Data Mining – Programming Assignment 2
Magesh Rajasekaran - 101676478

First Question:

SVM Classifier with Class Labels 1 and 2:

Code: (PA2_DM)

```
%# Initializing labels
Labels(1:5000)=0;
Labels(5001:10000)=1;
%# Reading Input
dataInput=load('data.txt');
%# Kfold validation
countForKFold=10;
crossValidationValue = crossvalind('Kfold',Labels, k);
classValue = classperf(Labels);
%# SVM Training with Polynomial Kernel Function of degree 3
for i = 1:countForKFold
      testIndiciesValue = (crossValidationValue == i);
      trainIndiciesValue = ~testIndiciesValue;
      svmModel = fitcsvm( dataInput(trainIndiciesValue,:),
      Labels(trainIndiciesValue),'BoxConstraint',2e-1,
      'KernelFunction','polynomial', 'PolynomialOrder',3);
      pred = predict(svmModel, dataInput(testIndiciesValue,:));
      classValue = classperf(classValue, pred, testIndiciesValue);
end
%# accuracy
classValue.CorrectRate
%# confusion matrix
classValue.CountingMatrix
```

Parameter: Kernel Function used was Polynomial with the degree 3

Result:

```
>> PA2_DM

Accuracy =

   0.9379


Confusion Matrix =

        4482        103
        518         4897
           0         0
```

Neural Network with Class 1 and 2:

Code:

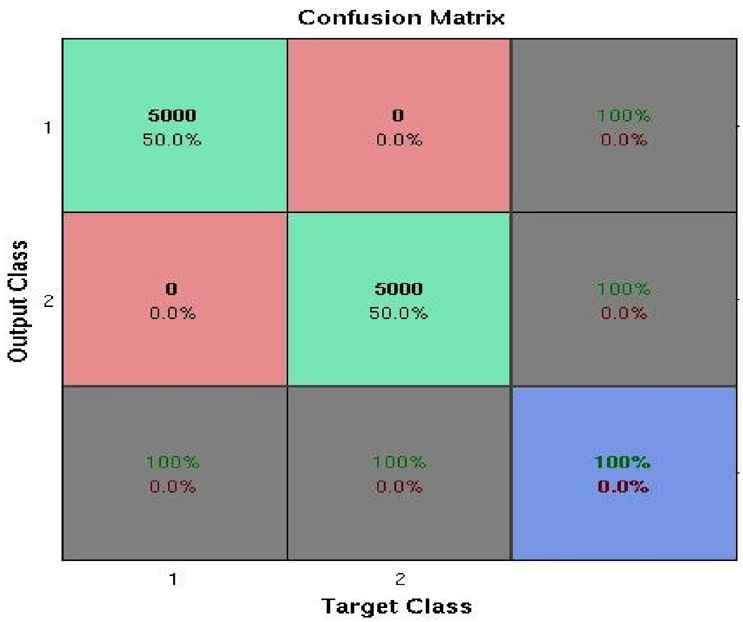
```
dataInput= load('data.txt');

%labelling the the rows as per the given problem statement.
Label(1:5000,1)=0;
Label(5001:10000,1)=1;
x = dataInput';
t = Label';

%cross validation by cvpartition
crossValidationPartition=cvpartition(10000,'kfold',10);
for i=1:10
    indexValue(:,i)=test(crossValidationPartition,i);
end

indexValue=indexValue';
% K folding of Train and Test indicies
for Kvalue=1:10
    count1=1;
    count2=1;
    rows=1;
    while (rows <=10000)

        if(indexValue(Kvalue,rows)==0)
            trainIndicies(Kvalue,count1)=rows;
            count1=count1+1;
        else
            testIndicies(Kvalue,count2)=rows;
            count2=count2+1;

        end
        rows=rows+1;
    end
 count1=1;
 count2=1;
end

% Create a Pattern Recognition Network
hiddenLayerSize = 10;
net = patternnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
for l=1:10
    net.divideFcn = 'divideind';  % Divide data randomly
    net.divideMode = 'sample';  % Divide up every sample
    net.divideParam.trainInd =trainIndicies(l,1:9000);
    net.divideParam.testInd =testIndicies(1,:);
    %net.divideParam.valRatio = 5/100;

    % Choose a Performance Function
    net.performFcn = 'crossentropy';  % Cross-Entropy

    % Choose Plot Functions
    net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
        'plotconfusion', 'plotroc'};

    % Train the Network
    [net,tr] = train(net,x,t);

    % Test the Network
    y = net(x);
    e = gsubtract(t,y);
    performance = perform(net,t,y)
    tind = vec2ind(t);
    yind = vec2ind(y);
    percentErrors = sum(tind ~= yind)/numel(tind);
    performance = perform(net, t,y);

    % Recalculate Training, Validation and Test Performance
    trainTargets = t .* tr.trainMask{1};
    %valTargets = t .* tr.valMask{1};
    testTargets = t .* tr.testMask{1};
    trainPerformance = perform(net,trainTargets,y)
    %valPerformance = perform(net,valTargets,y)
    testPerformance = perform(net,testTargets,y)
    [c,cm,ind,per] = confusion(t,y);
    accuracyFinal(1,1)=((1-c)*100);

    % View the Network
    %view(net)

    figure, plotconfusion(t,y)
end
```

Parameter: Hidden Neurons – 10

Result:

Accuracy = 1.000

Confusion Matrix Plot:



Second Question:

SVM Classifier:

Code: (PA2_DM2_SVM)

```
%# Initializing labels
Labels=ones(10000,1);
Labels([1:500,1001:1500,2001:2500,3001:3500,4001:4500,5001:5500,6001:6500,7001:7500
,8001:8500,9001:9500])=2;
%# Reading Input
dataInput=load('data.txt');
%# Kfold validation
countForKFold=10;
crossValidationValue = crossvalind('Kfold',Labels, k);
classValue = classperf(Labels);
%# SVM Training with Polynomial Kernel Function of degree 3
for i = 1:countForKFold
      testIndiciesValue = (crossValidationValue == i);
      trainIndiciesValue = ~testIndiciesValue;
      svmModel = fitcsvm( dataInput(trainIndiciesValue,:),
      Labels(trainIndiciesValue),'BoxConstraint',2e-1,
      'KernelFunction','polynomial', 'PolynomialOrder',3);
      pred = predict(svmModel, dataInput(testIndiciesValue,:));
      classValue = classperf(classValue, pred, testIndiciesValue);
end
%# accuracy
classValue.CorrectRate
%# confusion matrix
classValue.CountingMatrix
```

Result:

```
>> PA2_DM2_SVM

Accuracy =

    0.5018


Confusion Matrix =

        2582        2564
        2418        2436
           0           0
```

Parameter: Kernel Function used was Polynomial with the degree 3

Neural Network:

Code:

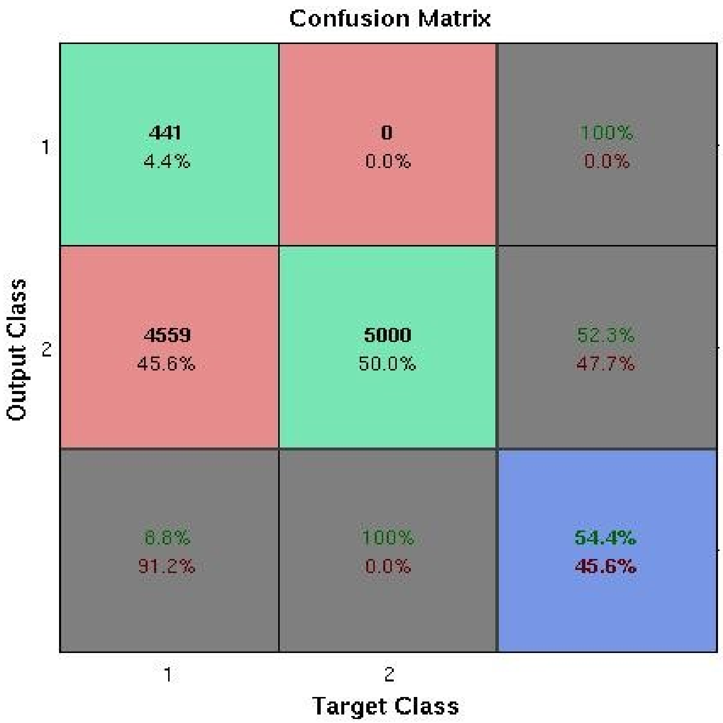
```
%labelling the the rows as per the given problem statement
Label=zeros(10000,1);
Label([1:500,1001:1500,2001:2500,3001:3500,4001:4500,5001:5500,6001:6500,7001:7500,
8001:8500,9001:9500])=1;
Label=ones(10000,2);
Label([1:500,1001:1500,2001:2500,3001:3500,4001:4500,5001:5500,6001:6500,7001:7500,
8001:8500,9001:9500])=0;
dataInput= load('data.txt');

x = dataInput';
t = Label';

%cross validation by cvpartition
crossValidationPartition=cvpartition(10000,'kfold',10);
for i=1:10
    indexValue(:,i)=test(crossValidationPartition,i);
end

indexValue=indexValue';
% K folding of Train and Test indicies
for Kvalue=1:10
    count1=1;
    count2=1;
    rows=1;
    while (rows <=10000)

        if(indexValue(Kvalue,rows)==0)
            trainIndicies(Kvalue,count1)=rows;
            count1=count1+1;
        else
            testIndicies(Kvalue,count2)=rows;
            count2=count2+1;

        end
        rows=rows+1;
    end
 count1=1;
 count2=1;
end

% Create a Pattern Recognition Network
hiddenLayerSize = 10;
net = patternnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.input.processFcns = {'removeconstantrows','mapminmax'};
net.output.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
for l=1:10
    net.divideFcn = 'divideind';  % Divide data randomly
    net.divideMode = 'sample';  % Divide up every sample
    net.divideParam.trainInd =trainIndicies(l,1:9000);
    net.divideParam.testInd =testIndicies(l,:);
    %net.divideParam.valRatio = 5/100;
    % Choose a Performance Function
    net.performFcn = 'crossentropy';  % Cross-Entropy

    % Choose Plot Functions
    net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
        'plotconfusion', 'plotroc'};

    % Train the Network
    [net,tr] = train(net,x,t);

    % Test the Network
    y = net(x);
    e = gsubtract(t,y);
    performance = perform(net,t,y)
    tind = vec2ind(t);
    yind = vec2ind(y);
    percentErrors = sum(tind ~= yind)/numel(tind);
    performance = perform(net, t,y);

    % Recalculate Training, Validation and Test Performance
    trainTargets = t .* tr.trainMask{1};
    %valTargets = t .* tr.valMask{1};
    testTargets = t .* tr.testMask{1};
    trainPerformance = perform(net,trainTargets,y)
    %valPerformance = perform(net,valTargets,y)
    testPerformance = perform(net,testTargets,y)
    [c,cm,ind,per] = confusion(t,y);
    accuracyFinal(1,1)=((1-c)*100);

    % View the Network
    %view(net)

    figure, plotconfusion(t,y)
end
```

Parameter: Hidden Neurons – 10

Result:

Accuracy: 53.4%

Confusion Matrix:



	1	2	
1	441 4.4%	0 0.0%	100% 0.0%
2	4559 45.6%	5000 50.0%	52.3% 47.7%
	8.8% 91.2%	100% 0.0%	54.4% 45.6%

Reference:

[1] Matlab Neural Network Pattern Recognition tool for Neural Network Problem - http://www.mathworks.com/products/neural-network/features.html#data-fitting%2C-clustering%2C-and-pattern-recognition.

[2] Fitcsvm method - http://www.mathworks.com/help/stats/fitcsvm.html

[3] http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

[4] http://www.svms.org/parameters/