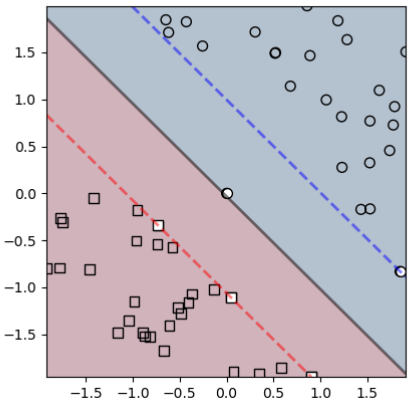

Figure 3.3: C value is 1

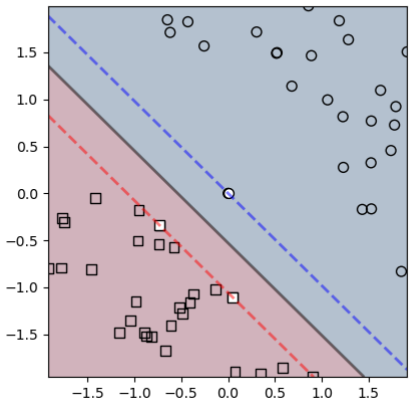

Figure 3.4: C value is 10

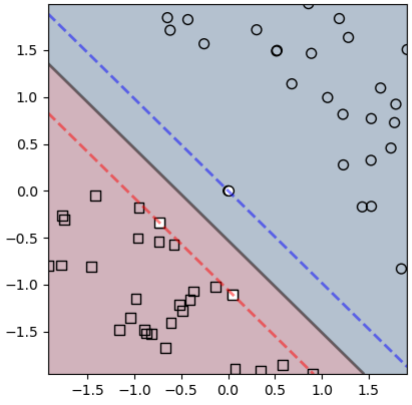

Figure 3.5: C value is 100

3.2 Task2

For this part we used a 2D non-linearly separable dataset. Therefore, as we can see in the figures below, the most suitable kernel is "Rbf".

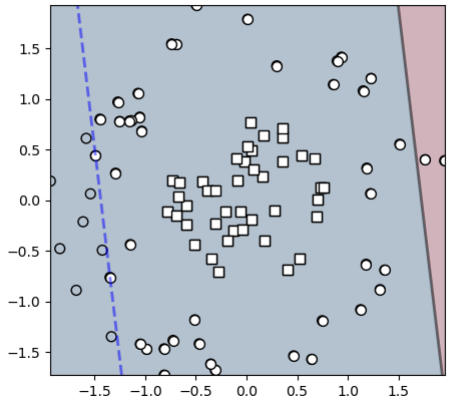

Figure 3.6: Kernel is "Linear"

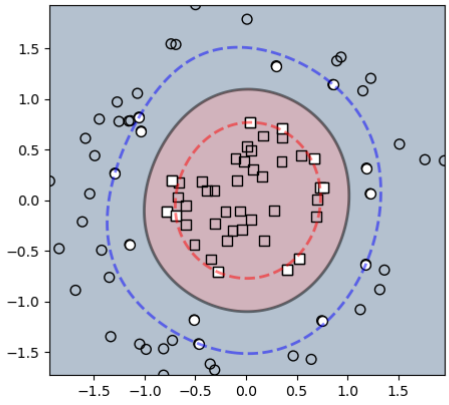

Figure 3.7: Kernel is "Rbf"

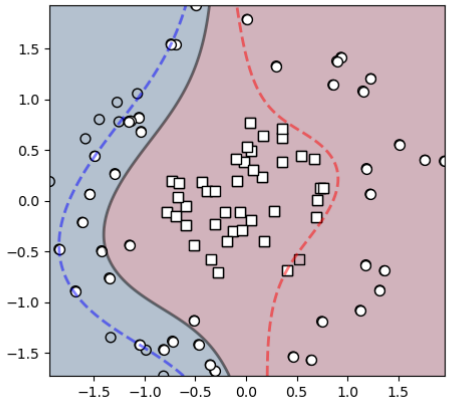

Figure 3.8: Kernel is "Polynomial"

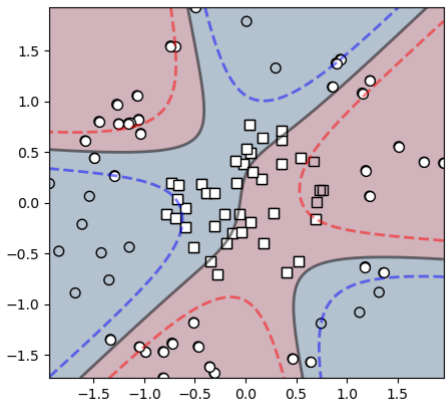

Figure 3.9: Kernel is "Sigmoid"

3.3 Task3

For each hyperparameter configuration, you can see the validation accuracy in the form of tables below. There are 65 different hyperparameter configurations in the table.

Best hyperparameter configuration

- **Kernel:** "Linear"
- **C Value:** 100
- **Gamma Value:** 0.1

⇒ Test Accuracy is **0.749**

Kernel	*C Value*	*Gamma Value*	*Validation Accuracy*
Linear	0.01	-	0.693
Linear	0.1	-	0.693
Linear	1	-	0.693
Linear	10	-	0.705
Linear	100	-	0.762
Rbf	0.01	0.00001	0.631
Rbf	0.01	0.0001	0.692
Rbf	0.01	0.001	0.694
Rbf	0.01	0.01	0.693
Rbf	0.01	0.1	0.693
Rbf	0.1	0.00001	0.631
Rbf	0.1	0.0001	0.692
Rbf	0.1	0.001	0.694
Rbf	0.1	0.01	0.693
Rbf	0.1	0.1	0.693
Rbf	1	0.00001	0.631
Rbf	1	0.0001	0.692
Rbf	1	0.001	0.694
Rbf	1	0.01	0.693
Rbf	1	0.1	0.693

Figure 3.10: Table of Accuracy

Rbf	10	0.00001	0.631
Rbf	10	0.0001	0.692
Rbf	10	0.001	0.694
Rbf	10	0.01	0.693
Rbf	10	0.1	0.693
Rbf	100	0.00001	0.631
Rbf	100	0.0001	0.692
Rbf	100	0.001	0.694
Rbf	100	0.01	0.693
Rbf	100	0.1	0.729
Polynomial	0.01	0.00001	0.500
Polynomial	0.01	0.0001	0.500
Polynomial	0.01	0.001	0.500
Polynomial	0.01	0.01	0.546
Polynomial	0.01	0.1	0.546
Polynomial	0.1	0.00001	0.500
Polynomial	0.1	0.0001	0.500
Polynomial	0.1	0.001	0.647
Polynomial	0.1	0.01	0.546
Polynomial	0.1	0.1	0.546

Figure 3.11: Table of Accuracy

Polynomial	1	0.00001	0.500
Polynomial	1	0.0001	0.500
Polynomial	1	0.001	0.546
Polynomial	1	0.01	0.546
Polynomial	1	0.1	0.546
Polynomial	10	0.00001	0.500
Polynomial	10	0.0001	0.500
Polynomial	10	0.001	0.546
Polynomial	10	0.01	0.546
Polynomial	10	0.1	0.546
Polynomial	100	0.00001	0.500
Polynomial	100	0.0001	0.647
Polynomial	100	0.001	0.546
Polynomial	100	0.01	0.546
Polynomial	100	0.1	0.546

Figure 3.12: Table of Accuracy

Sigmoid	0.01	0.00001	0.693
Sigmoid	0.01	0.0001	0. 693
Sigmoid	0.01	0.001	0. 693
Sigmoid	0.01	0.01	0. 693
Sigmoid	0.01	0.1	0. 693
Sigmoid	100	0.00001	0. 693
Sigmoid	100	0.0001	0. 693
Sigmoid	100	0.001	0. 693
Sigmoid	100	0.01	0. 693
Sigmoid	100	0.1	0.705

Figure 3.13: Table of Accuracy

3.4 Task4

3.4.1 Imbalanced train data

Since our dataset is an imbalanced dataset, accuracy cannot be used as a good performance metric. If we had a balanced dataset then accuracy would be fine for us. There is not enough data in the minority group. The Test Accuracy value is quite high, but that doesn't mean it's a good performance metric.

Below you can see the confusion matrix table of the imbalanced dataset.

⇒ Test Accuracy is **0.950**

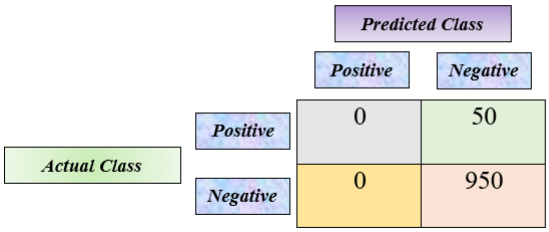

Figure 3.14: Confusion Matrix-1

3.4.2 Oversample the minority class

Below you can see the confusion matrix table of the oversampled the minority class.

⇒ Test Accuracy is **0.951**

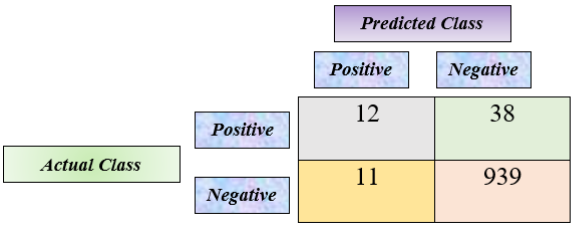

Figure 3.15: Confusion Matrix-2

3.4.3 Undersample the majority class

Below you can see the confusion matrix table of the undersampled the majority class.

⇒ Test Accuracy is **0.798**

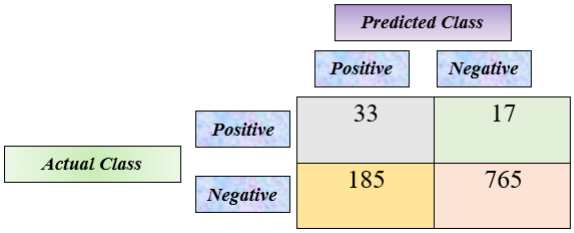

Figure 3.16: Confusion Matrix-3

3.4.4 With "class weight"

Below you can see the confusion matrix table of the balanced dataset with "class weight".

⇒ Test Accuracy is **0.936**

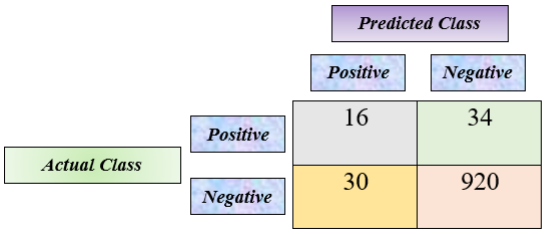

Figure 3.17: Confusion Matrix-4