%Decision Tree

fprintf('Decision Tree')

pca\_input1 = myoupdated(:, 2:9);

[coeff1, score1, latent1] = pca(pca\_input1);

reduced\_dim1 = coeff1;

reduced\_matrix1 = pca\_input1 \* reduced\_dim1;

%[trainA,testA] = divideblock(reduced\_matrix1, 0.7, 0.3);

cv1 = cvpartition(size(reduced\_matrix1,1),'HoldOut',0.3);

idx = cv1.test;

dataTrain1 = reduced\_matrix1(~idx,:);

dataTest1 = reduced\_matrix1(idx,:);

org\_y = myoupdated(:, 10);

cv11 = cvpartition(size(org\_y,1),'HoldOut',0.3);

idx1 = cv11.test;

yTrain1 = org\_y(~idx1,:);

yTest1 = org\_y(idx1,:);

tree = fitctree(dataTrain1, yTrain1);

%svmc.predict(dataTest1)

pred\_label1 = predict(tree,dataTest1);

tp=0;tn=0;fp=0;fn=0;

for i=1:size(yTest1,1)

act=yTest1(i);

pred1=pred\_label1(i);

if act==1 && pred1==1

tp=tp+1;

elseif act==1 && pred1==0

fn=fn+1;

elseif act==0 && pred1==0

tn=tn+1;

else

fp=fp+1;

end

end

precision1=tp/(tp+fp)

recall1=tp/(tp+fn)

f1score1=2\*(precision1\*recall1)/(recall1+precision1)

%SVM

fprintf('SVM')

pca\_input = myoupdated(:, 2:11);

[coeff, score, latent] = pca(pca\_input);

reduced\_dim = coeff;

reduced\_matrix = pca\_input \* reduced\_dim;

%[trainA,testA] = divideblock(reduced\_matrix, 0.7, 0.3);

cv = cvpartition(size(reduced\_matrix,1),'HoldOut',0.3);

idx = cv.test;

dataTrain = reduced\_matrix(~idx,:);

dataTest = reduced\_matrix(idx,:);

org\_y = myoupdated(:, 12);

cv1 = cvpartition(size(org\_y,1),'HoldOut',0.3);

idx1 = cv1.test;

yTrain = org\_y(~idx1,:);

yTest = org\_y(idx1,:);

svmc = fitcsvm(dataTrain, yTrain);

%svmc.predict(dataTest)

pred\_label = predict(svmc,dataTest);

tp=0;tn=0;fp=0;fn=0;

for i=1:size(yTest,1)

actual=yTest(i);

pred=pred\_label(i);

if actual==1 && pred==1

tp=tp+1;

elseif actual==1 && pred==0

fn=fn+1;

elseif actual==0 && pred==0

tn=tn+1;

else

fp=fp+1;

end

end

precision=tp/(tp+fp)

recall=tp/(tp+fn)

f1score=2\*(precision\*recall)/(recall+precision)

% %Neural Networks

% Solve an Input-Output Fitting problem with a Neural Network

% Script generated by Neural Fitting app

% Created 29-Nov-2018 10:48:33

%

% This script assumes these variables are defined:

%

% reducedmatrix - input data.

% org\_y - target data.

x = reducedmatrix';

t = org\_y';

% Choose a Training Function

% For a list of all training functions type: help nntrain

% 'trainlm' is usually fastest.

% 'trainbr' takes longer but may be better for challenging problems.

% 'trainscg' uses less memory. Suitable in low memory situations.

trainFcn = 'trainbr'; % Bayesian Regularization backpropagation.

% Create a Fitting Network

hiddenLayerSize = 30;

net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing

net.divideParam.trainRatio = 60/100;

net.divideParam.valRatio = 5/100;

net.divideParam.testRatio = 35/100;

% Train the Network

[net,tr] = train(net,x,t);

% Test the Network

y = net(x);

e = gsubtract(t,y);

performance = perform(net,t,y)

tp=0;tn=0;fp=0;fn=0;

for i=1:size(t,2)

act=t(i);

pred1=y(i);

if act==1 && pred1==1

tp=tp+1;

elseif act==1 && pred1==0

fn=fn+1;

elseif act==0 && pred1==0

tn=tn+1;

else

fp=fp+1;

end

end

precision1=tp/(tp+fp)

recall1=tp/(tp+fn)

f1score1=2\*(precision1\*recall1)/(recall1+precision1)

% View the Network

view(net)

% Plots

% Uncomment these lines to enable various plots.

figure, plotperform(tr)

figure, plottrainstate(tr)

figure, ploterrhist(e)

figure, plotregression(t,y)

figure, plotfit(net,x,t)