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.

```
1       1:0.971354      2:0.209317
0       1:-0.971354     2:-0.209317
1       1:0.906112      2:0.406602
0       1:-0.906112     2:-0.406602
1       1:0.807485      2:0.584507
0       1:-0.807485     2:-0.584507
1       1:0.679909      2:0.736572
0       1:-0.679909     2:-0.736572
1       1:0.528858      2:0.857455
0       1:-0.528858     2:-0.857455
1       1:0.360603      2:0.943128
0       1:-0.360603     2:-0.943128
1       1:0.181957      2:0.991002
0       1:-0.181957     2:-0.991002
1       1:-0.000003     2:1.000000
0       1:0.000003      2:-1.000000
1       1:-0.178211     2:0.970568
0       1:0.178211      2:-0.970568
1       1:-0.345891     2:0.904630
0       1:0.345891      2:-0.904630
1       1:-0.496812     2:0.805483
0       1:0.496812      2:-0.805483
1       1:-0.625522     2:0.677640
0       1:0.625522      2:-0.677640
1       1:-0.727538     2:0.526630
0       1:0.727538      2:-0.526630
1       1:-0.799514     2:0.358760
0       1:0.799514      2:-0.358760
1       1:-0.839328     2:0.180858
```

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:

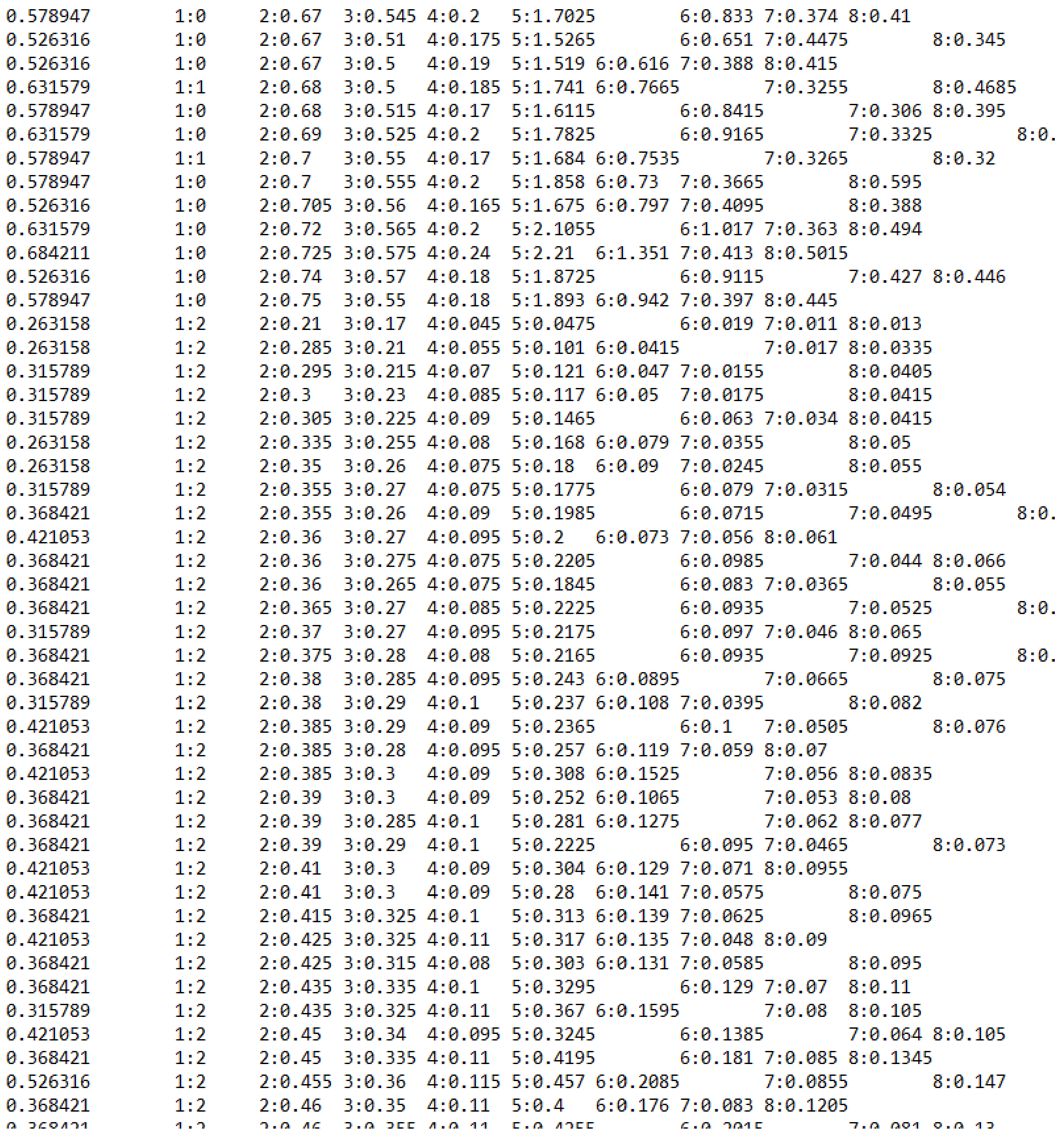
```
\windows>svm-train -s 0 -t 2 -c 1 data1-AMTrain.txt data1.model
optimization finished, #iter = 89
nu = 0.931034
obj = -107.496573, rho = -0.796102
nSV = 110, nBSV = 104
Total nSV = 110
\windows>svm-predict data1-AMTest.txt data1.model data1.output
Accuracy = 44.7368% (34/76) (classification)
\windows>svm-train -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.



```
0.578947        1:0     2:0.67  3:0.545 4:0.2   5:1.7025        6:0.833 7:0.374 8:0.41
0.526316        1:0     2:0.67  3:0.51  4:0.175 5:1.5265        6:0.651 7:0.4475        8:0.345
0.526316        1:0     2:0.67  3:0.5   4:0.19  5:1.519 6:0.616 7:0.388 8:0.415
0.631579        1:1     2:0.68  3:0.5   4:0.185 5:1.741 6:0.7665        7:0.3255        8:0.4685
0.578947        1:0     2:0.68  3:0.515 4:0.17  5:1.6115        6:0.8415        7:0.306 8:0.395
0.631579        1:0     2:0.69  3:0.525 4:0.2   5:1.7825        6:0.9165        7:0.3325        8:0.
0.578947        1:1     2:0.7   3:0.55  4:0.17  5:1.684 6:0.7535        7:0.3265        8:0.32
0.578947        1:0     2:0.7   3:0.555 4:0.2   5:1.858 6:0.73  7:0.3665        8:0.595
0.526316        1:0     2:0.705 3:0.56  4:0.165 5:1.675 6:0.797 7:0.4095        8:0.388
0.631579        1:0     2:0.72  3:0.565 4:0.2   5:2.1055        6:1.017 7:0.363 8:0.494
0.684211        1:0     2:0.725 3:0.575 4:0.24  5:2.21  6:1.351 7:0.413 8:0.5015
0.526316        1:0     2:0.74  3:0.57  4:0.18  5:1.8725        6:0.9115        7:0.427 8:0.446
0.578947        1:0     2:0.75  3:0.55  4:0.18  5:1.893 6:0.942 7:0.397 8:0.445
0.263158        1:2     2:0.21  3:0.17  4:0.045 5:0.0475        6:0.019 7:0.011 8:0.013
0.263158        1:2     2:0.285 3:0.21  4:0.055 5:0.101 6:0.0415        7:0.017 8:0.0335
0.315789        1:2     2:0.295 3:0.215 4:0.07  5:0.121 6:0.047 7:0.0155        8:0.0405
0.315789        1:2     2:0.3   3:0.23  4:0.085 5:0.117 6:0.05  7:0.0175        8:0.0415
0.315789        1:2     2:0.305 3:0.225 4:0.09  5:0.1465        6:0.063 7:0.034 8:0.0415
0.263158        1:2     2:0.335 3:0.255 4:0.08  5:0.168 6:0.079 7:0.0355        8:0.05
0.263158        1:2     2:0.35  3:0.26  4:0.075 5:0.18  6:0.09  7:0.0245        8:0.055
0.315789        1:2     2:0.355 3:0.27  4:0.075 5:0.1775        6:0.079 7:0.0315        8:0.054
0.368421        1:2     2:0.355 3:0.26  4:0.09  5:0.1985        6:0.0715        7:0.0495        8:0.
0.421053        1:2     2:0.36  3:0.27  4:0.095 5:0.2   6:0.073 7:0.056 8:0.061
0.368421        1:2     2:0.36  3:0.275 4:0.075 5:0.2205        6:0.0985        7:0.044 8:0.066
0.368421        1:2     2:0.36  3:0.265 4:0.075 5:0.1845        6:0.083 7:0.0365        8:0.055
0.368421        1:2     2:0.365 3:0.27  4:0.085 5:0.2225        6:0.0935        7:0.0525        8:0.
0.315789        1:2     2:0.37  3:0.27  4:0.095 5:0.2175        6:0.097 7:0.046 8:0.065
0.368421        1:2     2:0.375 3:0.28  4:0.08  5:0.2165        6:0.0935        7:0.0925        8:0.
0.368421        1:2     2:0.38  3:0.285 4:0.095 5:0.243 6:0.0895        7:0.0665        8:0.075
0.315789        1:2     2:0.38  3:0.29  4:0.1   5:0.237 6:0.108 7:0.0395        8:0.082
0.421053        1:2     2:0.385 3:0.29  4:0.09  5:0.2365        6:0.1   7:0.0505        8:0.076
0.368421        1:2     2:0.385 3:0.28  4:0.095 5:0.257 6:0.119 7:0.059 8:0.07
0.421053        1:2     2:0.385 3:0.3   4:0.09  5:0.308 6:0.1525        7:0.056 8:0.0835
0.368421        1:2     2:0.39  3:0.3   4:0.09  5:0.252 6:0.1065        7:0.053 8:0.08
0.368421        1:2     2:0.39  3:0.285 4:0.1   5:0.281 6:0.1275        7:0.062 8:0.077
0.368421        1:2     2:0.39  3:0.29  4:0.1   5:0.2225        6:0.095 7:0.0465        8:0.073
0.421053        1:2     2:0.41  3:0.3   4:0.09  5:0.304 6:0.129 7:0.071 8:0.0955
0.421053        1:2     2:0.41  3:0.3   4:0.09  5:0.28  6:0.141 7:0.0575        8:0.075
0.368421        1:2     2:0.415 3:0.325 4:0.1   5:0.313 6:0.139 7:0.0625        8:0.0965
0.421053        1:2     2:0.425 3:0.325 4:0.11  5:0.317 6:0.135 7:0.048 8:0.09
0.368421        1:2     2:0.425 3:0.315 4:0.08  5:0.303 6:0.131 7:0.0585        8:0.095
0.368421        1:2     2:0.435 3:0.335 4:0.1   5:0.3295        6:0.129 7:0.07  8:0.11
0.315789        1:2     2:0.435 3:0.325 4:0.11  5:0.367 6:0.1595        7:0.08  8:0.105
0.421053        1:2     2:0.45  3:0.34  4:0.095 5:0.3245        6:0.1385        7:0.064 8:0.105
0.368421        1:2     2:0.45  3:0.335 4:0.11  5:0.4195        6:0.181 7:0.085 8:0.1345
0.526316        1:2     2:0.455 3:0.36  4:0.115 5:0.457 6:0.2085        7:0.0855        8:0.147
0.368421        1:2     2:0.46  3:0.35  4:0.11  5:0.4   6:0.176 7:0.083 8:0.1205
0.368421        1:2     2:0.46  3:0.355 4:0.11  5:0.4255        6:0.2015        7:0.081 8:0.13
```

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)
```

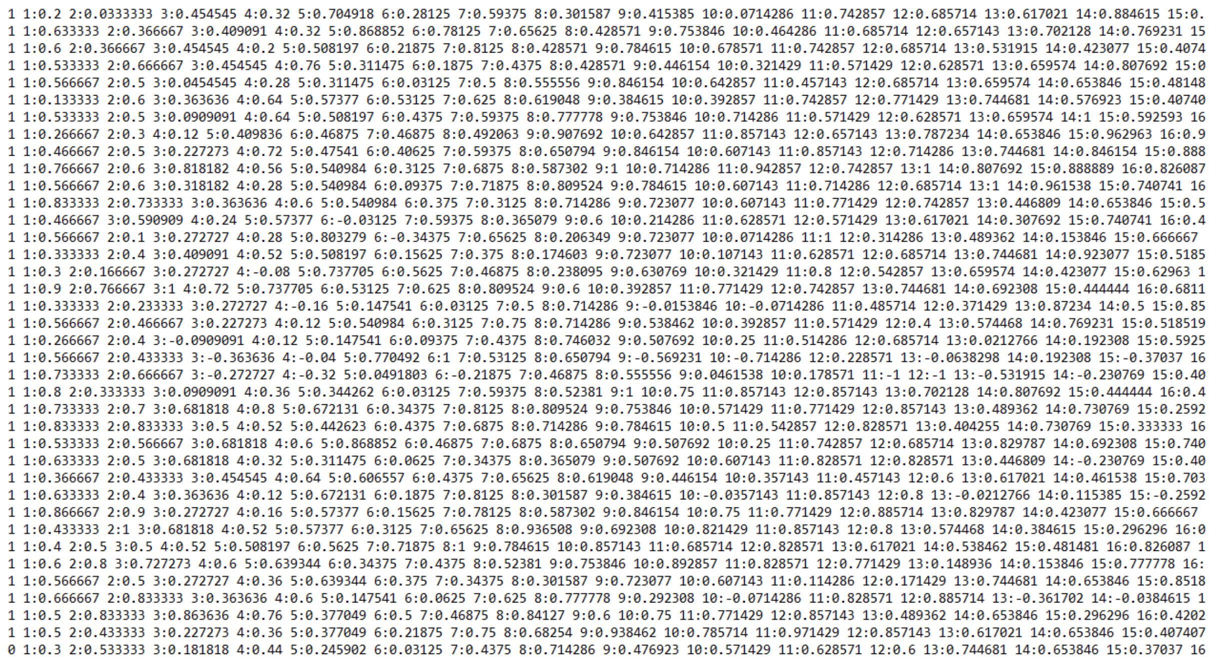
SPECT Heart Diagnosis Problem

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.



```
1 1:0.2 2:0.0333333 3:0.454545 4:0.32 5:0.704918 6:0.28125 7:0.59375 8:0.301587 9:0.415385 10:0.0714286 11:0.742857 12:0.685714 13:0.617021 14:0.884615 15:0.
1 1:0.633333 2:0.366667 3:0.409091 4:0.32 5:0.868852 6:0.78125 7:0.65625 8:0.428571 9:0.753846 10:0.464286 11:0.685714 12:0.657143 13:0.702128 14:0.769231 15
1 1:0.6 2:0.366667 3:0.454545 4:0.2 5:0.508197 6:0.21875 7:0.8125 8:0.428571 9:0.784615 10:0.678571 11:0.742857 12:0.685714 13:0.531915 14:0.423077 15:0.4074
1 1:0.533333 2:0.666667 3:0.454545 4:0.76 5:0.311475 6:0.1875 7:0.4375 8:0.428571 9:0.446154 10:0.321429 11:0.571429 12:0.628571 13:0.659574 14:0.807692 15:0
1 1:0.566667 2:0.5 3:0.0454545 4:0.28 5:0.311475 6:0.03125 7:0.5 8:0.555556 9:0.846154 10:0.642857 11:0.457143 12:0.685714 13:0.659574 14:0.653846 15:0.48148
1 1:0.133333 2:0.6 3:0.363636 4:0.64 5:0.57377 6:0.53125 7:0.625 8:0.619048 9:0.384615 10:0.392857 11:0.742857 12:0.771429 13:0.744681 14:0.576923 15:0.40740
1 1:0.533333 2:0.5 3:0.0909091 4:0.64 5:0.508197 6:0.4375 7:0.59375 8:0.777778 9:0.753846 10:0.714286 11:0.571429 12:0.628571 13:0.659574 14:1 15:0.592593 16
1 1:0.266667 2:0.3 4:0.12 5:0.409836 6:0.46875 7:0.46875 8:0.492063 9:0.907692 10:0.642857 11:0.857143 12:0.657143 13:0.787234 14:0.653846 15:0.962963 16:0.9
1 1:0.466667 2:0.5 3:0.227273 4:0.72 5:0.47541 6:0.40625 7:0.59375 8:0.650794 9:0.846154 10:0.607143 11:0.857143 12:0.714286 13:0.744681 14:0.846154 15:0.888
1 1:0.766667 2:0.6 3:0.818182 4:0.56 5:0.540984 6:0.3125 7:0.6875 8:0.587302 9:1 10:0.714286 11:0.942857 12:0.742857 13:1 14:0.807692 15:0.888889 16:0.826087
1 1:0.566667 2:0.6 3:0.318182 4:0.28 5:0.540984 6:0.09375 7:0.71875 8:0.809524 9:0.784615 10:0.607143 11:0.714286 12:0.685714 13:1 14:0.961538 15:0.740741 16
1 1:0.833333 2:0.733333 3:0.363636 4:0.6 5:0.540984 6:0.375 7:0.3125 8:0.714286 9:0.723077 10:0.607143 11:0.771429 12:0.742857 13:0.446809 14:0.653846 15:0.5
1 1:0.466667 3:0.590909 4:0.24 5:0.57377 6:-0.03125 7:0.59375 8:0.365079 9:0.6 10:0.214286 11:0.628571 12:0.571429 13:0.617021 14:0.307692 15:0.740741 16:0.4
1 1:0.566667 2:0.1 3:0.272727 4:0.28 5:0.803279 6:-0.34375 7:0.65625 8:0.206349 9:0.723077 10:0.0714286 11:1 12:0.314286 13:0.489362 14:0.153846 15:0.666667
1 1:0.333333 2:0.4 3:0.409091 4:0.52 5:0.508197 6:0.15625 7:0.375 8:0.174603 9:0.723077 10:0.107143 11:0.628571 12:0.685714 13:0.744681 14:0.923077 15:0.5185
1 1:0.3 2:0.166667 3:0.272727 4:-0.08 5:0.737705 6:0.5625 7:0.46875 8:0.238095 9:0.630769 10:0.321429 11:0.8 12:0.542857 13:0.659574 14:0.423077 15:0.62963 1
1 1:0.9 2:0.766667 3:1 4:0.72 5:0.737705 6:0.53125 7:0.625 8:0.809524 9:0.6 10:0.392857 11:0.771429 12:0.742857 13:0.744681 14:0.692308 15:0.444444 16:0.6811
1 1:0.333333 2:0.233333 3:0.272727 4:-0.16 5:0.147541 6:0.03125 7:0.5 8:0.714286 9:-0.0153846 10:-0.0714286 11:0.485714 12:0.371429 13:0.87234 14:0.5 15:0.85
1 1:0.566667 2:0.466667 3:0.227273 4:0.12 5:0.540984 6:0.3125 7:0.75 8:0.714286 9:0.538462 10:0.392857 11:0.571429 12:0.4 13:0.574468 14:0.769231 15:0.518519
1 1:0.266667 2:0.4 3:-0.0909091 4:0.12 5:0.147541 6:0.09375 7:0.4375 8:0.746032 9:0.507692 10:0.25 11:0.514286 12:0.685714 13:0.0212766 14:0.192308 15:0.5925
1 1:0.566667 2:0.433333 3:-0.363636 4:-0.04 5:0.770492 6:1 7:0.53125 8:0.650794 9:-0.569231 10:-0.714286 12:0.228571 13:-0.0638298 14:0.192308 15:-0.37037 16
1 1:0.733333 2:0.666667 3:-0.272727 4:-0.32 5:0.0491803 6:-0.21875 7:0.46875 8:0.555556 9:0.0461538 10:0.178571 11:-1 12:-1 13:-0.531915 14:-0.230769 15:0.40
1 1:0.8 2:0.333333 3:0.0909091 4:0.36 5:0.344262 6:0.03125 7:0.59375 8:0.52381 9:1 10:0.75 11:0.857143 12:0.857143 13:0.702128 14:0.807692 15:0.444444 16:0.4
1 1:0.733333 2:0.7 3:0.681818 4:0.8 5:0.672131 6:0.34375 7:0.8125 8:0.809524 9:0.753846 10:0.571429 11:0.771429 12:0.857143 13:0.489362 14:0.730769 15:0.2592
1 1:0.833333 2:0.833333 3:0.5 4:0.52 5:0.442623 6:0.4375 7:0.6875 8:0.714286 9:0.784615 10:0.5 11:0.542857 12:0.828571 13:0.404255 14:0.730769 15:0.333333 16
1 1:0.533333 2:0.566667 3:0.681818 4:0.6 5:0.868852 6:0.46875 7:0.6875 8:0.650794 9:0.507692 10:0.25 11:0.742857 12:0.685714 13:0.829787 14:0.692308 15:0.740
1 1:0.633333 2:0.5 3:0.681818 4:0.32 5:0.311475 6:0.0625 7:0.34375 8:0.365079 9:0.507692 10:0.607143 11:0.828571 12:0.828571 13:0.446809 14:-0.230769 15:0.40
1 1:0.366667 2:0.433333 3:0.454545 4:0.64 5:0.606557 6:0.4375 7:0.65625 8:0.619048 9:0.446154 10:0.357143 11:0.457143 12:0.6 13:0.617021 14:0.461538 15:0.703
1 1:0.633333 2:0.4 3:0.363636 4:0.12 5:0.672131 6:0.1875 7:0.8125 8:0.301587 9:0.384615 10:-0.0357143 11:0.857143 12:0.8 13:-0.0212766 14:0.115385 15:-0.2592
1 1:0.866667 2:0.9 3:0.272727 4:0.16 5:0.57377 6:0.15625 7:0.78125 8:0.587302 9:0.846154 10:0.75 11:0.771429 12:0.885714 13:0.829787 14:0.423077 15:0.666667
1 1:0.433333 2:1 3:0.681818 4:0.52 5:0.57377 6:0.3125 7:0.65625 8:0.936508 9:0.692308 10:0.821429 11:0.857143 12:0.8 13:0.574468 14:0.384615 15:0.296296 16:0
1 1:0.4 2:0.5 3:0.5 4:0.52 5:0.508197 6:0.5625 7:0.71875 8:1 9:0.784615 10:0.857143 11:0.685714 12:0.828571 13:0.617021 14:0.538462 15:0.481481 16:0.826087 1
1 1:0.6 2:0.8 3:0.727273 4:0.6 5:0.639344 6:0.34375 7:0.4375 8:0.52381 9:0.753846 10:0.892857 11:0.828571 12:0.771429 13:0.148936 14:0.153846 15:0.777778 16:
1 1:0.566667 2:0.5 3:0.272727 4:0.36 5:0.639344 6:0.375 7:0.34375 8:0.301587 9:0.723077 10:0.607143 11:0.114286 12:0.171429 13:0.744681 14:0.653846 15:0.8518
1 1:0.666667 2:0.833333 3:0.363636 4:0.6 5:0.147541 6:0.0625 7:0.625 8:0.777778 9:0.292308 10:-0.0714286 11:0.828571 12:0.885714 13:-0.361702 14:-0.0384615 1
1 1:0.5 2:0.833333 3:0.863636 4:0.76 5:0.377049 6:0.5 7:0.46875 8:0.84127 9:0.6 10:0.75 11:0.771429 12:0.857143 13:0.489362 14:0.653846 15:0.296296 16:0.4202
1 1:0.5 2:0.433333 3:0.227273 4:0.36 5:0.377049 6:0.21875 7:0.75 8:0.68254 9:0.938462 10:0.785714 11:0.971429 12:0.857143 13:0.617021 14:0.653846 15:0.407407
0 1:0.3 2:0.533333 3:0.181818 4:0.44 5:0.245902 6:0.03125 7:0.4375 8:0.714286 9:0.476923 10:0.571429 11:0.628571 12:0.6 13:0.744681 14:0.653846 15:0.37037 16
```

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].