University of Wollongong

School of Computing and Information Technology

CSCI464/964 - Computational Intelligence

Autumn 2020 - Assignment 2

Harsha Vardhan Seelam –6403190

[Hvs883@uowmail.edu.au](mailto:Hvs883@uowmail.edu.au)

Contents

[Two Spiral Problem 2](#_Toc40102869)

[Introduction: 2](#_Toc40102870)

[Choosing SVM: 2](#_Toc40102871)

[Parameter Tuning: 3](#_Toc40102872)

[Accuracy: 3](#_Toc40102873)

[Observations: 3](#_Toc40102874)

[Commands used: 4](#_Toc40102875)

[Abalone Age Problem 6](#_Toc40102876)

[Introduction: 6](#_Toc40102877)

[Choosing SVM: 7](#_Toc40102878)

[Parameter Tuning: 7](#_Toc40102879)

[Accuracy: 7](#_Toc40102880)

[Observations: 7](#_Toc40102881)

[Commands used: 8](#_Toc40102882)

[SPECT Heart Diagnosis Problem 10](#_Toc40102883)

[Introduction: 10](#_Toc40102884)

[Choosing SVM: 10](#_Toc40102885)

[Parameter Tuning: 11](#_Toc40102886)

[Accuracy: 11](#_Toc40102887)

[Observations: 11](#_Toc40102888)

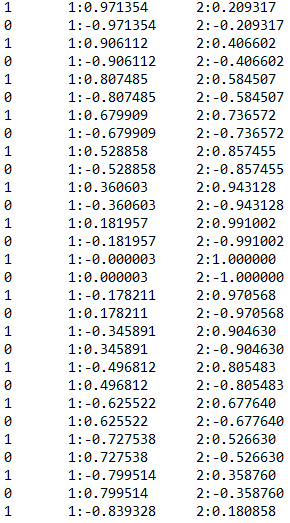
[Commands used: 11](#_Toc40102889)

[References: 14](#_Toc40102890)

# Two Spiral Problem

## Introduction:

The two-spiral problem is a two-class pattern classification. It was considered as one of the most difficult problems due to its complicated decision boundary. These two classes form two intertwined spirals on a two-dimensional plane. As the spirals form a coil around each other with not linearly separable dataset, it will be impossible to classify data. The given datafile consists of 192 datasets with 2 inputs and 1 output with either 0 or 1. According to the Assignment specification, I randomly picked 60% of data as training dataset, which is 116 datasets of 192. And I randomly picked 40% of data as testing dataset, which is 76 datasets of 192. In addition, the dataset is not in the acceptable form of SVM model for processing the data. So, the data file has been transformed into acceptable form of SVM model as shown below. The first column is the class of dataset and the second and third columns is the inputs.



## Choosing SVM:

The two spirals made a benchmark as one of most difficult problems, because of its characteristic of impossibility to linearly separate the data and classify them. In this dataset we need to classify the data into either 0 or 1. Firstly, in order to classify this dataset to either of the two classes 0 or 1, we can use multi class classifier to separate our dataset. Secondly, the dataset should be linearly separated and to achieve this, we can map the data into a higher dimensional space using the kernel trick. ­­­­­­­­­­­­In a nutshell, after parameter tuning with different kernel’s and classifiers, I would choose nu-SVC with a linear kernel function.

## Parameter Tuning:

Parameter tuning could be used to control the behaviour of the model, by adjusting the elements. As this problem needs to be classified, I choose to use only c-SVC and nu-SVC. However, I tried exploring different available kernels and tuned their parameters depending on the formulas i.e., what kind of equations they use to achieve these results.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | Kernel | Degree | Gamma | Nu | C | Accuracy% |
| C-SVC | RBF | Nan | Nan | Nan | 1 | 44.7368% |
| C-SVC | Sigmoid | Nan | Nan | Nan | 2 | 44.7368% |
| C-SVC | polynomial | Default | Default | Nan | 5 | 46.0526% |
| C-SVC | Linear | Nan | 2 | Nan | 10 | 44.7368% |
| Nu-SVC | Linear | Nan | Nan | 0.9 | Nan | 51.3158% |
| Nu-SVC | RBF | Nan | 5 | 0.1 | Nan | 43.4211% |
| Nu-SVC | polynomial | 3 | 2 | 0.5 | Nan | 53.9474% |
| Nu-SVC | Linear | Nan | Nan | 0.2 | Nan | 57.8947% |
| Nu-SVC | Sigmoid | Nan | Nan | 0.4 | Nan | 55.2632% |

## Accuracy:

I found that nu-SVC classifier with linear kernel approach gives me the highest possible accuracy of **57.8947%**. This accuracy could be reached by setting up the nu (i.e.., -n) parameter with 0.2.

## Observations:

Firstly, while working with classifiers, I’ve observed that, **parameter C** will tell the SVM optimization how much you want to avoid misclassifying each training dataset. But **nu parameter** is both a lower bound for the number of samples that are support vectors and an upper bound for the number of samples that are on the wrong side of the hyperplane.

Secondly, even though **parameter C** can take any positive value, it has no direct interpretation which makes it hard to find the suitable value for the model. However, **nu Parameter** has a direct interpretation as the value is bound between 0 and 1. Moreover, c and nu parameters are equal regarding their classification power. Its just that, C parameter is hard to interpret when compared to nu.

Also, I reached 57.8% accuracy using nu-SVM, but in assignment-1 I had an error rate of 53%.

## Commands used:

**\windows>svm-train -s 0 -t 2 -c 1 data1-AMTrain.txt data1.model**

optimization finished, #iter = 89

nu = 0.931034

obj = -107.496573, rho = -0.796102

nSV = 110, nBSV = 104

Total nSV = 110

\windows>svm-predict data1-AMTest.txt data1.model data1.output

**Accuracy** = 44.7368% (34/76) (classification)

**\windows>svm-train -s 0 -t 1 -c 5 -d 3 -g 1 data1-AMTrain.txt data1.model**

optimization finished, #iter = 165

nu = 0.927644

obj = -536.497731, rho = -0.535736

nSV = 109, nBSV = 105

Total nSV = 109

\windows>svm-predict data1-AMTest.txt data1.model data1.output

**Accuracy** = 46.0526% (35/76) (classification)

**\windows>svm-train -s 1 -t 2 -n 0.1 -g 5 data1-AMTrain.txt data1.model**

optimization finished, #iter = 1217

C = 16823.947200

obj = 199944.049320, rho = -11.978341

nSV = 81, nBSV = 0

Total nSV = 81

\windows>svm-predict data1-AMTest.txt data1.model data1.output

Accuracy = 43.4211% (33/76) (classification)

**\windows>svm-train -s 1 -t 1 -n 0.5 -g 2 -d 3 data1-AMTrain.txt data1.model**

optimization finished, #iter = 10

C = 102109.752251

obj = -6983.862349, rho = 0.199909

nSV = 65, nBSV = 53

Total nSV = 65

\windows>svm-predict data1-AMTest.txt data1.model data1.output

Accuracy = 53.9474% (41/76) (classification)

**\windows>svm-train -s 1 -t 0 -n 0.2 data1-AMTrain.txt data1.model**

optimization finished, #iter = 3

C = 43354.808766

obj = 209.217483, rho = 4.239806

nSV = 26, nBSV = 20

Total nSV = 26

\windows>svm-predict data1-AMTest.txt data1.model data1.output

Accuracy = 57.8947% (44/76) (classification)

**\windows>** **svm-train -s 1 -t 3 -n 0.4 data1-AMTrain.txt data1.model**

optimization finished, #iter = 258

C = -84.203148

obj = -6290.006929, rho = 0.088236

nSV = 50, nBSV = 50

Total nSV = 50

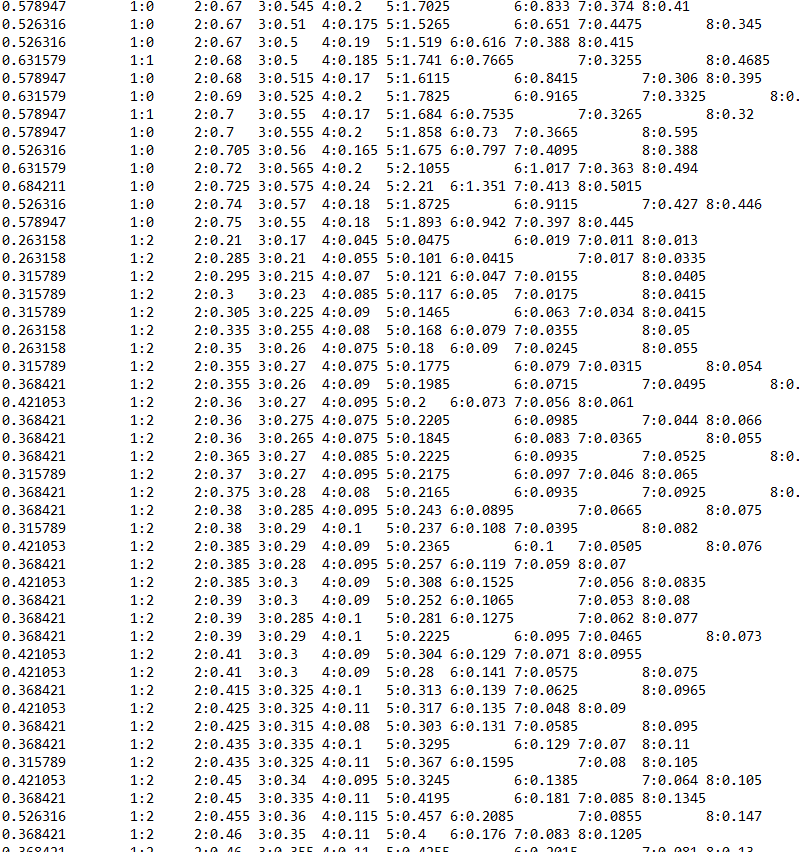
\windows>svm-predict data1-AMTest.txt data1.model data1.output

Accuracy = 55.2632% (42/76) (classification)

# Abalone Age Problem

## Introduction:

Abalone Age problem dataset is used for predicting the abalone’s age using the physical measurements. In late 1990’s, the abalone’s age is identified by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. The provided dataset consists of, 8 input data columns (i.e., Sex, length, Diameter, height, whole weight, shucked weight, viscera weight, shell weight) and one output data column (i.e., Rings). Yet, the dataset is not in an acceptable form for the SVM model. However, the dataset should transform in each below format where first column is class and from second to ninth column are the inputs for the regressor.



## Choosing SVM:

In Terms of regression analysis and prediction, the term error is included to show that the model does not fully represent the actual relation ship between independent and dependent variables. A standard error can tell you how wrong the regression model is on average using the units of the response variable.

However, in this regression analysis we aim to minimize the error rate. So, we use the mean square error as it works in a way where, the bigger the dataset, less the error rate.

Since, this model works with regression, I chose to use epsilon-SVR and nu-SVR regressor’s and as the data set is nonlinear, I would like to use kernels like RBF, polynomial and sigmoid.

## Parameter Tuning:

As Heart diagnosis problem needs regression, I choose to use only Epsilon-SVR and nu-SVR. However, I tried exploring different available kernels and tuned their parameters depending on the formulas i.e., what kind of equations they use to achieve these results.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Regressor | Kernel | Epsilon  ( - p ) | Epsilon  ( -e ) | Gamma  ( -g) | Coef0  ( -r) | Degree  (-d) | Cost  (-c) | Nu  (-n) | Mean square error | Squared correlation coefficient |
| Epsilon-SVR | RBF | 0.1 | 0.001 | 2 | Nan | Nan | 1 | Nan | 0.00961201 | 0.537253 |
| Epsilon-SVR | RBF | 0.2 | 0.2 | 2 | Nan | Nan | 1 | Nan | 0.0145578 | 0.504271 |
| Epsilon-SVR | RBF | 0.001 | 0.099 | 2 | Nan | Nan | 1 | Nan | 0.0100214 | 0.515878 |
| Nu-SVR | RBF | Nan | Nan | 1 | Nan | Nan | 1 | 0.1 | 0.0132714 | 0.509113 |
| Nu-SVR | RBF | Nan | Nan | 2 | Nan | Nan | 1 | 0.5 | 0.00941009 | 0.531032 |
| Nu-SVR | RBF | Nan | Nan | 2 | Nan | Nan | 0.5 | 0.9 | 0.00927586 | 0.533232 |
| Nu-SVR | Polynomial | Nan | Nan | 2.5 | 1 | 2 | 0.5 | 0.5 | 0.00940283 | 0.530129 |
| Nu-SVR | Sigmoid | Nan | Nan | 3 | 0.3 | Nan | 1 | 0.7 | 1843.77 | 0.181597 |

## Accuracy:

I found that Epsilon-SVR regressor with radial basis function kernel approach gives me the lowest means square error **0.00961201** and it has decent squared correlation coefficient of 0.537253.This could be reached by setting up the Epsilon (-p) to 0.1, Epsilon (-e) to 0.001, gamma to 2 and cost –(c) to 1.

## Observations:

As we know that this problem is a predicted the age of abalone, we won’t be able to use the SVM classification. To achieve this, we can use nu-SVR and Epsilon-SVR. After tuning different parameters with both regression methods, the lowest MSE has been achieved by setting the -p=0.1, -e=0.001, -g=2 and -c=1.

However, in assignment 1 I have achieved the lowest error percentage around 20.0% and in assignment 2, using regression we achieved the low means square error of 0.00961201

## Commands used:

**\windows>svm-train -s 3 -t 2 -p 0.1 -e 0.001 -g 2 -c 1 data2-AMTrain.txt data2.model**

optimization finished, #iter = 1915

nu = 0.265577

obj = -75.450416, rho = -0.620677

nSV = 1018, nBSV = 935

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.00961201 (regression)

Squared correlation coefficient = 0.537253 (regression)

**\windows>svm-train -s 3 -t 2 -p 0.2 -e 0.2 -g 2 -c 1 data2-AMTrain.txt data2.model**

optimization finished, #iter = 132

nu = 0.058402

obj = -17.726654, rho = -0.657581

nSV = 227, nBSV = 204

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.0145578 (regression)

Squared correlation coefficient = 0.504271 (regression)

**\windows>svm-train -s 3 -t 2 -p 0.001 -e 0.099 -g 2 -c 1 data2-AMTrain.txt data2.model**

optimization finished, #iter = 1896

nu = 0.893792

obj = -274.201559, rho = -0.726537

nSV = 3342, nBSV = 3236

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.0100214 (regression)

Squared correlation coefficient = 0.515878 (regression)

**\windows>svm-train -s 4 -t 2 -g 1 -c 1 -n 0.1 data2-AMTrain.txt data2.model**

optimization finished, #iter = 3604

epsilon = 0.179623

obj = -92.266667, rho = -0.602034

nSV = 400, nBSV = 341

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.0132714 (regression)

Squared correlation coefficient = 0.509113 (regression)

**\windows>svm-train -s 4 -t 2 -g 2 -c 1 -n 0.5 data2-AMTrain.txt data2.model**

optimization finished, #iter = 21381

epsilon = 0.052569

obj = -236.203007, rho = -0.664623

nSV = 1880, nBSV = 1791

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.00941009 (regression)

Squared correlation coefficient = 0.531032 (regression)

**\windows>svm-train -s 4 -t 2 -g 2 -c 0.5 -n 0.9 data2-AMTrain.txt data2.model**

optimization finished, #iter = 12965

epsilon = 0.008367

obj = -141.325979, rho = -0.658285

nSV = 3351, nBSV = 3265

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 0.00927586 (regression)

Squared correlation coefficient = 0.533232 (regression)

**\windows>svm-train -s 4 -t 3 -g 3 -r 0.3 -c 1 -n 0.7 data2-AMTrain.txt data2.model**

optimization finished, #iter = 2214

epsilon = 11.962235

obj = -50266.197166, rho = 27.219002

nSV = 2574, nBSV = 2572

\windows>svm-predict data2-AMTest.txt data2.model data2.output

Mean squared error = 1843.77 (regression)

Squared correlation coefficient = 0.181597 (regression)

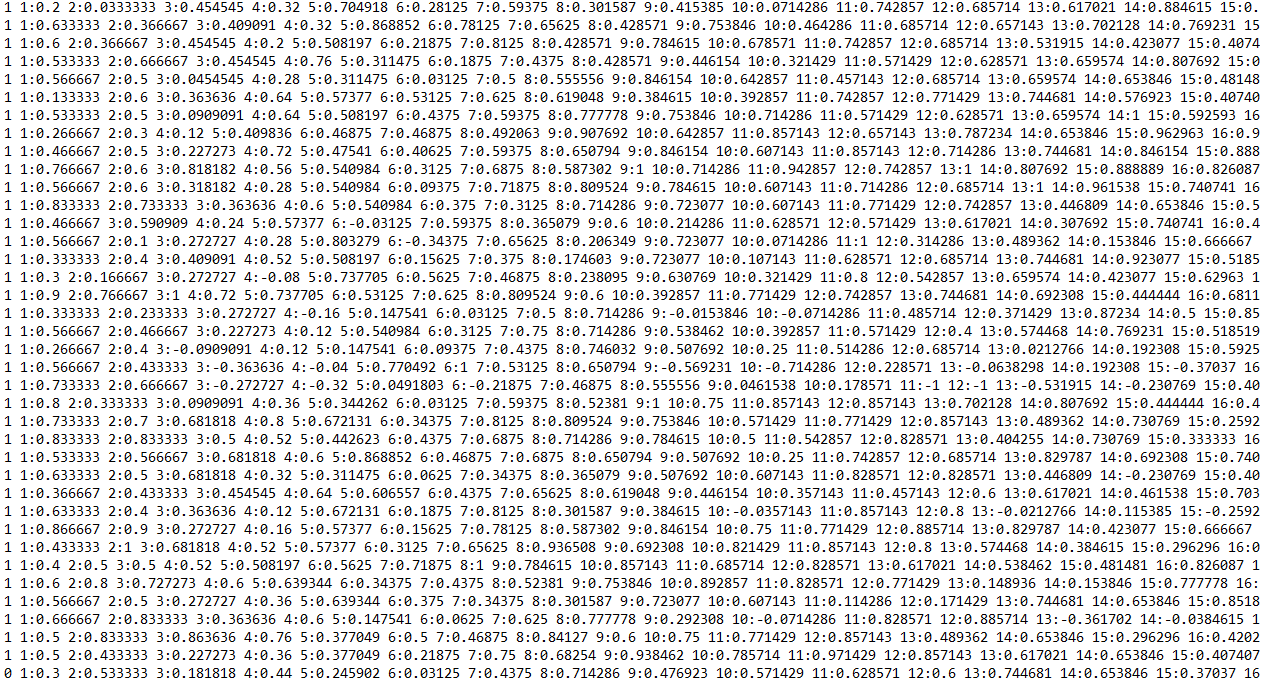
# SPECT Heart Diagnosis Problem

## Introduction:

The dataset contains the information of patients who are either abnormal or normal. In addition, the dataset is acquired from diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. The dataset of 299 SPECT image sets were processed to extract features that summarize the original SPECT images. As a result, 44 continuous feature patterns were extracted to represent each patient.

However, to determine whether the patient is abnormal or normal our classifier need to classify them to either 0 or 1. This must be obtained by using the 44 continuous features provided in the dataset.

Moreover, the given dataset consists of 45 columns, which are not in the acceptable form of SVM model processing. So, I transformed the order of columns which supports the SVM as, first column is the output class and next 44 columns are the input classes to classify the data.



## Choosing SVM:

The obtained data set needs to be classified in either class 1 or class 0 and, to do so, we need to use c-SVC and nu-SVC classification methods to achieve the desired output. In addition, as the data set is nonlinear, I would like to use kernels like RBF, polynomial and sigmoid.

Moreover, the input values of the dataset are more than 0 which is not the acceptable by the SVM models. So, we need to scale down the data to values ranging from -1 to 1. In order to achieve this, we need to use the program svm-scale.c and apply it both training and testing data inputs and then send the scaled files for training and classification. The above provided picture of the dataset is already scaled.

# Parameter Tuning:

. As this problem needs to be classified, I choose to use only c-SVM and nu-SVM. However, I tried exploring different available kernels and tuned their parameters depending on the formulas i.e., what kind of equations they use to achieve these results.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Classifier | Kernel | Cost(-c) | Nu(-n) | Gamma | Accuracy |
| c-SVC | RBF | 1 | nan | 1 | 40% |
| c-SVC | RBF | 1.5 | nan | nan | 58% |
| c-SVC | Linear | 1.9 | nan | nan | 58% |
| c-SVC | Linear | 1.8 | nan | nan | 60% |
| Nu-SVC | RBF | nan | 0.1 | 0.001 | 70% |
| Nu-SVC | RBF | nan | 0.2 | 2 | 40% |
| Nu-SVC | Linear | nan | 0.4 | Nan | 48% |
| Nu-SVC | Linear | nan | 0.3 | nan | 44% |

## Accuracy:

I found that nu-SVC classifier with Radial basis kernel approach gives me the highest possible accuracy of **70%**. This accuracy could be reached by setting up the nu (i.e.., -n) parameter with 0.1 and gamma (-g) with 0.001.

## Observations:

This given data set to heart diagnosis problem need to be classified based on their health condition. So, I choose to try tunning parameter of c-SVC and nu-SVC with Liner and RBF kernels. However, while I was tunning parameter of nu-svc, I found that nu parameter bound of 0.5 to 0.9 is not feasible to this dataset. So, for tunning the parameter of nu I limited the bound to 0.1 to 0.5 instead of 0.1 to 0.9. For, this dataset the accuracy is as low as 40% to as high as 70%. However, the accuracy level of nu-svc was not much affected the gamma parameter is tuned to different parameter in some cases.

Moreover, my best possible accuracy for this dataset was 70% in assignment 2. But, in assignment-1 using MLP I have achieved the accuracy of 44% only. So, I think that, using multi-class classifier is good for this problem.

## Commands used:

**\windows>svm-scale -l -1 -u 1 data3-AMTrain.txt > data3-AMTrain.scale**

**\windows>svm-scale -l -1 -u 1 data3-AMTest.txt > data3-AMTest.scale**

**\windows>svm-train -s 0 -t 2 -c 1 -g 1 data3-AMTrain.scale data3.model**

optimization finished, #iter = 325

nu = 0.394987

obj = -77.856661, rho = -0.912130

nSV = 242, nBSV = 75

Total nSV = 242

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 40% (20/50) (classification)

**\windows>svm-train -s 0 -t 2 -c 1.5 data3-AMTrain.scale data3.model**

optimization finished, #iter = 680

nu = 0.364401

obj = -145.740219, rho = -7.016452

nSV = 127, nBSV = 94

Total nSV = 127

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 58% (29/50) (classification)

**\windows>svm-train -s 0 -t 0 -c 1.9 data3-AMTrain.scale data3.model**

optimization finished, #iter = 799

nu = 0.352787

obj = -179.486457, rho = -7.205474

nSV = 126, nBSV = 89

Total nSV = 126

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 58% (29/50) (classification)

**\windows>svm-train -s 0 -t 0 -c 1.8 data3-AMTrain.scale data3.model**

optimization finished, #iter = 642

nu = 0.355424

obj = -171.125229, rho = -7.136728

nSV = 128, nBSV = 92

Total nSV = 128

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 60% (30/50) (classification)

**\windows>svm-train -s 1 -t 2 -n 0.1 -g 0.001 data3-AMTrain.scale data3.model**

optimization finished, #iter = 20

C = 12695.966268

obj = 202733.870819, rho = -41.643612

nSV = 41, nBSV = 19

Total nSV = 41

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 70% (35/50) (classification)

**\windows>svm-train -s 1 -t 2 -n 0.2 -g 2 data3-AMTrain.scale data3.model**

optimization finished, #iter = 436

C = 2.712136

obj = 81.092497, rho = -0.738142

nSV = 276, nBSV = 0

Total nSV = 276

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 40% (20/50) (classification)

**\windows>svm-train -s 1 -t 0 -n 0.4 data3-AMTrain.scale data3.model**

optimization finished, #iter = 285

C = 0.742444

obj = 9.935053, rho = -6.845753

nSV = 131, nBSV = 108

Total nSV = 131

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 48% (24/50) (classification)

**\windows>svm-train -s 1 -t 0 -n 0.3 data3-AMTrain.scale data3.model**

optimization finished, #iter = 1107

C = 8.236123

obj = 75.227163, rho = -10.730752

nSV = 107, nBSV = 73

Total nSV = 107

\windows>svm-predict data3-AMTest.scale data3.model data3.output

Accuracy = 44% (22/50) (classification)

# References:

Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: Abalone Data Set*. [online] Available at: <http://archive.ics.uci.edu/ml/datasets/Abalone> [Accessed 9 May 2020].

Archive.ics.uci.edu. 2020. *UCI Machine Learning Repository: SPECT Heart Data Set*. [online] Available at: <http://archive.ics.uci.edu/ml/datasets/SPECT+Heart> [Accessed 9 May 2020].

Behery, G., El-Harby, A. and El-Bakry, M., 2009. Reorganizing Neural Network System for Two Spirals and Linear Low-Density Polyethylene Copolymer Problems. *Applied Computational Intelligence and Soft Computing*, 2009, pp.1-11.

Medium. 2020. *Regression — Why Mean Square Error?*. [online] Available at: <https://towardsdatascience.com/https-medium-com-chayankathuria-regression-why-mean-square-error-a8cad2a1c96f> [Accessed 9 May 2020].