Classification
Artificial Intelligence
1 A

Task 1 - Data

The data that was given is not in an appropriate format for LibSVM. The first task is to convert the data sets into the format specified in the LibSVM 'README' file located in its root directory.

The source code for this task is located in ***appendix A***.

The result was tested with the checkdata.py using the command line "*python checkdata.py fileName*" and the results were "*no error*".

Task 2- Normalisation

The data that was converted is raw data, it has not been normalised (scaled). Normalising the data is very important when using SVMs as it stops attributes with a large numeric range from dominating the classification model.

This task required the use of svm-scale.exe in the command prompt, with the need of the un-normalised data and the new file as parameters, as shown in ***appendix B***.

Task 3 - Grid Search

When using the C-SVC SVM with the Gaussian radial basis kernel there are two tuneable parameters, C (cost) and γ (gamma). To achieve the highest classification rate possible it is very important to search for an optimal pair of these values. LibSVM makes this process very simple by including a Python script which carries out a grid search, a systematic search for optimal SVM parameters.

This task is to carry out a grid search on the training data set.

At first a general idea for this task was made by using the default values for C and γ using the command prompt - *python grid.py normalisedFile*, results seen in ***appendix C***.

Using the values of C and γ, and working out the log2 of each a finer search could be made using a longer command prompt - *python grid.py -log2c beg, end, step - log2g beg, end, step normalisedFile*, as seen in ***appendix D***.

The values of the first grid search for C was 2 to power 9, so for the finer search a +1/-1 of the power value was used as the beginning and end of the search.

Task 4 - Classification

This task involves using LibSVM's 'svm-train' and 'svm-predict', both command line applications. With the normalised training set as the input file, 'svm-train' can be used with the suitable parameter values discovered for c and γ during task 3(seen at the bottom of ***appendix E***). When the classification model is built the use of 'svm-predict' will be needed to classify the normalised testing set.

The command line of *svm-train.exe - g 0.0473661427034 - c 362.038671968 train.dat* was used first which outputted a .model file to be used by 'svm-predict'.

'svm-predict' used this file in conjunction with the normalised training set to create a predicted file, using the command line *svm-predict.exe test.dat train.dat.model train.dat*

As seen in ***appendix F***.

SVM	Number of misclassified instances	Accuracy rate %
Non-linear	47	60.8333

Task 5 - Classification Analysis

By using the output file of 'svm-predict' (generated during task 4), it is possible to map the classifications made back to the original instances. The order of the predictions in the output file is the same as the order of the instances in the testing data set.

SVM	Number of misclassified instances	Accuracy rate %
Non-linear	47	80.83333333

Label	Actual label	Predicted label
Defective	60	29
Non-defective	60	91

Could for this can be found in ***appendix G*** as well as the command prompt view in ***appendix H***.

As it can be seen by the output there was a total of 47 misclassified instances which was the same as task 4 but the only difference is that the accuracy between task 4 and 5 has a 20% difference.

This is because the features used for this test did not separate the classes were well.

Task 6 - Linear Classification

When using the C-SVC SVM with the linear kernel there is only one tunable parameter, C (cost). In this experiment, it is required to use LibSVM's 'svm-train' to train models using C = 10, C = 100, and C = 1000, in turn.

SVM	C	γ	Accuracy rate %
Linear	10	-	60.8333
Linear	100	-	60.8333
Linear	1000	-	61.6667

A look at the command prompt can be found in ***appendix I***.

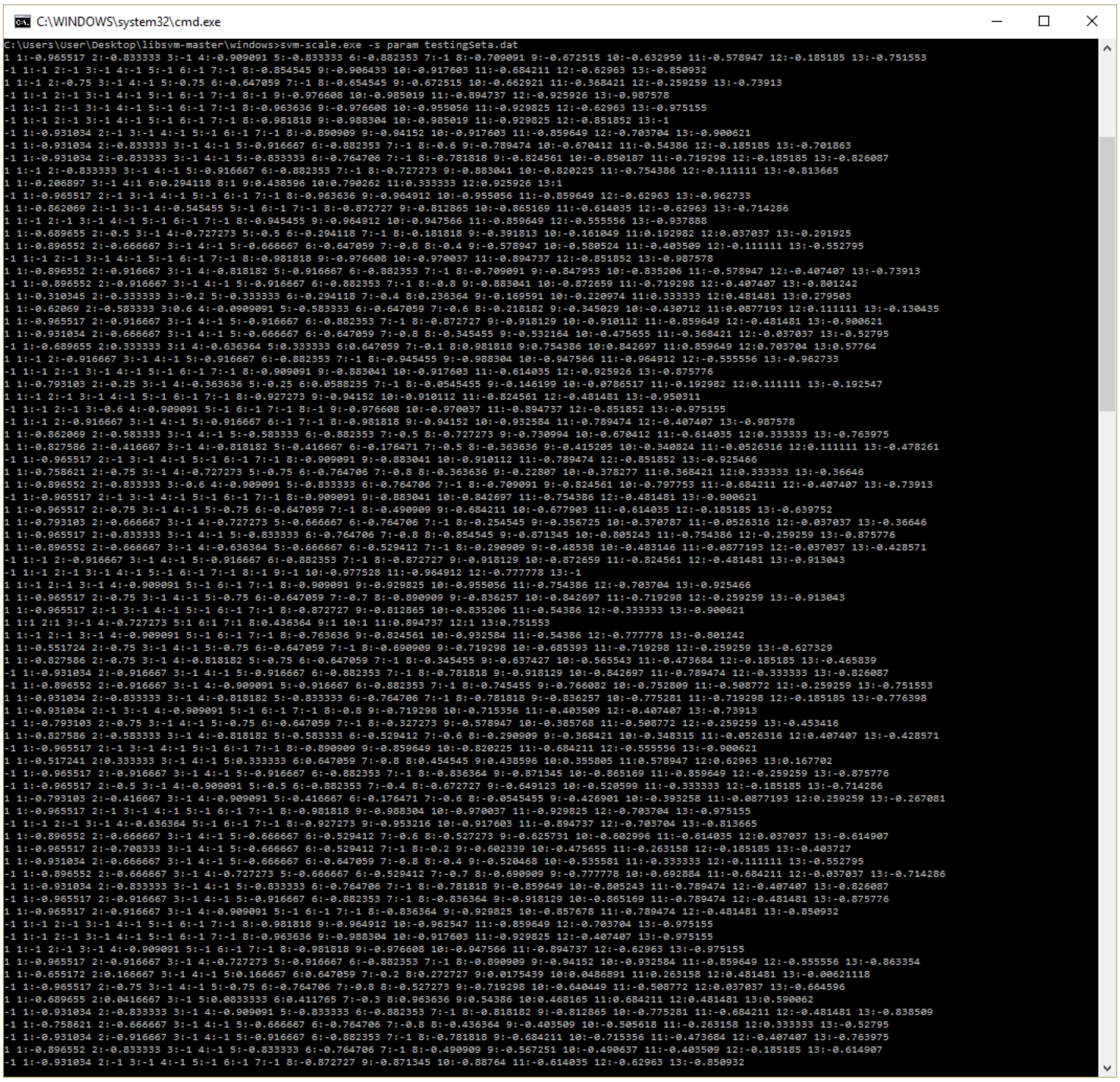
As shown by the table C = 1000 gives the best accuracy which is total different from task 4, as task 4 has the same accuracy as C = 10 and C = 100. Without a look at the full accuracy rate(full decimal) it is hard to determine if the accuracy is increasing with cost with just these three C values.

Appendices

Appendix A:

```
import csv
import sys

inputFile = sys.argv[1] #input file
outputFile = sys.argv[2] # output file

with open(inputFile, 'rb') as csvFile:
    writeToFile = [] #list
    myReader = csv.reader(csvFile, delimiter=',')#read from csv file
    for row in myReader:
        if row[len(row) - 1]== '1': # check character at end of line
            newLine = "+" + row[len(row)- 1] # add new line with '+1' at front
        else:
            newLine = "" + row[len(row) - 1]# add new line with '-1' at front
        del row[len(row) - 1]# delete it
        count = 1
        for v in row:
            newLine = newLine + " " + str(count) + ":" + v # add contents to send line
            count += 1
        newLine = newLine + "\n"
        writeToFile.append(newLine)
csvFile.close()# close input file

del writeToFile[0]# delete title

newFile = open(outputFile, "w")# create new file
for w in writeToFile:
    newFile.write(w)#add contents of the list to the new file

newFile.close()# close output file
```

Appendix B:


C:\WINDOWS\system32\cmd.exe
C:\Users\User\Desktop\libsvm-master\windows>svm-scale.exe -s param testingSeta.dat

Appendix C

C:\WINDOWS\system32\cmd.exe

```
C:\Users\User\Desktop\libsvm-master\tools>python grid.py train.dat
gnuplot executable not found
[local] 5 -7 63.9474 (best c=32.0, g=0.0078125, rate=63.9474)
[local] -1 -7 59.2105 (best c=32.0, g=0.0078125, rate=63.9474)
[local] 5 -1 64.7368 (best c=32.0, g=0.5, rate=64.7368)
[local] -1 -1 62.6316 (best c=32.0, g=0.5, rate=64.7368)
[local] 11 -7 66.3158 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 11 -1 62.6316 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 5 -13 58.6842 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -1 -13 54.4737 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 11 -13 63.9474 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 -7 55.5263 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 -1 60.2632 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 -13 54.4737 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 5 1 62.6316 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -1 1 63.1579 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 11 1 63.1579 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 1 60.2632 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 -7 65.0 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 -1 61.0526 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 -13 61.5789 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 1 62.3684 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 5 -11 61.3158 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -1 -11 54.4737 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 11 -11 65.0 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 -11 54.4737 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 -11 64.2105 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 3 -7 61.5789 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 3 -1 64.2105 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 3 -13 54.4737 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 3 1 62.8947 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 3 -11 58.6842 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 5 -5 63.6842 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -1 -5 60.5263 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 11 -5 65.2632 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] -3 -5 60.2632 (best c=2048.0, g=0.0078125, rate=66.3158)
[local] 9 -5 66.8421 (best c=512.0, g=0.03125, rate=66.8421)
[local] 3 -5 63.1579 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -7 65.7895 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -1 62.6316 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -13 66.0526 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 1 63.6842 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -11 66.0526 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -5 63.9474 (best c=512.0, g=0.03125, rate=66.8421)
[local] 5 -15 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] -1 -15 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 11 -15 61.5789 (best c=512.0, g=0.03125, rate=66.8421)
[local] -3 -15 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 9 -15 61.3158 (best c=512.0, g=0.03125, rate=66.8421)
[local] 3 -15 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -15 65.0 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -7 55.5263 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -1 60.0 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -13 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 1 59.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -11 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -5 59.2105 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -15 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 5 3 63.9474 (best c=512.0, g=0.03125, rate=66.8421)
[local] -1 3 63.1579 (best c=512.0, g=0.03125, rate=66.8421)
[local] 11 3 65.0 (best c=512.0, g=0.03125, rate=66.8421)
[local] -3 3 61.5789 (best c=512.0, g=0.03125, rate=66.8421)
[local] 9 3 64.7368 (best c=512.0, g=0.03125, rate=66.8421)
[local] 3 3 65.0 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 3 64.7368 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 3 60.7895 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -7 64.7368 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -1 62.8947 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -13 61.3158 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 1 64.7368 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -11 61.5789 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -5 65.2632 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -15 58.6842 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 3 64.7368 (best c=512.0, g=0.03125, rate=66.8421)
[local] 5 -9 61.5789 (best c=512.0, g=0.03125, rate=66.8421)
[local] -1 -9 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 11 -9 65.7895 (best c=512.0, g=0.03125, rate=66.8421)
[local] -3 -9 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 9 -9 65.2632 (best c=512.0, g=0.03125, rate=66.8421)
[local] 3 -9 61.3158 (best c=512.0, g=0.03125, rate=66.8421)
[local] 15 -9 65.5263 (best c=512.0, g=0.03125, rate=66.8421)
[local] -5 -9 54.4737 (best c=512.0, g=0.03125, rate=66.8421)
[local] 7 -9 63.9474 (best c=512.0, g=0.03125, rate=66.8421)
```

Appendix D

```
C:\WINDOWS\system32\cmd.exe - python  grid.py -log2c 8,10,0.1 -log2g -6,-4,0.1  train.dat
[local] 8.8 -4.7 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -5.2 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -4.1 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -4.1 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -4.1 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -4.1 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -4.1 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -4.1 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.0 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.5 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.4 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.8 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.7 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.2 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.1 64.4737 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -5.9 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -5.9 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -5.9 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -5.9 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -5.9 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -5.9 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.9 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.0 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.5 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.4 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.8 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.2 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.1 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.9 64.7368 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -4.8 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -4.8 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -4.8 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -4.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -4.8 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -4.8 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.0 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.5 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.4 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.2 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.1 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.9 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.8 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -5.3 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -5.3 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -5.3 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -5.3 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -5.3 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -5.3 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.3 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.3 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.3 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.0 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.5 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.4 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.8 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.7 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.2 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.1 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.9 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.8 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.3 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -4.2 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -4.2 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -4.2 64.7368 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -4.2 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -4.2 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -4.2 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.2 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.2 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.2 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.2 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.0 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.5 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -4.4 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.8 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -4.7 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.2 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -4.1 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.9 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
```

Appendix E

C:\WINDOWS\system32\cmd.exe

```
[local] 9.0 -5.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -5.7 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -5.7 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -5.7 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -5.7 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -5.7 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -5.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -5.7 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -5.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -5.7 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -5.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.4 -5.7 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.5 -5.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.9 -5.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 10.0 -5.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.0 -5.7 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.1 -5.7 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.6 -5.7 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.7 -5.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.0 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.5 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.4 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.8 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.7 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.2 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.1 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.9 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.3 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.2 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.6 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.5 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.1 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.0 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -6.0 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.9 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.4 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.3 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -5.7 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.0 -4.6 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.5 -4.6 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.6 -4.6 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.2 -4.6 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.3 -4.6 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.8 -4.6 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.9 -4.6 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.1 -4.6 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.2 -4.6 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.7 -4.6 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.8 -4.6 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.4 -4.6 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.5 -4.6 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.9 -4.6 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 10.0 -4.6 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.0 -4.6 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.1 -4.6 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.6 -4.6 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.7 -4.6 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 8.3 -4.6 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.0 67.6316 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.5 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.4 65.5263 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.8 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.7 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.2 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.1 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.9 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.8 66.3158 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.3 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.2 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.6 67.1053 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.5 65.7895 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.1 67.3684 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.0 65.2632 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -6.0 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.9 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.4 66.8421 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.3 65.0 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -5.7 66.5789 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
[local] 9.4 -4.6 66.0526 (best c=362.038671968, g=0.0473661427034, rate=67.6316)
362.038671968 0.0473661427034 67.6316

C:\Users\User\Desktop\libsvm-master\tools>
```

Appendix F

C:\WINDOWS\system32\cmd.exe

```
C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC              (multi-class classification)
        1 -- nu-SVC             (multi-class classification)
        2 -- one-class SVM
        3 -- epsilon-SVR        (regression)
        4 -- nu-SVR             (regression)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-q : quiet mode (no outputs)

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -g 0.0473661427034 -c 362.038671968
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC              (multi-class classification)
        1 -- nu-SVC             (multi-class classification)
        2 -- one-class SVM
        3 -- epsilon-SVR        (regression)
        4 -- nu-SVR             (regression)
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/num_features)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking : whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates : whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight : set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-q : quiet mode (no outputs)

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -g 0.0473661427034 -c 362.038671968 train.dat
.........*.........*
optimization finished, #iter = 6891
nu = 0.679019
obj = -88940.469600, rho = -13.388583
nSV = 275, nBSV = 241
Total nSV = 275

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe
Usage: svm-predict [options] test_file model_file output_file
options:
-b probability_estimates: whether to predict probability estimates, 0 or 1 (default 0); for one-class SVM only 0 is supported
-q : quiet mode (no outputs)

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat train.dat.model train.dat
Accuracy = 60.8333% (73/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat train.dat.model trainpred.dat
Accuracy = 60.8333% (73/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>
```

Appendix G

```python
testFile = "test.dat"
trainFile = "trainpred.dat"

actual = []#list of acutal instances
predicted = []#list of predicted instances
#Combine files into a single list
with open (testFile) as textFile1:# open test file
    with open(trainFile) as textFile2:# open predicted file
        for e in range(0, 120): # look at all instances
            line = textFile1.readline().rstrip('\n')#read line from test

            actual.append(int(line[0:2]))
            line = textFile2.readline().rstrip('\n')
            predicted.append(int(line[0:2]))
textFile1.close()
textFile2.close()

#Analysis
incorrect = 0
incorrectly_predicted_nondefective = 0
incorrectly_predicted_defective = 0
act_defective = 0
act_nondefective = 0
pred_defective = 0
pred_nondefective = 0

for e in range(0,120):
    original = actual[e]
    classification = predicted[e]

    if original == 1 and classification == -1:
        incorrectly_predicted_nondefective +=1

    if original == -1 and classification == 1:
        incorrectly_predicted_defective += 1

    if original == 1:
        act_defective += 1

    if original == -1:
        act_nondefective += 1

    if classification == 1:
        pred_defective += 1

    if classification == -1:
        pred_nondefective += 1


incorrect = incorrectly_predicted_nondefective + incorrectly_predicted_defective

print "No of incorrect instances: " + str(incorrect)
print "No of instances that labeled as defective and incorrectly predicted as non-defective: " + str(incorrectly_predicted_nondefective)
print "No of instances that labeled as non-defective and incorrectly predicted as defective: " + str(incorrectly_predicted_defective)
print "No of actual defective: " + str(act_defective)
print "No of actual non-defective: " + str(act_nondefective)
print "No of predicted defective: " + str(pred_defective)
print "No of predicted non-defective: " + str(pred_nondefective)
```

Appendix H

C:\WINDOWS\system32\cmd.exe

```
C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
Traceback (most recent call last):
  File "Svm-Compare.py", line 9, in <module>
    with open (testFile) as textFile1:
IOError: [Errno 2] No such file or directory: 'test.dat'

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of actual defective: 0
No of actual non-defective: 120
No of predicted defective: 120
No of predicted non-defective: 0

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of actual defective: 60
No of actual non-defective: 60
No of predicted defective: 29
No of predicted non-defective: 91

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of missclassified instances: 47
39 instances that labeled as defective and incorrectly predicted as non-defective
8 instances that labeled as non-defective and incorrectly predicted as defective

No of actual defective: 60
No of actual non-defective: 60
No of predicted defective: 29
No of predicted non-defective: 91

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of missclassified instances: 47
No of instances that labeled as defective and incorrectly predicted as non-defective: 39
No of instances that labeled as non-defective and incorrectly predicted as defective8
No of actual defective: 60
No of actual non-defective: 60
No of predicted defective: 29
No of predicted non-defective: 91

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of incorrect instances: 0
No of instances that labeled as defective and incorrectly predicted as non-defective: 39
No of instances that labeled as non-defective and incorrectly predicted as defective: 8
No of actual defective: 60
No of actual non-defective: 60
No of predicted defective: 29
No of predicted non-defective: 91

C:\Users\User\Desktop\AI - Coursework 2>python Svm-Compare.py
No of incorrect instances: 47
No of instances that labeled as defective and incorrectly predicted as non-defective: 39
No of instances that labeled as non-defective and incorrectly predicted as defective: 8
No of actual defective: 60
No of actual non-defective: 60
No of predicted defective: 29
No of predicted non-defective: 91

C:\Users\User\Desktop\AI - Coursework 2>
```

Appendix I

C:\WINDOWS\system32\cmd.exe

```
C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat train.dat.model trainpred.dat
Accuracy = 60.8333% (73/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -c 10 linear10.dat
can't open input file linear10.dat

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -c 10 linear10.dat
*
optimization finished, #iter = 353
nu = 0.761022
obj = -2786.493036, rho = -1.986552
nSV = 296, nBSV = 282
Total nSV = 296

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -c 10 linear100.dat
*
optimization finished, #iter = 353
nu = 0.761022
obj = -2786.493036, rho = -1.986552
nSV = 296, nBSV = 282
Total nSV = 296

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -c 100 linear100.dat
...*.*
optimization finished, #iter = 1546
nu = 0.697854
obj = -25215.107082, rho = -5.370211
nSV = 279, nBSV = 256
Total nSV = 279

C:\Users\User\Desktop\libsvm-master\windows>svm-train.exe -c 1000 linear1000.dat
....................*.....................*........*
optimization finished, #iter = 18928
nu = 0.619781
obj = -226640.637343, rho = -12.777832
nSV = 263, nBSV = 217
Total nSV = 263

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat linear10.dat.model linear10pred.dat
Accuracy = 60.8333% (73/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat linear100.dat.model linear100pred.dat
Accuracy = 60.8333% (73/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>svm-predict.exe test.dat linear1000.dat.model linear1000pred.dat
Accuracy = 61.6667% (74/120) (classification)

C:\Users\User\Desktop\libsvm-master\windows>_
```