3

**Assessment Submission Form**

|  |  |
| --- | --- |
| **Student name** | Shayan Kabiri |
| **Assessment Title** | Assignment 3 |
| **Module Code** | BSEN40870 |
| **Module Title** |  |
| **Module Coordinator** | Prof. Aoife Gowen |
| **Tutor** | Gözde Özdoğan |
| **Date Submitted** | /5/2024 |
| **Date Received** |  |
| **Grade/Mark** |  |

**A SIGNED COPY OF THIS FORM MUST ACCOMPANY ALL SUBMISSIONS FOR ASSESSMENT.**

**STUDENTS SHOULD KEEP A COPY OF ALL WORK SUBMITTED**.

**Procedures for Submission and Late Submission** There are penalties for the late submission of assessments. For further information please see the University’s Policy on Late Submission of Coursework.

**Plagiarism:** the unacknowledged inclusion of another person’s writings or ideas or works, in any formally presented work (including essays, examinations, projects, laboratory reports or presentations). The penalties associated with plagiarism designed to impose sanctions that reflect the seriousness of University’s commitment to academic integrity. Ensure that you have read UCD’s student plagiarism policy.

**Declaration of Authorship**

I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.

Signed....................................................... Date ...................................................



**Title:** BSEN40870 Assignment 3: Analysis of multivariate datasets: creation and critical

evaluation of regression and classification models.

**Objective**

To create regression and classification models for the provided multivariate data using the

tools introduced in the course.

**Dataset description**  
You can download the “Assignment\_data” dataset from your Brightspace locker. This  
MATLAB workspace contains 2 sets of data and a Wavelength vector (which describes the variables in the X matrices). Set 1 contains a calibration set (“cal\_X”) of 150 samples measured at 100 different wavelengths, and a prediction set (“pred\_X”) of 50 samples, measured at the same 100 wavelengths as the calibration set. The independent variable Y represents the concentrations of 5 different components in each sample. Set 2 contains a calibration set (“cal\_X”) of 100 samples measured at 100 different wavelengths, and a prediction set (“pred\_X”) of 100 samples, measured at the same 100 wavelengths as the calibration set. The independent variable Y represents the class membership of each sample.

**Assignment Instructions**

For Set 1 of data:

* Inspect the data and explore with PCA. Identify and remove any outliers.
* Build PCR and PLSR models to predict all 5 Y variables provided
* Optimise meta-parameter selection for both PCR and PLSR
* Compare the performance of each model on the prediction set
* Apply 2 different variable selection techniques to find the most important variables
* Compare the PLSR model built on the full dataset and on the selected variables

For Set 2 of data:

* Inspect the data and explore with PCA. Identify and remove any outliers.
* Build 2 types of classification models to predict class membership
* Compare the performance of each model on the prediction set
* Apply 2 different variable selection techniques to find the most important variables
* Compare the classification model built on the full dataset and on the selected variables

**Methodology**

**Dataset 1:**

First the PCAs, the variance explained by each PCA and correlation of each response to the first five PCAs were investigated. Then the response boxplots were used to determine if there are any outliers in the dataset. Outlier datapoints were determined by quartiles method and a dataset without outliers were created.

The tuning mechanism for both Principal Component Regression (PCR) and Partial Least Squares Regression (PLSR) was based on 5-fold cross validation and fining the number of components used in the model that produces the best root mean square error (RMSE).

Two variable selection methods, minimum redundancy maximum relevance (MRMR) and F-tests feature ranking (FFR), were used to select the 10 most important variables for regression.

Models were tuned and regression metrics were produced for PCR and PLSR, for datasets with and without outliers and selected variables based on MRMR and FFR for the five responses.

**Dataset 2:**

First the PCAs, the variance explained by each PCA and degree of apparent separation by PCAs were investigated. Predictor outliers were detected using the quartiles method and a dataset without predictor outliers were created.

Responses were converted to binary and a PCR and PLSR were adapted to act as classifiers. Predicted responses closer to 1 were converted to 1 and predicted responses closer to 0 were converted to 0.

Two variable selection methods, minimum redundancy maximum relevance (MRMR) and F-tests feature ranking (FFR), were used to select the 10 most important variables for classification.

Models were tuned and classification metrics were produced for PCR and PLSR, for datasets with and without outliers and selected variables based on MRMR and FFR for response classes.

**Results & Short Discussion**

**Dataset 1:**

The dataset consists of 100 predictors and 5 responses. The training and test sets have 150 and 50 data points. The first 5 PC components explain 60.7% variance in the predictors (Figure 1). Response 1 is linearly related to PC1 (R2 = 0.73), Response 2 is linearly related to PC4 (R2 = 0.66), Response 3 is linearly related to PC1 (R2 = 0.51), Response 4 is linearly related to PC5 (R2 = 0.48) and Response 5 is linearly related to PC3 (R2 = 0.61) (Figure 2). Five datapoints had outlier responses and were discarded to create the dataset without outliers (Figure 3). PCR generally performed better for each response compared to PLSR (R2s for each response in order: 0.94, 0,84, 0.97, 0.86, 0.95). Removing outliers didn’t meaningfully affect the PCR but improved PLSR performance for response 1 by 0.02. Variable selection strategies worsened the PLSR performances in all cases except for response 1 with slight improvement of 0.02 (Table 1).

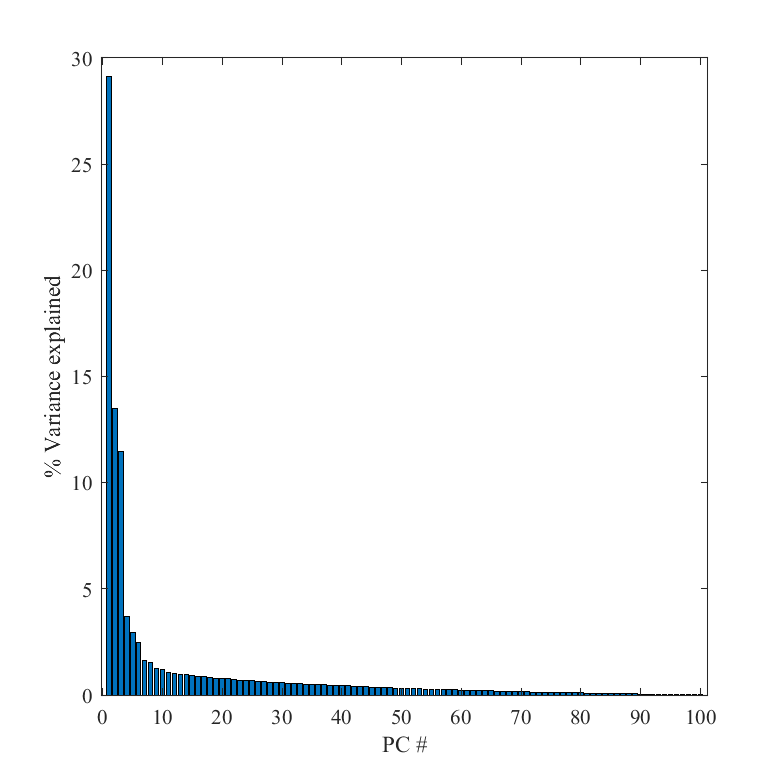
****

Figure 1.The percentage of variance explain by each principal component for dataset 1.

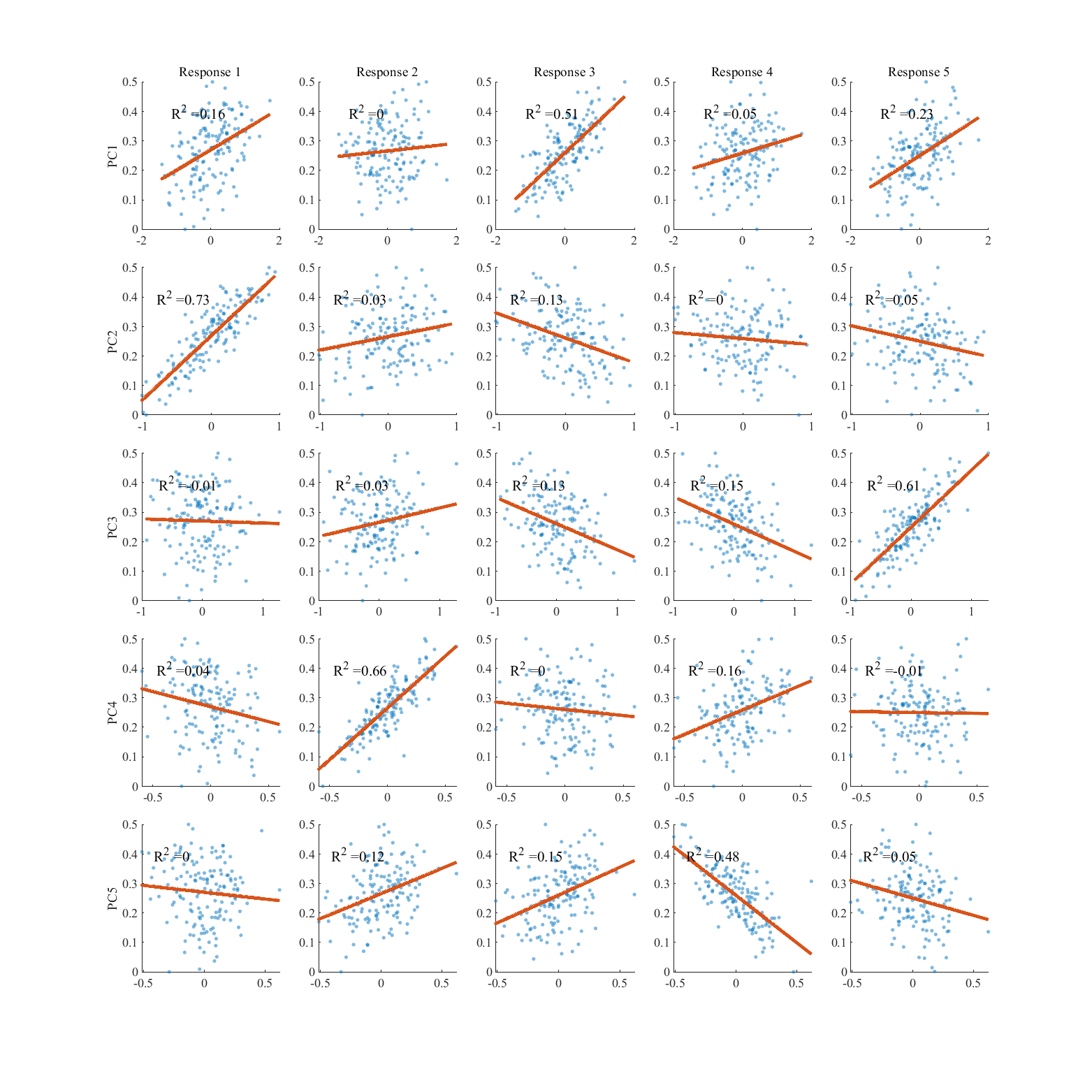
****

Figure 2. Linear relationship between the responses and first 5 principal components.

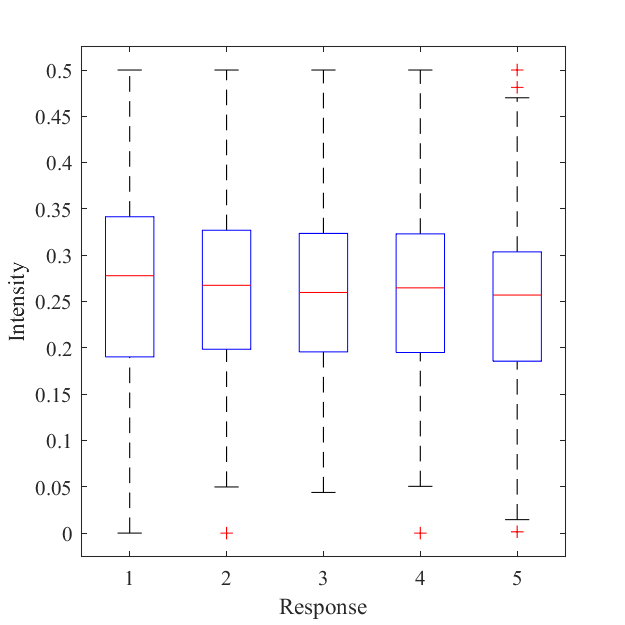
****

Figure 3. Boxplots for responses and associated outliers.

Table 1. Performance metrics for regression models.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Response | R2 test PCR | R2 test PCR WOO | R2 test PLSR | R2 test PLSR WOO | R2 test PLSR FFR | R2 test PLSR MRMR | |
| 1 | 0.94 | 0.94 | 0.9 | 0.92 | 0.92 | 0.88 |  |
| 2 | 0.84 | 0.83 | 0.85 | 0.85 | 0.66 | 0.74 |  |
| 3 | 0.97 | 0.96 | 0.97 | 0.97 | 0.92 | 0.89 |  |
| 4 | 0.86 | 0.87 | 0.87 | 0.86 | 0.41 | 0.42 |  |
| 5 | 0.95 | 0.95 | 0.95 | 0.96 | 0.87 | 0.81 |  |

PCR = principal component regression; PLSR = partial least square regression WOO = without outlier; FFS = F-tests feature ranking; MRMR = minimum redundancy maximum relevance

**Dataset 2:**

The dataset consists of 100 predictors and 2 classes. The training and test sets each have 200 data points. The first 5 PC components explain 75.6% variance in the predictors (Figure 4). Classes are well distinguishable based on the first four PCs (Figure 5). 47 datapoints had outlier predictors and were discarded to create the dataset without outliers. PCA classifier worked better than PLS classifier with test accuracy of 0.995. Removing outliers didn’t affect the PCA classifier performance but improved PLS classifier accuracy by 0.045. Variable selection strategies worsened PCA classifier performances but improved PLS classifier performances in both cases (Table 2).

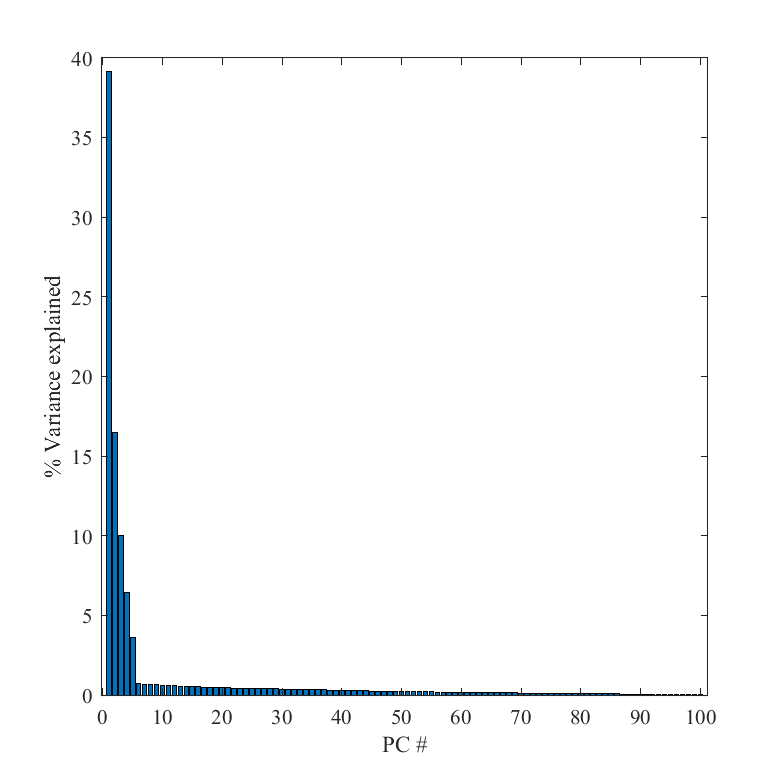
****

Figure 4. The percentage of variance explain by each principal component for dataset 2.

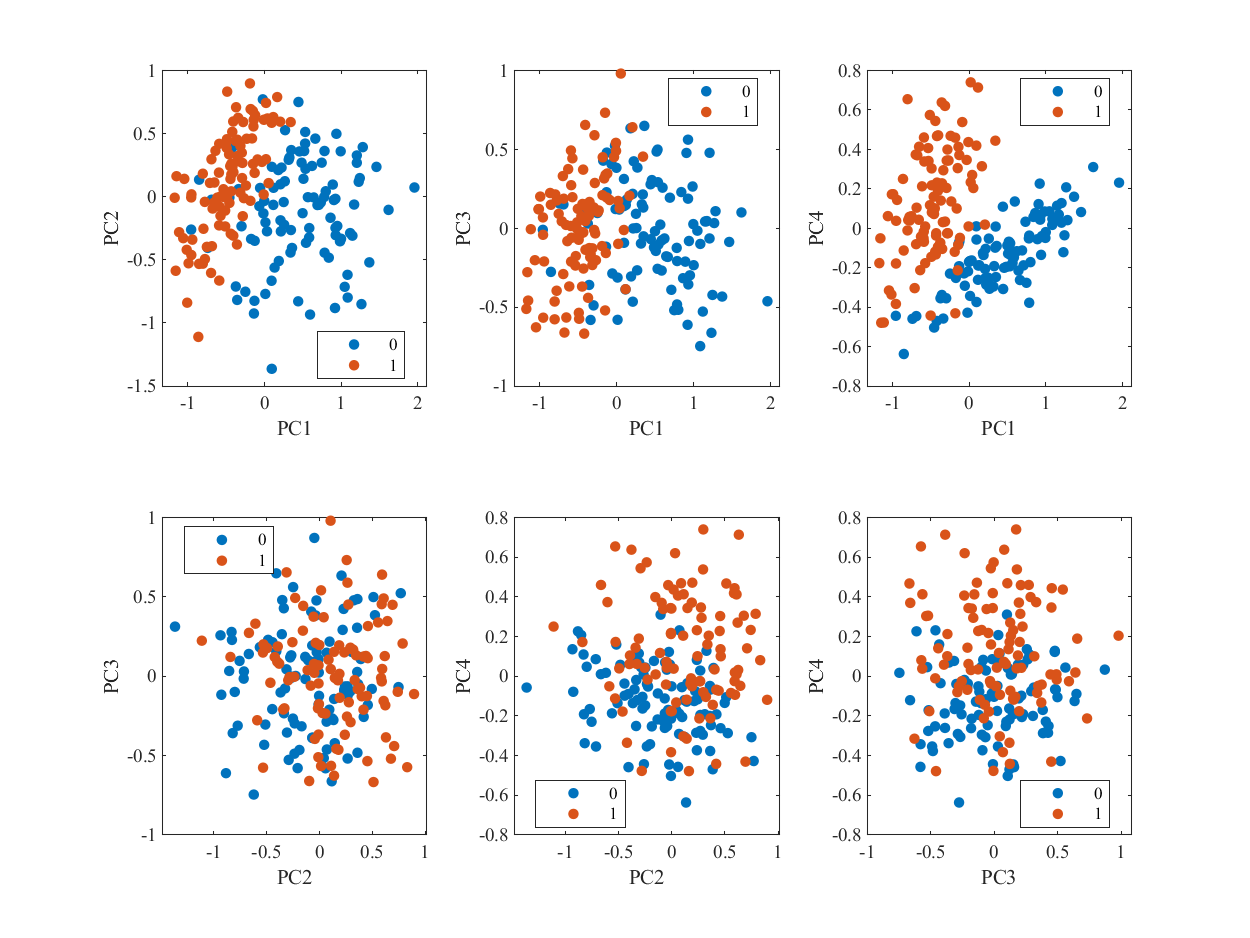
****

Figure 5. Pairwise principal component scatter plot for the first four principal components.

Table 2. Performance metrics for classification models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | Accuracy for WOO dataset | Accuracy for FFR | Accuracy for MRMR |
| PCA Classifier | 0.995 | 0.995 | 0.945 | 0.935 |
| PLS Classifier | 0.895 | 0.94 | 0.935 | 0.955 |

PCA= principal component analysis; PLSR = partial least square; WOO = without outlier; FFS = F-tests feature ranking; MRMR = minimum redundancy maximum relevance.

**Conclusions**

**Appendix**

%% Main Function: Run This to produce assignment results

function[] = assignment3()

%% Set 1

% load the dataset

load Assignment\_data.mat

% template for dataset 1 results

result\_table = array2table(zeros(5,13),'VariableNames',{'Variable #', ...

'R2 test PCR', 'R2 test PCR WOO', 'R2 test PLSR', 'R2 test PLSR WOO', 'R2 test PLSR Var Sel 1', 'R2 test PLSR Var Sel 2',...

'RMSE test PCR', 'RMSE test PCR WOO', 'RMSE test PLSR','RMSE test PLSR WOO','RMSE test PLSR Var Sel 1','RMSE test PLSR Var Sel 2'});

% assing response numbers

result\_table.("Variable #") = [1;2;3;4;5];

%Assign training and test data

xTrain = Set1.cal\_X;

yTrain = Set1.cal\_Y;

xTest = Set1.pred\_X;

yTest = Set1.pred\_Y;

%Are the predictors correlated?

h = heatmap(corrcoef(xTrain));

h.GridVisible = 'off';

%Some degree of correlation betwen adjacent and ends of the spectra

%boxplots for responses

f=figure('Renderer', 'painters', 'Position', [10 10 400 400]);

boxplot(yTrain)

xlabel('Response')

ylabel('Intensity')

fontname(f,"Times New Roman");

saveas(f,'fig1.png')

%find outliers in for each Y and remove outlier datapoints using the

%quartile method

outidxY = isoutlier(yTrain,"quartiles",1);

xTrainWOO= xTrain(~max(outidxY,[],2),:);

yTrainWOO= yTrain(~max(outidxY,[],2),:);

%calculate PCs for X

[xCoeff,xScore,xLatent,~,xExplained] = pca(xTrain,"Centered",true);

%plot the variance explained by each PC

f=figure('Renderer', 'painters', 'Position', [10 10 500 500]);

bar(xExplained);

xlabel('PC #')

ylabel('% Variance explained')

fontname(f,"Times New Roman");

saveas(f,'fig1.5.png')

rng("default")

%plot the correlation plots for first 5 PCs and each 5 responses

f=figure('Renderer', 'painters', 'Position', [10 10 1000 1000]);

tiledlayout(5,5);

for i=1:5

for j=1:5

nexttile;

s = scatter(xScore(:,i), yTrain(:,j), 6, "filled", "MarkerFaceAlpha", 0.5);

m = fitlm(xScore(:,i), yTrain(:,j));

hold on

plot(xScore(:,i), m.Fitted, "LineWidth", 2)

text(min(xScore(:,i)).\*0.8,max(yTrain(:,j))\*0.8,strcat('R^2 = ',string(round(m.Rsquared.Adjusted,2))));

if j==1

ylabel(strcat('PC ', string(i)));

end

if i == 1

title(append('Response ', string(j)),'FontWeight','normal');

end

end

end

fontname(f,"Times New Roman");

saveas(f,'fig2.png')

%for each of the responses tune and test regressions for PCR and PLSR

for i=1:5

%Tune and test PCR

[~, ~, ~, result\_table.("RMSE test PCR")(i), ~, result\_table.("R2 test PCR")(i)] = pca\_regression(xTrain,xTest,yTrain(:,i),yTest(:,i));

%Tune and test PCR for dataset without outliers

[~, ~, ~, result\_table.("RMSE test PCR WOO")(i), ~, result\_table.("R2 test PCR WOO")(i)] = pca\_regression(xTrainWOO,xTest,yTrainWOO(:,i),yTest(:,i));

%Tune and test PLSR

[~, ~, ~, result\_table.("RMSE test PLSR")(i), ~, result\_table.("R2 test PLSR")(i)] = pls\_regression(xTrain,xTest,yTrain(:,i),yTest(:,i));

%Tune and test PLSR for dataset without outliers

[~, ~, ~, result\_table.("RMSE test PLSR WOO")(i), ~, result\_table.("R2 test PLSR WOO")(i)] = pls\_regression(xTrainWOO,xTest,yTrainWOO(:,i),yTest(:,i));

%Tune and test PLSR for 10 variables selected based on FFS

[~, ~, ~, result\_table.("RMSE test PLSR Var Sel 1")(i), ~, result\_table.("R2 test PLSR Var Sel 1")(i)] = pls\_regression\_fsrtest(xTrain,xTest,yTrain(:,i),yTest(:,i));

%Tune and test PLSR for 10 variables selected based on MRMR

[~, ~, ~, result\_table.("RMSE test PLSR Var Sel 2")(i), ~, result\_table.("R2 test PLSR Var Sel 2")(i)] = pls\_regression\_fsrmrmr(xTrain,xTest,yTrain(:,i),yTest(:,i));

end

%save the results table

writetable(result\_table,'res1.csv');

%% Set 2

res = zeros(2,4);

%Assign training and test sets

xTrain = Set2.cal\_X;

yTrain = Set2.cal\_Y;

xTest = Set2.pred\_X;

yTest = Set2.pred\_Y;

%Convert the responses to binary

yTest = yTest - 1;

yTrain = yTrain - 1;

%Are the predictors correlated?

figure;

h = heatmap(corrcoef(xTrain));

h.GridVisible = 'off';

%Some degree of correlation betwen adjacent and ends of the spectra

%find outliers in for each X and remove outlier datapoints

outidxX = isoutlier(xTrain,"quartiles",1);

xTrainWOO= xTrain(~max(outidxX,[],2),:);

yTrainWOO= yTrain(~max(outidxX,[],2),:);

%PCA of the xCal

[xCoeff,xScore,xLatent,~,xExplained] = pca(xTrain,"Centered",true);

%plot the variance explained by each PC

f=figure('Renderer', 'painters', 'Position', [10 10 500 500]);

bar(xExplained);

xlabel('PC #')

ylabel('% Variance explained')

fontname(f,"Times New Roman");

saveas(f,'fig3.png')

rng("default")

%plot the datapoints with their clases for first four pairwise PCs

f=figure('Renderer', 'painters', 'Position', [10 10 800 600]);

tiledlayout(2,3);

nexttile;

gscatter(xScore(:,1),xScore(:,2),yTrain);

xlabel(strcat('PC1'));

ylabel(strcat('PC2'))

nexttile;

gscatter(xScore(:,1),xScore(:,3),yTrain);

xlabel(strcat('PC1'));

ylabel(strcat('PC3'))

nexttile;

gscatter(xScore(:,1),xScore(:,4),yTrain);

xlabel(strcat('PC1'));

ylabel(strcat('PC4'))

nexttile;

gscatter(xScore(:,2),xScore(:,3),yTrain);

xlabel(strcat('PC2'));

ylabel(strcat('PC3'))

nexttile;

gscatter(xScore(:,2),xScore(:,4),yTrain);

xlabel(strcat('PC2'));

ylabel(strcat('PC4'));

nexttile;

gscatter(xScore(:,3),xScore(:,4),yTrain);

xlabel(strcat('PC3'));

ylabel(strcat('PC4'));

fontname(f,"Times New Roman");

saveas(f,'fig4.png')

%train and test PCA classifier

[~,~,~,~,res(1,1)] = pca\_classifier(xTrain,xTest,yTrain,yTest);

%train and test PCA classifier for data without outlier

[~,~,~,~,res(1,2)] = pca\_classifier(xTrainWOO,xTest,yTrainWOO,yTest);

%train and test PCA classifier for 10 selected variables based on FFS

[~,~,~,~,res(1,3)] = pca\_classifier\_fsrtest(xTrain,xTest,yTrain,yTest);

%train and test PCA classifier for 10 selected variables based on MRMR

[~,~,~,~,res(1,4)] = pca\_classifier\_fsrmrmr(xTrain,xTest,yTrain,yTest);

%train and test PLS classifier

[~,~,~,~,res(2,1)] = pls\_classifier(xTrain,xTest,yTrain,yTest);

%train and test PLS classifier for data without outlier

[~,~,~,~,res(2,2)] = pls\_classifier(xTrainWOO,xTest,yTrainWOO,yTest);

%train and test PLS classifier for 10 selected variables based on FFS

[~,~,~,~,res(2,3)] = pls\_classifier\_fsrtest(xTrain,xTest,yTrain,yTest);

%train and test PLS classifier for 10 selected variables based on MRMR

[~,~,~,~,res(2,4)] = pls\_classifier\_fsrmrmr(xTrain,xTest,yTrain,yTest);

res = array2table(res);

res.Properties.VariableNames = {'Tuned Model','Tuned Model Without Outliers', 'Tuned Model on 10 Variables Selected by F-tests', 'Tuned Model on 10 Variables Selected by MRMR'};

res.Properties.RowNames = {'PCA Classifier', 'PLS Classifier'};

%save the results

writetable(res,'res2.csv','WriteRowNames',true);

end

%% Auxilary functions to the main function

%this function calculates model metrics for predicted and actual ys of a

%regression

function [RMSE,R2,bias] = model\_metrics(yPred,yAct)

RMSE = rmse(yPred,yAct);

SSE = sum((yPred-yAct).^2);

bias = SSE/size(yPred,1);

SSR = sum((yAct-mean(yAct)).^2);

R2 = 1 - SSE/SSR;

end

%this function calculates model metrics for predicted and actual ys of a

%classification

function [acc,sens] = model\_metrics\_classifier(yPred,yAct)

perf = classperf(yAct,yPred);

acc = perf.CorrectRate;

sens = perf.Sensitivity;

end

%this function trains a PCA classifier and produces model metrics

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pca\_classifier(xTrain,xTest,yTrain,yTest)

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

%tune for the best number of components with 5 folds

ncomp = pca\_tuner\_class(xTrain,yTrain,5);

%get the PCAs

[pcaTrain,pcaScoresTrain] = pca(xTrainScaled);

%regress for PCAs and calculate beta

betaPCR = regress(yTrain-mean(yTrain), pcaScoresTrain(:,1:ncomp));

betaPCR = pcaTrain(:,1:ncomp)\*betaPCR;

betaPCR = [mean(yTrain) - mean(xTrainScaled)\*betaPCR; betaPCR];

%predict for training and test set based on calculated beta

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*betaPCR,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*betaPCR,0);

%convert the results to binary

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

%produce the metrics for the training and test set

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains a PCA classifier and produces model metrics based on

%10 variables selected by MRMR

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pca\_classifier\_fsrmrmr(xTrain,xTest,yTrain,yTest)

%calculate mrmrs and select the first 10 most important variables

[idx,~] = fsrmrmr(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

%tune for the best number of components with 5 folds

ncomp = pca\_tuner\_class(xTrain,yTrain,5);

%get the PCAs

[pcaTrain,pcaScoresTrain] = pca(xTrainScaled);

%calculate beta

betaPCR = regress(yTrain-mean(yTrain), pcaScoresTrain(:,1:ncomp));

betaPCR = pcaTrain(:,1:ncomp)\*betaPCR;

betaPCR = [mean(yTrain) - mean(xTrainScaled)\*betaPCR; betaPCR];

%predict for the training and test set based on calculated beta

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*betaPCR,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*betaPCR,0);

%convert the predictions to binary

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

%produce model metrics

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains a PCA classifier and produces model metrics based on

%10 variables selected by FFS

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pca\_classifier\_fsrtest(xTrain,xTest,yTrain,yTest)

%calculate FFS and select the first 10 most important variables

[idx,~] = fsrftest(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

%tune for the best number of components with 5 folds

ncomp = pca\_tuner\_class(xTrain,yTrain,5);

%get the PCAs

[pcaTrain,pcaScoresTrain] = pca(xTrainScaled);

%calculate beta

betaPCR = regress(yTrain-mean(yTrain), pcaScoresTrain(:,1:ncomp));

betaPCR = pcaTrain(:,1:ncomp)\*betaPCR;

betaPCR = [mean(yTrain) - mean(xTrainScaled)\*betaPCR; betaPCR];

%predict for the training and test set based on calculated beta

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*betaPCR,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*betaPCR,0);

%convert the predictions to binary

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

%produce model metrics

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains a PCA regression, similar to PCA classifier

function [predYtest, predYtrain, rmseVal, rmseTest, r2Val, r2Test, biasVal, biasTest,ncomp] = pca\_regression(xTrain,xTest,yTrain,yTest)

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pca\_tuner(xTrain,yTrain,5);

[pcaTrain,pcaScoresTrain] = pca(xTrainScaled);

betaPCR = regress(yTrain-mean(yTrain), pcaScoresTrain(:,1:ncomp));

betaPCR = pcaTrain(:,1:ncomp)\*betaPCR;

betaPCR = [mean(yTrain) - mean(xTrainScaled)\*betaPCR; betaPCR];

predYtrain = [ones(size(xTrainScaled,1),1) xTrainScaled]\*betaPCR;

predYtest = [ones(size(xTestScaled,1),1) xTestScaled]\*betaPCR;

[rmseVal,r2Val,biasVal] = model\_metrics(predYtrain,yTrain);

[rmseTest,r2Test,biasTest] = model\_metrics(predYtest,yTest);

end

%this function tunes a PCA regression for number of components based on k

%folds

function ncomp = pca\_tuner(xTrain,yTrain,k)

%assume maximum number of components to test

ncompTest = min((size(xTrain,1))/2,(size(xTrain,2))/2);

rng(0)

%create a cross-validation partitions

c = cvpartition(size(xTrain,1),"KFold",k);

%template for results

res\_comp = zeros(ncompTest,1);

%iterate through each number of components and train and test for all

%folds and calculate RMSE

for i=1:ncompTest

predFolds = zeros(0,2);

for j=1:k

xTrainK = xTrain(training(c,k),:);

yTrainK = yTrain(training(c,k));

xTestK = xTrain(test(c,k),:);

yTestK = yTrain(test(c,k));

[xTrainKScaled, xTestKScaled] = scale\_train\_test(xTrainK,xTestK);

[pcaTrainK,pcaScoresTrainK] = pca(xTrainKScaled);

betaPCR = regress(yTrainK-mean(yTrainK), pcaScoresTrainK(:,1:i));

betaPCR = pcaTrainK(:,1:i)\*betaPCR;

betaPCR = [mean(yTrainK) - mean(xTrainKScaled)\*betaPCR; betaPCR];

predYtestK = [ones(size(xTestKScaled,1),1) xTestKScaled]\*betaPCR;

predFolds = [predFolds;[predYtestK,yTestK]];

end

res\_comp(i) = model\_metrics(predFolds(:,1),predFolds(:,2));

end

%find the best number of components

[~,ncomp\_cand] = min(res\_comp);

ncomp\_cand\_new = ncomp\_cand;

while true

if ncomp\_cand\_new == 1

break;

%to make sure unnecessary large number of components is not

%selected, the minimum RMSE is compared to the n-1 components RMSE

%and if they are only 1 percent different, n-1 is selected as the

%new number of components. This step is repeated until the

%difference is greater than 1 percent.

elseif ((res\_comp(ncomp\_cand\_new)- res\_comp(ncomp\_cand\_new-1))/res\_comp(ncomp\_cand))<0.01

ncomp\_cand\_new = ncomp\_cand\_new - 1;

else

break;

end

end

ncomp = ncomp\_cand\_new;

end

%this function tunes a PCA classifier for number of components based on k

%folds. Similar to previous tuner.

function ncomp = pca\_tuner\_class(xTrain,yTrain,k)

%determine the number of components to test for

ncompTest = min((size(xTrain,1))/2,(size(xTrain,2))/2);

rng(0)

c = cvpartition(size(xTrain,1),"KFold",k);

res\_comp = zeros(ncompTest,1);

for i=1:ncompTest

predFolds = zeros(0,2);

for j=1:k

xTrainK = xTrain(training(c,k),:);

yTrainK = yTrain(training(c,k));

xTestK = xTrain(test(c,k),:);

yTestK = yTrain(test(c,k));

[xTrainKScaled, xTestKScaled] = scale\_train\_test(xTrainK,xTestK);

[pcaTrainK,pcaScoresTrainK] = pca(xTrainKScaled);

betaPCR = regress(yTrainK-mean(yTrainK), pcaScoresTrainK(:,1:i));

betaPCR = pcaTrainK(:,1:i)\*betaPCR;

betaPCR = [mean(yTrainK) - mean(xTrainKScaled)\*betaPCR; betaPCR];

predYtestK = round([ones(size(xTestKScaled,1),1) xTestKScaled]\*betaPCR,0);

predYtestK(predYtestK<0) = 0;

predYtestK(predYtestK>1) = 1;

predFolds = [predFolds;[predYtestK,yTestK]];

end

res\_comp(i) = model\_metrics\_classifier(predFolds(:,1),predFolds(:,2));

end

[~,ncomp\_cand] = max(res\_comp);

ncomp\_cand\_new = ncomp\_cand;

while true

if ncomp\_cand\_new == 1

break;

elseif ((res\_comp(ncomp\_cand\_new)- res\_comp(ncomp\_cand\_new-1))/res\_comp(ncomp\_cand))<0.01

ncomp\_cand\_new = ncomp\_cand\_new - 1;

else

break;

end

end

ncomp = ncomp\_cand\_new;

end

%this function trains and tests a PLS classifier, similar to PCA classifier

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pls\_classifier(xTrain,xTest,yTrain,yTest)

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner\_class(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*beta,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*beta,0);

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains and tests a PLS classifier with fist 10 variables

%selected based on MRMR

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pls\_classifier\_fsrmrmr(xTrain,xTest,yTrain,yTest)

%select the first 10 varibles

[idx,~] = fsrmrmr(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner\_class(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*beta,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*beta,0);

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains and tests a PLS classifier with fist 10 variables

%selected based on FFS

function [predYtest, predYtrain, accuVal, sensVal, accuTest, sensTest,ncomp] = pls\_classifier\_fsrtest(xTrain,xTest,yTrain,yTest)

%select the first 10 varibles

[idx,~] = fsrftest(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner\_class(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = round([ones(size(xTrainScaled,1),1) xTrainScaled]\*beta,0);

predYtest = round([ones(size(xTestScaled,1),1) xTestScaled]\*beta,0);

predYtrain(predYtrain<0) = 0;

predYtrain(predYtrain>1) = 1;

predYtest(predYtest<0) = 0;

predYtest(predYtest>1) = 1;

[accuVal,sensVal] = model\_metrics\_classifier(predYtrain,yTrain);

[accuTest,sensTest] = model\_metrics\_classifier(predYtest,yTest);

end

%this function trains and tests a PLS regression

function [predYtest, predYtrain, rmseVal, rmseTest, r2Val, r2Test, biasVal, biasTest,ncomp] = pls\_regression(xTrain,xTest,yTrain,yTest)

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = [ones(size(xTrainScaled,1),1) xTrainScaled]\*beta;

predYtest = [ones(size(xTestScaled,1),1) xTestScaled]\*beta;

[rmseVal,r2Val,biasVal] = model\_metrics(predYtrain,yTrain);

[rmseTest,r2Test,biasTest] = model\_metrics(predYtest,yTest);

end

%this function trains and tests a PLS classifier with fist 10 variables

%selected based on MRMR

function [predYtest, predYtrain, rmseVal, rmseTest, r2Val, r2Test, biasVal, biasTest,ncomp] = pls\_regression\_fsrmrmr(xTrain,xTest,yTrain,yTest)

%select the first 10 variables

idx = fsrmrmr(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = [ones(size(xTrainScaled,1),1) xTrainScaled]\*beta;

predYtest = [ones(size(xTestScaled,1),1) xTestScaled]\*beta;

[rmseVal,r2Val,biasVal] = model\_metrics(predYtrain,yTrain);

[rmseTest,r2Test,biasTest] = model\_metrics(predYtest,yTest);

end

%this function trains and tests a PLS classifier with fist 10 variables

%selected based on FFS

function [predYtest, predYtrain, rmseVal, rmseTest, r2Val, r2Test, biasVal, biasTest,ncomp] = pls\_regression\_fsrtest(xTrain,xTest,yTrain,yTest)

%select the first 10 variables

idx = fsrftest(xTrain,yTrain);

xTrain = xTrain(:,idx(1:10));

xTest = xTest(:,idx(1:10));

%scale the training and test set

[xTrainScaled, xTestScaled] = scale\_train\_test(xTrain,xTest);

ncomp = pls\_tuner(xTrain,yTrain,5);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainScaled,yTrain,ncomp);

predYtrain = [ones(size(xTrainScaled,1),1) xTrainScaled]\*beta;

predYtest = [ones(size(xTestScaled,1),1) xTestScaled]\*beta;

[rmseVal,r2Val,biasVal] = model\_metrics(predYtrain,yTrain);

[rmseTest,r2Test,biasTest] = model\_metrics(predYtest,yTest);

end

%this function tunes a PLS regression for number of components based on k

%folds

function ncomp = pls\_tuner(xTrain,yTrain,k)

ncompTest = min((size(xTrain,1))/2,(size(xTrain,2))/2);

rng(0)

c = cvpartition(size(xTrain,1),"KFold",k);

res\_comp = zeros(ncompTest,1);

for i=1:ncompTest

predFolds = zeros(0,2);

for j=1:k

xTrainK = xTrain(training(c,k),:);

yTrainK = yTrain(training(c,k));

xTestK = xTrain(test(c,k),:);

yTestK = yTrain(test(c,k));

[xTrainKScaled, xTestKScaled] = scale\_train\_test(xTrainK,xTestK);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainKScaled,yTrainK,i);

predYtestK = [ones(size(xTestKScaled,1),1) xTestKScaled]\*beta;

predFolds = [predFolds;[predYtestK,yTestK]];

end

res\_comp(i) = model\_metrics(predFolds(:,1),predFolds(:,2));

end

[~,ncomp\_cand] = min(res\_comp);

ncomp\_cand\_new = ncomp\_cand;

while true

if ncomp\_cand\_new == 1

break;

elseif (abs((res\_comp(ncomp\_cand\_new)- res\_comp(ncomp\_cand\_new-1))/res\_comp(ncomp\_cand)))<0.001

ncomp\_cand\_new = ncomp\_cand\_new - 1;

else

break;

end

end

ncomp = ncomp\_cand\_new;

end

%this function tunes a PLS classifier for number of components based on k

%folds

function ncomp = pls\_tuner\_class(xTrain,yTrain,k)

ncompTest = min((size(xTrain,1))/2,(size(xTrain,2))/2);

rng(0)

c = cvpartition(size(xTrain,1),"KFold",k);

res\_comp = zeros(ncompTest,1);

for i=1:ncompTest

predFolds = zeros(0,2);

for j=1:k

xTrainK = xTrain(training(c,k),:);

yTrainK = yTrain(training(c,k));

xTestK = xTrain(test(c,k),:);

yTestK = yTrain(test(c,k));

[xTrainKScaled, xTestKScaled] = scale\_train\_test(xTrainK,xTestK);

[XL,yl,XS,YS,beta,PCTVAR] = plsregress(xTrainKScaled,yTrainK,i);

predYtestK = round([ones(size(xTestKScaled,1),1) xTestKScaled]\*beta,0);

predYtestK(predYtestK<0) = 0;

predYtestK(predYtestK>1) = 1;

predFolds = [predFolds;[predYtestK,yTestK]];

end

res\_comp(i) = model\_metrics\_classifier(predFolds(:,1),predFolds(:,2));

end

[~,ncomp\_cand] = min(res\_comp);

ncomp\_cand\_new = ncomp\_cand;

while true

if ncomp\_cand\_new == 1

break;

elseif (abs((res\_comp(ncomp\_cand\_new)- res\_comp(ncomp\_cand\_new-1))/res\_comp(ncomp\_cand)))<0.001

ncomp\_cand\_new = ncomp\_cand\_new - 1;

else

break;

end

end

ncomp = ncomp\_cand\_new;

end

%This function takes a training set, mean center it and also mean center

%the test set based on the training set mean center parameters

function [xTrainScaled,xTestScaled] = scale\_train\_test(xTrain,xTest)

%scale the training set set

[xTrainScaled,C,S] = normalize(xTrain,1,'center','mean','scale');

xTestScaled = (xTest-C)./S;

end