**i) MATLAB Code for applying DoA Index Model to Test Patient Data Set and Measuring Performance**

%%

%%%PROJECT FINAL REPORT

%%% Load test data set

test1 = readtable('Data set (stage 2) with BIS value','Sheet','Test1');

test2 = readtable('Data set (stage 2) with BIS value','Sheet','Test2');

test3 = readtable('Data set (stage 2) with BIS value','Sheet','Test3');

test4 = readtable('Data set (stage 2) with BIS value','Sheet','Test4');

test5 = readtable('Data set (stage 2) with BIS value','Sheet','Test5');

%%

%%%Total data set

test = readtable('Data set (stage 2) with BIS value','Sheet','TestTotal');

%%

%%% Apply feature selection and linear model to test data

X = [test.x2, test.x4, test.x5];

linResult2 = predict(lin,X);

%%

%%%Test Model on combined test data set

%%%Input data for model testing from combined test data set

PInput = transpose(linResult2);

%%%Target values from combined test data set

Q = table2array(test)

PResult = transpose(Q(:, 1));

%%

%%%Calculate model performance for combined test data set

predP = sim(net,PInput);

[RP,PP,RLP,RUP] = corrcoef(predP,PResult);

eP = PResult-predP;

perfP = mae(eP); %%MAE

errP = mse(net,PResult,predP) %%MSE

%%

%%Plot Results of Test

%%%BIS an predP values for whole data set

figure; hold on;

plot (predP);

plot(PResult,'m');

%%

%%%%Scatter plot of NN results and BIS scores

figure; hold on;

plot(predP, PResult, 'x');

h = lsline;

h.Color = 'k';

%%

%%% Error histogram

figure;

ploterrhist(eP);

%%

% Apply model to test1

%Calculate inputs from linear model

TP1 = [test1.x2, test1.x4, test1.x5]

TR1 = [test1.BIS];

linResultTP1 = predict(lin,TP1);

TI1 = transpose(linResultTP1);

TR1 = transpose(TR1);

predT1 = sim(net,TI1);

[RP,PP,RLP,RUP] = corrcoef(predT1,TR1);

eT1 = TR1-predT1;

perfT1 = mae(eT1); %%MAE

errT1 = mse(net,TR1,predT1) %%MSE

%%Plot Results of Test 1

figure; hold on;

plot (predT1);

plot(TR1,'m');

%% %%

%%% Apply model to test2

%Calculate inputs from linear model

TP2 = [test2.x2, test2.x4, test2.x5]

TR2 = [test2.BIS];

linResultTP2 = predict(lin,TP2);

TI2 = transpose(linResultTP2);

TR2 = transpose(TR2);

predT2 = sim(net,TI2);

[RP,PP,RLP,RUP] = corrcoef(predT2,TR2);

eT2 = TR2-predT2;

perfT2 = mae(eT2); %%MAE

errT2 = mse(net,TR2,predT2) %%MSE

%%Plot Results of Patient test2

figure; hold on;

plot (predT2);

plot(TR2,'m');

%%

% Apply model to test3

%Calculate inputs from linear model

TP3 = [test3.x2, test3.x4, test3.x5]

TR3 = [test3.BIS];

linResultTP3 = predict(lin,TP3);

TI3 = transpose(linResultTP3);

TR3 = transpose(TR3);

predT3 = sim(net,TI3);

[RP,PP,RLP,RUP] = corrcoef(predT3,TR3);

eT3 = TR3-predT3;

perfT3 = mae(eT3); %%MAE

errT3 = mse(net,TR3,predT3) %%MSE

%%Plot Results of Patient test3

figure; hold on;

plot (predT3);

plot(TR3,'m');

%%

%%% Apply model to test4

%Calculate inputs from linear model

TP4 = [test4.x2, test4.x4, test4.x5]

TR4 = [test4.BIS];

linResultTP4 = predict(lin,TP4);

TI4 = transpose(linResultTP4);

TR4 = transpose(TR4);

predT4 = sim(net,TI4);

[RP,PP,RLP,RUP] = corrcoef(predT4,TR4);

eT4 = TR4-predT4;

perfT4 = mae(eT4); %%MAE

errT4 = mse(net,TR4,predT4) %%MSE

%%Plot Results of Patient test4

figure; hold on;

plot (predT4);

plot(TR4,'m');

%%

% Apply model to test5

%Calculate inputs from linear model

TP5 = [test5.x2, test5.x4, test5.x5]

TR5 = [test5.BIS];

linResultTP5 = predict(lin,TP5);

TI5 = transpose(linResultTP5);

TR5 = transpose(TR5);

predT5 = sim(net,TI5);

[RP,PP,RLP,RUP] = corrcoef(predT5,TR5);

eT5 = TR5-predT5;

perfT5 = mae(eT5); %%MAE

errT5 = mse(net,TR5,predT5) %%MSE

%%Plot Results of Patient test5

figure; hold on;

plot (predT5);

plot(TR5,'m');

%%

%%%Confusion Matrix - Total patient test set

predP = transpose(predP);

PResult = transpose(PResult);

%%

%%%%Convert predicted results to range 0-40, 40-60, 60-80, 80-100

A = predP(:,1) < 100;

predP(A,2) = 100;

A = predP(:,1) < 80;

predP(A,2) = 80;

A = predP(:,1) < 60;

predP(A,2) = 60;

A = predP(:,1) < 40;

predP(A,2) = 40;

%%

%%% Convert actual results to range 0-40, 40-60, 60-80, 80-100

A = PResult(:,1) < 100;

PResult(A,2) = 100;

A = PResult(:,1) < 80;

PResult(A,2) = 80;

A = PResult(:,1) < 60;

PResult(A,2) = 60;

A = PResult(:,1) < 40;

PResult(A,2) = 40;

BIS1 = PResult(:,2);

DoA1 = predP(:,2);

CM = confusionmat(BIS1, DoA1);

CC = confusionchart(BIS1,DoA1);

**ii) MATLAB Code for DoA Index Model Improvements**

%%

%%% FINAL PROJECT IMPROVE MODEL

%%%Updated model #1

%%%Combine use 10 training patients instead of 7 and increase number of

%%%data

X = [eeg.x2, eeg.x4, eeg.x5];

%%%Target values from eeg data

TrainResult = g(:, 1);

%%

%%%DECISION TREE + NEURAL NETWORK

%%% Build regression tree with training data and apply tree to data to

%%% obtain prediction for neural network input

tree = fitrtree(X, TrainResult);

Pred = predict(tree,X)

Pred = transpose(Pred);

TrainResult = transpose(TrainResult);

%%

%%%Build neural Network

netconf = [10]

net = feedforwardnet(netconf, 'trainlm');

net = train(net,Pred,TrainResult);

net.trainParam.epochs=1000;

net.trainParam.goal=0.05;

net.trainParam.show=50;

net.trainParam.lr=0.05;

net.trainParam.mc=0.8;

net.divideFcn= 'dividerand';

%%

%%% Apply new model to test data set

XT = [test.x2, test.x4, test.x5];

%%

%%%Test Model on combined test data set

%%%Input data for model testing from combined test data set

PTest = predict(tree,XT)

%%

PTest = transpose(PTest);

%%%Target values from combined test data set

Q = table2array(test)

PResult = transpose(Q(:, 1));

%%

%%%Calculate model performance for combined test data set

predTest = sim(net,PTest);

[RP,PP,RLP,RUP] = corrcoef(predTest,PResult);

eP = PResult-predTest;

perfP = mae(eP); %%MAE

errP = mse(net,PResult,predTest); %%MSE

%%

%%% UPDATED MODEL #2 - SVR + NEURAL NETWORK

%%%Build neural Network

%%% Input into SVR

Input = X;

%%

Input = transpose(Input);

TrainResult = transpose(TrainResult);

%%

%%%%

%%Build SVM Model

mdl2 = fitrsvm(Input,TrainResult,'Standardize', true, 'KernelFunction', 'gaussian');

%%

TT = predict(mdl2, Input);

%%

TT = transpose(TT);

%%

netconf = [10]

net = feedforwardnet(netconf, 'trainlm');

net = train(net,TT,TrainResult);

net.trainParam.epochs=1000;

net.trainParam.goal=0.05;

net.trainParam.show=50;

net.trainParam.lr=0.05;

net.trainParam.mc=0.8;

net.divideFcn= 'dividerand';

%%

%%PERFORMANCE OF MODEL

%%%Test Data

%%Apply SVMR model

XT = transpose(XT);

predT2 = predict(mdl2,XT);

%%

predT2 = transpose(predT2);

%%

%%Apply NN Model

IT = sim (net, predT2);

%%

%%%Performance

[RP,PP,RLP,RUP] = corrcoef(IT,PResult);

eP = PResult-IT;

perfP = mae(eP); %%MAE

errP = mse(net,PResult,IT); %%MSE

R2 = corrcoef(IT,PResult);

%%

%%%UPDATED MODEL #3 - K MEANS + NEURAL NETWORK

X = transpose(X);

[idx,C] = kmeans(X,20);

%%

%%% Assign k menas values to variables in data set

XK = X;

XK = [XK, idx];

%%

%%% Assign kmeans values to observations

idx2 = XK(:,4) == 1;

XK(idx2,5) = C(1,1);

XK(idx2,6) = C(1,2);

XK(idx2,7) = C(1,3);

idx2 = XK(:,4) == 2;

XK(idx2,5) = C(2,1);

XK(idx2,6) = C(2,2);

XK(idx2,7) = C(2,3);

idx2 = XK(:,4) == 3;

XK(idx2,5) = C(3,1);

XK(idx2,6) = C(3,2);

XK(idx2,7) = C(3,3);

idx2 = XK(:,4) == 4;

XK(idx2,5) = C(4,1);

XK(idx2,6) = C(4,2);

XK(idx2,7) = C(4,3);

idx2 = XK(:,4) == 5;

XK(idx2,5) = C(5,1);

XK(idx2,6) = C(5,2);

XK(idx2,7) = C(5,3);

idx2 = XK(:,4) == 6;

XK(idx2,5) = C(6,1);

XK(idx2,6) = C(6,2);

XK(idx2,7) = C(6,3);

idx2 = XK(:,4) == 7;

XK(idx2,5) = C(7,1);

XK(idx2,6) = C(7,2);

XK(idx2,7) = C(7,3);

idx2 = XK(:,4) == 8;

XK(idx2,5) = C(8,1);

XK(idx2,6) = C(8,2);

XK(idx2,7) = C(8,3);

idx2 = XK(:,4) == 9;

XK(idx2,5) = C(9,1);

XK(idx2,6) = C(9,2);

XK(idx2,7) = C(9,3);

idx2 = XK(:,4) == 10;

XK(idx2,5) = C(10,1);

XK(idx2,6) = C(10,2);

XK(idx2,7) = C(10,3);

idx2 = XK(:,4) == 11;

XK(idx2,5) = C(11,1);

XK(idx2,6) = C(11,2);

XK(idx2,7) = C(11,3);

idx2 = XK(:,4) == 12;

XK(idx2,5) = C(12,1);

XK(idx2,6) = C(12,2);

XK(idx2,7) = C(12,3);

idx2 = XK(:,4) == 13;

XK(idx2,5) = C(13,1);

XK(idx2,6) = C(13,2);

XK(idx2,7) = C(13,3);

idx2 = XK(:,4) == 14;

XK(idx2,5) = C(14,1);

XK(idx2,6) = C(14,2);

XK(idx2,7) = C(14,3);

idx2 = XK(:,4) == 15;

XK(idx2,5) = C(15,1);

XK(idx2,6) = C(15,2);

XK(idx2,7) = C(15,3);

idx2 = XK(:,4) == 16;

XK(idx2,5) = C(16,1);

XK(idx2,6) = C(16,2);

XK(idx2,7) = C(16,3);

idx2 = XK(:,4) == 17;

XK(idx2,5) = C(17,1);

XK(idx2,6) = C(17,2);

XK(idx2,7) = C(17,3);

idx2 = XK(:,4) == 18;

XK(idx2,5) = C(18,1);

XK(idx2,6) = C(18,2);

XK(idx2,7) = C(18,3);

idx2 = XK(:,4) == 19;

XK(idx2,5) = C(19,1);

XK(idx2,6) = C(19,2);

XK(idx2,7) = C(19,3);

idx2 = XK(:,4) == 20;

XK(idx2,5) = C(20,1);

XK(idx2,6) = C(20,2);

XK(idx2,7) = C(20,3);

%%

%%% Select variables from XK for neural network model

XK2 = XK(:,5:7);

%%

XK2 = transpose(XK2);

%%

netconf = [10]

net = feedforwardnet(netconf, 'trainlm');

net = train(net,XK2,TrainResult);

net.trainParam.epochs=1000;

net.trainParam.goal=0.05;

net.trainParam.show=50;

net.trainParam.lr=0.05;

net.trainParam.mc=0.8;

net.divideFcn= 'dividerand';

%%

%%%Performance of model on test set data

[idxT,C] = kmeans(XT,20);

%%

%%% Assign k menas values to variables in data set

XKT = XT;

XKT = [XKT, idxT];

%%

%%%% Assign k mean values

idxT = XKT(:,4) == 1;

XKT(idxT,5) = C(1,1);

XKT(idxT,6) = C(1,2);

XKT(idxT,7) = C(1,3);

idxT = XKT(:,4) == 2;

XKT(idxT,5) = C(2,1);

XKT(idxT,6) = C(2,2);

XKT(idxT,7) = C(2,3);

idxT = XKT(:,4) == 3;

XKT(idxT,5) = C(3,1);

XKT(idxT,6) = C(3,2);

XKT(idxT,7) = C(3,3);

idxT = XKT(:,4) == 4;

XKT(idxT,5) = C(4,1);

XKT(idxT,6) = C(4,2);

XKT(idxT,7) = C(4,3);

idxT = XKT(:,4) == 5;

XKT(idxT,5) = C(5,1);

XKT(idxT,6) = C(5,2);

XKT(idxT,7) = C(5,3);

idxT = XKT(:,4) == 6;

XKT(idxT,5) = C(6,1);

XKT(idxT,6) = C(6,2);

XKT(idxT,7) = C(6,3);

idxT = XKT(:,4) == 7;

XKT(idxT,5) = C(7,1);

XKT(idxT,6) = C(7,2);

XKT(idxT,7) = C(7,3);

idxT = XKT(:,4) == 8;

XKT(idxT,5) = C(8,1);

XKT(idxT,6) = C(8,2);

XKT(idxT,7) = C(8,3);

idxT = XKT(:,4) == 9;

XKT(idxT,5) = C(9,1);

XKT(idxT,6) = C(9,2);

XKT(idxT,7) = C(9,3);

idxT = XKT(:,4) == 10;

XKT(idxT,5) = C(10,1);

XKT(idxT,6) = C(10,2);

XKT(idxT,7) = C(10,3);

idxT = XKT(:,4) == 11;

XKT(idxT,5) = C(11,1);

XKT(idxT,6) = C(11,2);

XKT(idxT,7) = C(11,3);

idxT = XKT(:,4) == 12;

XKT(idxT,5) = C(12,1);

XKT(idxT,6) = C(12,2);

XKT(idxT,7) = C(12,3);

idxT = XKT(:,4) == 13;

XKT(idxT,5) = C(13,1);

XKT(idxT,6) = C(13,2);

XKT(idxT,7) = C(13,3);

idxT = XKT(:,4) == 14;

XKT(idxT,5) = C(14,1);

XKT(idxT,6) = C(14,2);

XKT(idxT,7) = C(14,3);

idxT = XKT(:,4) == 15;

XKT(idxT,5) = C(15,1);

XKT(idxT,6) = C(15,2);

XKT(idxT,7) = C(15,3);

idxT = XKT(:,4) == 16;

XKT(idxT,5) = C(16,1);

XKT(idxT,6) = C(16,2);

XKT(idxT,7) = C(16,3);

idxT = XKT(:,4) == 17;

XKT(idxT,5) = C(17,1);

XKT(idxT,6) = C(17,2);

XKT(idxT,7) = C(17,3);

idxT = XKT(:,4) == 18;

XKT(idxT,5) = C(18,1);

XKT(idxT,6) = C(18,2);

XKT(idxT,7) = C(18,3);

idxT = XKT(:,4) == 19;

XKT(idxT,5) = C(19,1);

XKT(idxT,6) = C(19,2);

XKT(idxT,7) = C(19,3);

idxT = XKT(:,4) == 20;

XKT(idxT,5) = C(20,1);

XKT(idxT,6) = C(20,2);

XKT(idxT,7) = C(20,3);

%%

XKT2 = XKT(:,5:7);

%%

XKT2 = transpose(XKT2);

%%

%%% Test model performance in NN

predTestK = sim(net,XKT2);

[RP,PP,RLP,RUP] = corrcoef(predTestK,PResult);

eP = PResult-predTest;

perfP = mae(eP); %%MAE

errP = mse(net,PResult,predTest); %%MSE

%%

%%%Confusion Matrix - Total patient test set

IT = transpose(IT);

PResult = transpose(PResult);

%%

%%%%Convert predicted results to range 0-40, 40-60, 60-80, 80-100

A = IT(:,1) < 100;

IT(A,2) = 100;

A = IT(:,1) < 80;

IT(A,2) = 80;

A = IT(:,1) < 60;

IT(A,2) = 60;

A = IT(:,1) < 40;

IT(A,2) = 40;

%%

%%% Convert actual results to range 0-40, 40-60, 60-80, 80-100

A = PResult(:,1) < 100;

PResult(A,2) = 100;

A = PResult(:,1) < 80;

PResult(A,2) = 80;

A = PResult(:,1) < 60;

PResult(A,2) = 60;

A = PResult(:,1) < 40;

PResult(A,2) = 40;

BIS1 = PResult(:,2);

DoA1 = IT(:,2);

CM = confusionmat(BIS1, DoA1);

CC = confusionchart(BIS1,DoA1);