A Print Out of my Matlab Code

RandomForest_BayesOpt.m

```
tic

clear;

clc;

% importing the data

data = readtable('Breastcancer.csv');

% An analysis of the data indicates that three columns

% (area_mean, area_se, and area_worst) contain substantial outliers.

summary(data);

% Independent and dependent variables separated

% The diagnosis feature will be my dependent variable (Y) because it is reliant on the
other features.

% As a result, the rest are independent features (X) because they are not dependent on
anything.

indepen_X = table2array(data ...

   (:,3:32));

depen_y = table2array(data ...

   (:,2));

% Variables are saved

depen_var = data.Properties.VariableNames;

indepen_var = data(:,3:end);

% defining the classes

% The number of 'M'alignant tumors is 212 (1) and the number of 'B'enign tumors is 357 (0)

M_B_count = tabulate(depen_y);

% use C_V partitioning to separate testing and traing data

C_V = cvpartition(depen_y, ...

   'holdout',0.3);

trainX = indepen_X(training ...

   (C_V,1),:);
```

```
trainy = depen_y(training ...
   (C_V,1));
testX = indepen_X(test ...
   (C_V,1),:);
testy = depen_y(test ...
   (C_V,1),:);
% In response to 3 significant outliers from the summary,
% I normalized the data with mu=0 and standard deviation = 1.
[trainX, mu, stddev] = normalize(trainX);
for i=1:size(testX, 2)
   testX(:,i) = (testX(:,i) ...
       -mu(1,i))/stddev(1,i);
end
% As a result of the highly correlated dataset, the tuning of hyperparameters was done by
Bayesian optimisation.
% We generate one figure using this method (because we optimize three variables):
% Defining the classes shows how the objective function is estimated and observed
% We create three variables with specific names, types, and ranges.
% VTS 'NumVariablesToSample'  lc 'learning cycle'
VTS_a = optimizableVariable('VTS_a',[1, 50], ...
   'Type','integer');
split_a = optimizableVariable('split_a',{'gdi', 'deviance'}, ...
   'Type','categorical');
num_lc_a = optimizableVariable('num_lc_a',[10, 400], ...
   'Type','integer');
n2 = size(trainX, 1);
rng(1);
cv2 = cvpartition(n2, ...
   'Kfold',10);
% The objective function is to return a measure of the tuning loss for hyperparameters.
fun = @(x)kfoldLoss(fitcensemble(trainX, trainy, 'CVPartition', cv2, 'Method', 'Bag'));
```

```
% When trying for 150 observations, set the objective evaluation limit higher than the acquisition function

% and expected improvement

bayesopt_results2 = bayesopt(fun,[VTS_a, split_a, num_lc_a],'Verbose',1, 'MaxObjectiveEvaluations',150);

% I have saved the best combination that gives the minimum error

vts_b_bo = bayesopt_results2.XAtMinObjective.VTS_a;

split_b_bo = bayesopt_results2.XAtMinObjective.split_a;

numlc_b_bo = bayesopt_results2.XAtMinObjective.num_lc_a;

% Minimum error for the set of hyperparameters that are evaluated on the validation set.

min_error = bayesopt_results2.MinObjective;

% Creating an optimised tree and generating results, and refit the RF with this tree template as the base learner

t = templateTree('NumVariablesToSample', vts_b_bo, ...

   'SplitCriterion', char(split_b_bo));

% refit the RF with this tree template as the base learner

rng(1)

bayes_opt_mdl_l = fitcensemble(trainX, trainy, 'Method', 'Bag', 'NumLearningCycles',numlc_b_bo,'learners', t);

bayes_opt_mdl_l_loss = loss(bayes_opt_mdl_l,testX, testy); % Using model predictions to classify training data

bayes_opt_mdl_l_rloss = resubLoss(bayes_opt_mdl_l); % Predictions that have been misclassified above.

% Creating a matrix for Bayesian optimisation results and test model

figure()

bayesopt_Predict = predict(bayes_opt_mdl_l, testX);

Confmat_bayesopt_rf = confusionmat(bayesopt_Predict, testy);

matrix = confusionchart(bayesopt_Predict, testy);

% Accuracy of Random Forest model with bayesopt tuning.

accuracyRF = 100*(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(2,2))./(Confmat_bayesopt_rf(1,1)+ ...

   Confmat_bayesopt_rf(2,2)+Confmat_bayesopt_rf(1,2)+Confmat_bayesopt_rf(2,1));

% Model accuracy with Bayes optimization tuning

precisionRF = Confmat_bayesopt_rf(1,1)./(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(1,2));
```

```
% As a result of recall, we can figure out how many samples have a real chance of being tumor positive.

recallRF = Confmat_bayesopt_rf(1,1)./(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(2,1));

% The accuracy of a test is measured by the F1 score. Additionally,

% True Negatives (TN) and False Negatives (FN) are crucial for our results, so we use the F1 score.

f1ScoresRF = 2*(precisionRF.*recallRF)./(precisionRF+recallRF);
```

KNearestNeighbor_BayesOpt.m

```
tic

clear;

clc;

% importing the data

data = readtable('Breastcancer.csv');

% An analysis of the data indicates that three columns

% (area_mean, area_se, and area_worst) contain substantial outliers.

summary(data);

% Independent and dependent variables separated

% The diagnosis feature will be my dependent variable (Y) because it is reliant on the other features.

% As a result, the rest are independent features (X) because they are not dependent on anything.

depen_Y = table2array(data ...

   (:,2));

indepen_X = table2array(data ...

   (:,3:32));

% Variables are saved

depen_variables = data.Properties.VariableNames;
```

```
indepen_variables = data(:,3:end);

% defining classes

% The number of 'M'alignant tumors is 212 (1) and the number of 'B'enign tumors is 357 (0)

M_B_count = tabulate(depen_Y);

C_V = cvpartition(depen_Y, ...

   'holdout',0.3);

trainX = indepen_X(training ...

   (C_V,1),:);

trainy = depen_Y(training ...

   (C_V,1));

testX = indepen_X(test ...

   (C_V,1),:);

testy = depen_Y(test ...

   (C_V,1),:);

% In response to 3 significant outliers from the summary,

% I normalized the data with mu=0 and standard deviation = 1.

[trainX, mu, stddev] = normalize(trainX);

for i=1:size(testX, 2)

   testX(:,i) = (testX(:,i) ...

       -mu(1,i))/stddev(1,i);

end

% As a result of the highly correlated dataset, the first tuning of hyperparameters was
done by Bayesian optimi.

% Based on this method, we are able to produce the following figures:

% The first one displays the two HP and the corresponding value of each repetition.

% The second graph shows the objective function's estimated and observed values.

rng(1)

Dis = optimizableVariable('Dis', {'minkowski','correlation','hamming',...

  'jaccard','mahalanobis','cityblock','euclidean','cosine','spearman'...

   'seuclidean','chebychev'},'Type','categorical');

k = optimizableVariable('k',[1,50],'Type','integer');

% Partitions and indexes for the second CVpartitioned dataset.
```

```
idxfold = size(trainX,1);

CV = cvpartition(idxfold, 'kfold', 10);

fun = @(x)kfoldLoss(fitcknn(trainX,trainy,'CVPartition',CV,'NumNeighbors', x.k, ...

   'Distance',char(x.Dis), 'NSMethod','exhaustive'));

% When trying for 200 observations, set the objective evaluation limit higher than the
acquisition function..

% ...and expected improvement

bayesopt_results = bayesopt(fun,[k,Dis], 'Verbose',1, 'MaxObjectiveEvaluations', 200);

k_bayesopt = bayesopt_results.XAtMinObjective.k;

Dis2 = bayesopt_results.XAtMinObjective.Dis;

min_error = bayesopt_results.MinObjective;

% An optimised hyperparameter model has been fitted.

bayes_opt_mdl_knn = fitcknn(trainX , trainy, 'NumNeighbors',
k_bayesopt,'Distance',char(Dis2));

% returns predicted class labels based on the trained classification model

[bayes_opt_yPrd_knn, knn_scr_bayesopt,conio] = predict(bayes_opt_mdl_knn,testX);

% classification effective of training data based on model predictions

bayes_opt_knn_loss = loss(bayes_opt_mdl_knn,testX, testy);

% Misclassifications from the predictions above

bayes_opt_knn_rloss = resubLoss(bayes_opt_mdl_knn);

rng(1)

% confusion matrix %

figure()

[Confmat_bayesopt_knn, order] = confusionmat(testy, bayes_opt_yPrd_knn);

matrix_graph = confusionchart( testy, bayes_opt_yPrd_knn);

% The accuracy

Accuracy =
100*(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(2,2))./(Confmat_bayesopt_knn(1,1)+Confm
at_bayesopt_knn(2,2)+Confmat_bayesopt_knn(1,2)+Confmat_bayesopt_knn(2,1));

% Precision.

Precision =
Confmat_bayesopt_knn(1,1)./(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(1,2));

% As a result of recall, we can figure out how many samples have a real chance of being
tumor positive.
```

```
Recall = Confmat_bayesopt_knn(1,1)./(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(2,1));

% The accuracy of a test is measured by the F1 score. Additionally,

% True Negatives (TN) and False Negatives (FN) are crucial for our results, so we use the
F1 score.

f1_Scores = 2*(Precision.*Recall)./(Precision+Recall);
```

RF_feature_selection.m

```
tic

clear;

clc;

% importing the data

data = readtable('Breastcancer.csv');

% An analysis of the data indicates that three columns

% (area_mean, area_se, and area_worst) contain substantial outliers.

summary(data);

% Independent and dependent variables separated

% The diagnosis feature will be my dependent variable (Y) because it is reliant on the
other features.

% As a result, the rest are independent features (X) because they are not dependent on
anything.

depen_Y = table2array(data ...

   (:,2));

indepen_X = table2array(data ...

   (:,3:32));

% Variables are saved

depen_variables = data.Properties.VariableNames;

indepen_variables = data(:,3:end);

% Through FSCMRMR, feature selection can be made

[idx,scores] = fscmrmr(indepen_X ...

   ,depen_Y);
```

```
bar(scores(idx))

xlabel('Predict rank')

ylabel('Predict importance score')

% Among the highest ranking features, 11 are chosen.

idx(1:11)

%Defining the new matrix

new_mat = indepen_X(:,[23 20 2 29 11 28 25 27 4 8 14]);

% defining classes

% The number of 'M'alignant tumors is 212 (1) and the number of 'B'enign tumors is 357 (0)

M_B_count = tabulate(depen_Y);

C_V = cvpartition(depen_Y, ...

   'holdout',0.3);

trainX = new_mat(training ...

   (C_V,1),:);

trainy = depen_Y(training ...

   (C_V,1));

testX = new_mat(test ...

   (C_V,1),:);

testy = depen_Y(test ...

   (C_V,1),:);

% In response to 3 significant outliers from the summary,

% I normalized the data with mu=0 and standard deviation = 1.

[trainX, mu, stddev] = normalize(trainX);

for i=1:size(testX, 2)

   testX(:,i) = (testX(:,i) ...

       -mu(1,i))/stddev(1,i);

end

% As a result of the highly correlated dataset, the tuning of hyperparameters was done by
Bayesian optimisation.

% We generate one figure using this method (because we optimize three variables):

% Defining the classes shows how the objective function is estimated and observed

% We create three variables with specific names, types, and ranges.
```

```
% VTS 'NumVariablesToSample'  lc 'learning cycle'

VTS_a = optimizableVariable('VTS_a',[1, 50], ...

   'Type','integer');

split_a = optimizableVariable('split_a',{'gdi', 'deviance'}, ...

   'Type','categorical');

num_lc_a = optimizableVariable('num_lc_a',[10, 400], ...

   'Type','integer');

n2 = size(trainX, 1);

rng(1);

cv2 = cvpartition(n2, ...

   'Kfold',10);

% The objective function is to return a measure of the tuning loss for hyperparameters.

fun = @(x)kfoldLoss(fitcensemble(trainX, trainy, 'CVPartition', cv2, 'Method', 'Bag'));

% When trying for 150 observations, set the objective evaluation limit higher than the
acquisition function

% and expected improvement

bayesopt_results2 = bayesopt(fun,[VTS_a, split_a, num_lc_a],'Verbose',1,
'MaxObjectiveEvaluations',150);

% I have saved the best combination that gives the minimum error

vts_b_bo = bayesopt_results2.XAtMinObjective.VTS_a;

split_b_bo = bayesopt_results2.XAtMinObjective.split_a;

numlc_b_bo = bayesopt_results2.XAtMinObjective.num_lc_a;

% Minimum error for the set of hyperparameters that are evaluated on the validation set.

min_error = bayesopt_results2.MinObjective;

% Creating an optimised tree and generating results, and refit the RF with this tree
template as the base learner

t = templateTree('NumVariablesToSample', vts_b_bo, ...

   'SplitCriterion', char(split_b_bo));

% refit the RF with this tree template as the base learner

rng(1)

bayes_opt_mdl_l = fitcensemble(trainX, trainy, 'Method', 'Bag',
'NumLearningCycles',numlc_b_bo,'learners', t);

bayes_opt_mdl_l_loss = loss(bayes_opt_mdl_l,testX, testy); % Using model predictions to
classify training data
```

```
bayes_opt_mdl_1_rloss = resubLoss(bayes_opt_mdl_1); % Predictions that have been
misclassified above.

% Creating a matrix for Bayesian optimisation results and test model

figure()

bayesopt_Predict = predict(bayes_opt_mdl_1, testX);

Confmat_bayesopt_rf = confusionmat(bayesopt_Predict, testy);

matrix = confusionchart(bayesopt_Predict, testy);

fs_AccuracyRF =
100*(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(2,2))./(Confmat_bayesopt_rf(1,1)+ ...

   Confmat_bayesopt_rf(2,2)+Confmat_bayesopt_rf(1,2)+Confmat_bayesopt_rf(2,1));

%fs "feature selection"

% Model accuracy with Bayes optimization tuning

fs_precisionRF =
Confmat_bayesopt_rf(1,1)./(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(1,2));

% As a result of recall, we can figure out how many samples have a real chance of being
tumor positive.

fs_recallRF =
Confmat_bayesopt_rf(1,1)./(Confmat_bayesopt_rf(1,1)+Confmat_bayesopt_rf(2,1));

% The accuracy of a test is measured by the F1 score. Additionally,

% True Negatives (TN) and False Negatives (FN) are crucial for our results, so we use the
F1 score.

fs_f1_Scores = 2*(fs_precisionRF.*fs_recallRF)./(fs_precisionRF+fs_recallRF);
```

KNN_feature_selection.m

```
tic

clear;

clc;

% importing the data

data = readtable('Breastcancer.csv');

% An analysis of the data indicates that three columns

% (area_mean, area_se, and area_worst) contain substantial outliers.

summary(data);

% Independent and dependent variables separated
```

```
% The diagnosis feature will be my dependent variable (Y) because it is reliant on the
other features.

% As a result, the rest are independent features (X) because they are not dependent on
anything.

depen_Y = table2array(data ...

   (:,2));

indepen_X = table2array(data ...

   (:,3:32));

% Variables are saved

depen_variables = data.Properties.VariableNames;

indepen_variables = data(:,3:end);

% Through FSCMRMR, feature selection can be made

[idx,scores] = fscmrmr(indepen_X ...

   ,depen_Y);

bar(scores(idx))

xlabel('Predict rank')

ylabel('Predict importance score') % figure 1

% Among the highest ranking features, 11 are chosen.

idx(1:11)

%Define new matrix

new_mat = indepen_X(:,[23 20 2 29 11 28 25 27 4 8 14]);

% defining classes

% The number of 'M'alignant tumors is 212 (1) and the number of 'B'enign tumors is 357 (0)

M_B_count = tabulate(depen_Y);

C_V = cvpartition(depen_Y, ...

   'holdout',0.3);

trainX = new_mat(training ...

   (C_V,1),:);

trainy = depen_Y(training ...

   (C_V,1));

testX = new_mat(test ...

   (C_V,1),:);
```

```
testy = depen_Y(test ...
   (C_V,1),:);
% In response to 3 significant outliers from the summary,
% I normalized the data with mu=0 and standard deviation = 1.
[trainX, mu, stddev] = normalize(trainX);
for i=1:size(testX, 2)
   testX(:,i) = (testX(:,i) ...
       -mu(1,i))/stddev(1,i);
end
% As a result of the highly correlated dataset, the first tuning of hyperparameters was done by Bayesian optimi.
% Based on this method, we are able to produce the following figures:
% The first one displays the two HP and the corresponding value of each repetition.
% The second graph shows the objective function's estimated and observed values.
rng(1)
Dis = optimizableVariable('Dis', {'minkowski','correlation','hamming',...
  'jaccard','mahalanobis','cityblock','euclidean','cosine','spearman'...
   'seuclidean','chebychev'},'Type','categorical');
k = optimizableVariable('k',[1,50],'Type','integer');
% Partitions and indexes for the second CVpartitioned dataset.
idxfold = size(trainX,1);
CV = cvpartition(idxfold, 'kfold', 10);
fun = @(x)kfoldLoss(fitcknn(trainX,trainy,'CVPartition',CV,'NumNeighbors', x.k, ...
   'Distance',char(x.Dis), 'NSMethod','exhaustive'));
% When trying for 200 observations, set the objective evaluation limit higher than the acquisition function..
% ...and expected improvement
bayesopt_results = bayesopt(fun,[k,Dis], 'Verbose',1, 'MaxObjectiveEvaluations', 200);
k_bayesopt = bayesopt_results.XAtMinObjective.k;
Dis2 = bayesopt_results.XAtMinObjective.Dis;
min_error = bayesopt_results.MinObjective;
% An optimised hyperparameter model has been fitted.
```

```
bayes_opt_mdl_knn = fitcknn(trainX , trainy, 'NumNeighbors',
k_bayesopt,'Distance',char(Dis2));

% returns predicted class labels based on the trained classification model

[bayes_opt_yPrd_knn, knn_scr_bayesopt,conio] = predict(bayes_opt_mdl_knn,testX);

% classification effective of training data based on model predictions

bayes_opt_knn_loss = loss(bayes_opt_mdl_knn,testX, testy);

% Misclassifications from the predictions above

bayes_opt_knn_rloss = resubLoss(bayes_opt_mdl_knn);

rng(1)

% confusion matrix %

figure()

[Confmat_bayesopt_knn, order] = confusionmat(testy, bayes_opt_yPrd_knn);

matrix_graph = confusionchart( testy, bayes_opt_yPrd_knn);

% The accuracy

fs_KNNaccuracy =
100*(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(2,2))./(Confmat_bayesopt_knn(1,1)+Confm
at_bayesopt_knn(2,2)+Confmat_bayesopt_knn(1,2)+Confmat_bayesopt_knn(2,1));

% Precision.

fs_KNNprecision =
Confmat_bayesopt_knn(1,1)./(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(1,2));

% As a result of recall, we can figure out how many samples have a real chance of being
tumor positive.

fs_KNNrecall =
Confmat_bayesopt_knn(1,1)./(Confmat_bayesopt_knn(1,1)+Confmat_bayesopt_knn(2,1));

% The accuracy of a test is measured by the F1 score. Additionally,

% True Negatives (TN) and False Negatives (FN) are crucial for our results, so we use the
F1 score.

fs_KNNf1_Scores = 2*(fs_KNNprecision.*fs_KNNrecall)./(fs_KNNprecision+fs_KNNrecall);
```

REFERENCE

[1] Rank features for classification using minimum redundancy maximum relevance (MRMR) algorithm ‘fscmmr’ (Mathworks): https://uk.mathworks.com/help/stats/fscmrmr.html#description

[2] How to implement Bayesian optimization to tune the hyperparameters of a random forest of regression trees using quantile error (Mathworks): https://uk.mathworks.com/help/stats/tune-random-forest-using-quantile-error-and-bayesian-optimization.html?s_tid=srchtitle

[3] Variable description for bayesopt or other optimizers (Mathworks): https://uk.mathworks.com/help/stats/optimizablevariable.html

[4] Select optimal machine learning hyperparameters using Bayesian optimization (Mathworks): https://uk.mathworks.com/help/stats/bayesopt.html l

[5] Fit ensemble of learners for classification (Mathworks): https://uk.mathworks.com/help/stats/fitcensemble.html

[6] Implementing Cross-validation for ML model (Mathworks): https://uk.mathworks.com/help/stats/classificationsvm.crossval.html

[7] Implementing *k*-nearest neighbour classification (Mathworks): https://uk.mathworks.com/help/stats/classificationknn.html

[8] How to implement a Partition data for cross-validation (Mathworks): https://uk.mathworks.com/help/stats/cvpartition.html

[9] How to Create confusion matrix chart for classification problem (Mathworks): https://uk.mathworks.com/help/stats/confusionchart.html

[10] Ensemble of bagged decision trees (Mathworks): https://uk.mathworks.com/help/stats/treebagger.html

[11] Sanyour Rawan, Abdullah (2019). Real time data analysis and visualisation for breast cancer.

[12] Markris, Radi (2020) KNN classification for the breast cancer wisconsin dataset.

[13] Riyanarto sarno (2018). Analysis of classification algorithm for breast cancer wisconsin.

[14] Sasmitha Kularathne (2020) Prediction of breast cancer using KNN algorithm