Contents

Apply Logistic Regression on Recidivism Data %%

```
clear all;
clc;
close all;
```

Import train and test data. %

```
train_data = readtable('Recidivismtrainset.csv');
test_data = readtable('Recidivismtestset.csv');
```

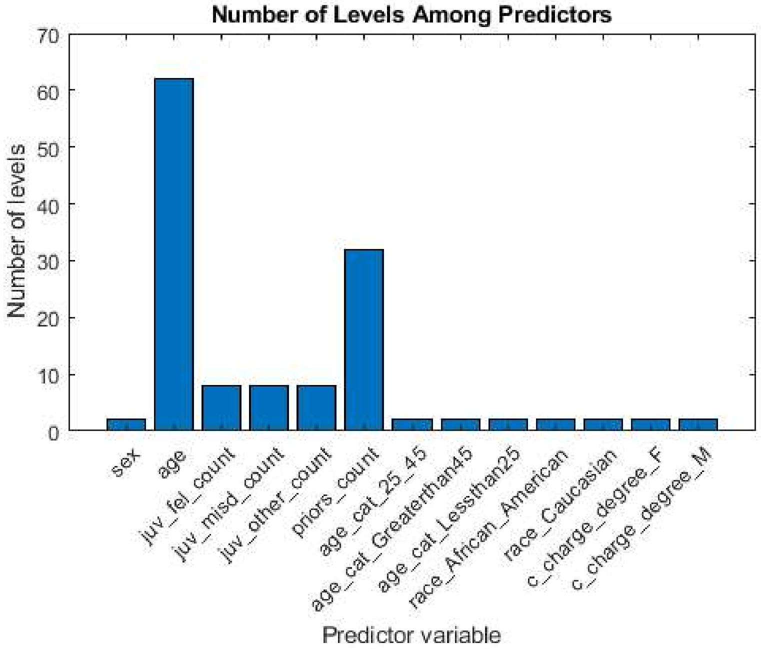
Split Predictor Variables and Response Variable in train % and test data. %

```
x_train = train_data(:,1:end-1);
y_train = train_data(:,end);
x_test = test_data(:,1:end-1);
y_test = test_data(:,end);
```

Create a function the converts all the features in to categorical data % type, determines all unique categories and conts them. %

```
countLevels = @(x)numel(categories(categorical(x)));
numLevels = varfun(countLevels,x_train,'OutputFormat','uniform');

% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

```
%Compare then number of categories among all features. %

figure
bar(numLevels)
title('Number of Levels Among Predictors')
xlabel('Predictor variable')
ylabel('Number of levels')
h = gca;
h.XTickLabel = x_train.Properties.VariableNames;
h.XTickLabelRotation = 45;
h.TickLabelInterpreter = 'none';

% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

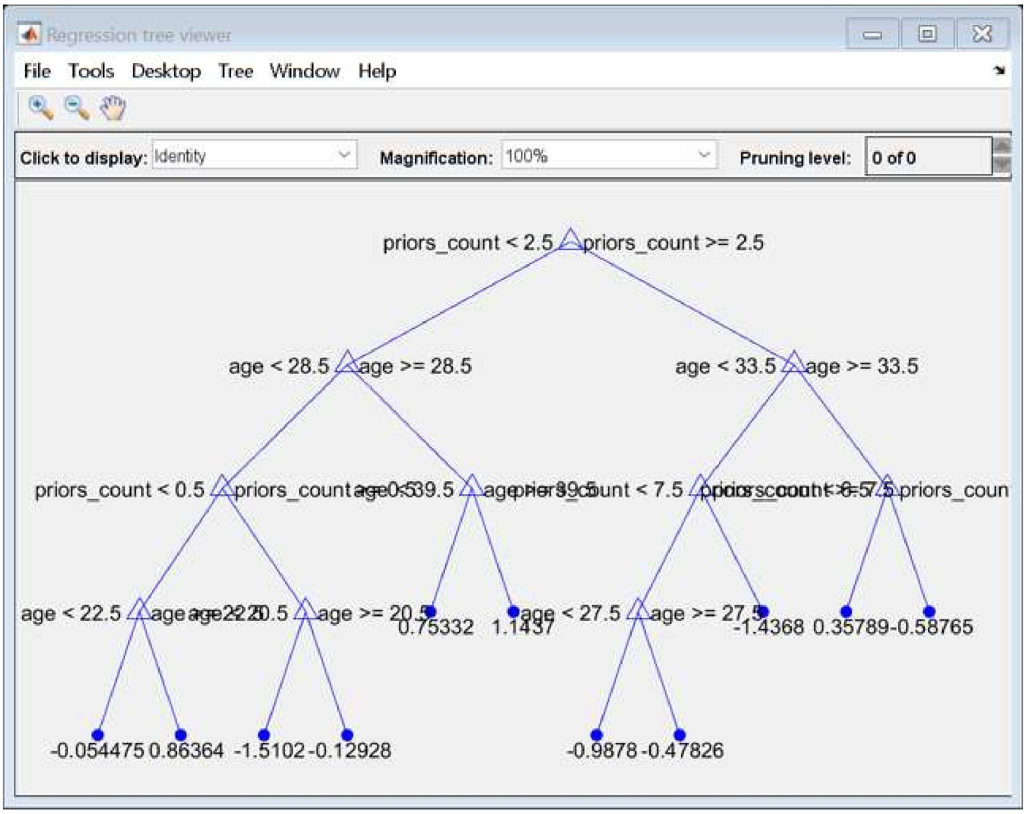


Fit train data into an ensemble algorithm, % fitcensemble was chosen over Treebagger because fo the advantage of % Hyperparameter optimization offered by it. %

```
rng(1);
Mdl1 = fitcensemble(train_data,'two_year_recid');

% This is the baseline model. %
% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

Plot the baseline model. %

```
view(Mdl1.Trained{1}.CompactRegressionLearner,"Mode","graph");
```



Predict Response for train data using the baseline Model. %

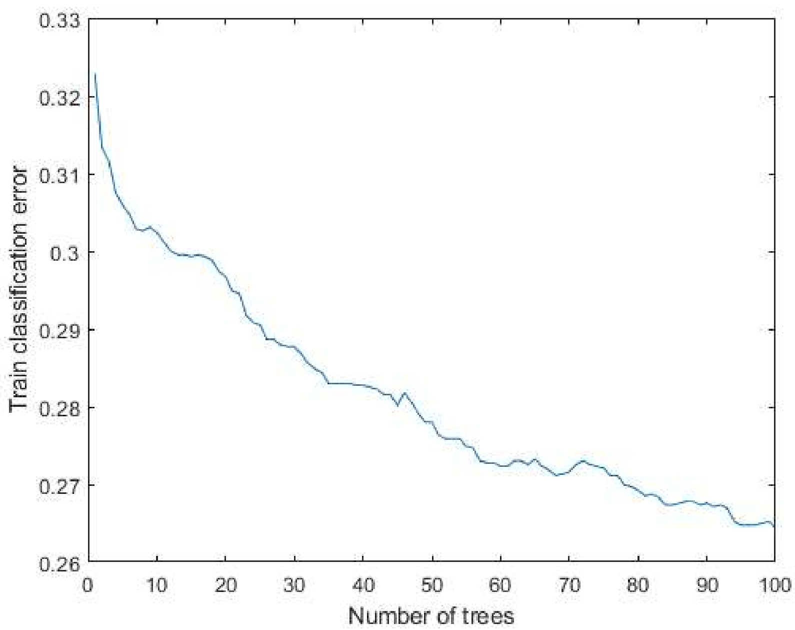
```
yfittrainm1 = predict(Mdl1,x_train);

% It is observed from workspace that the predicted values are on the form %
% 0 and 1. So, they can be directly compared with the original values of %
% the response variable. %

% Calculate error and accuracy of the Model for train data using the %
% comparision Vector. %

vtrainm1 = (yfittrainm1 == y_train.two_year_recid);
trainErrorm1 = 1- sum(vtrainm1)/size(vtrainm1,1);
trainaccuracym1 = sum(vtrainm1)/size(vtrainm1,1);

% fitcensemble has an option resubstitution loss, which compares the %
% predicted values with the original values, just as above technique. %

resubfitm1 = resubPredict(Mdl1);
resublossm1 = resubLoss(Mdl1);
```

Although from workspace it is seen that the resubstitution loss value % is same as the error calculated by earlier technique, it may be better % to compare the the resubstitution prediction as well. %

```
fitcheckm1 = (yfittrainm1 == resubfitm1);
match1 = sum(fitcheckm1)/numel(fitcheckm1)*100;

% After executing this section of code, it can be seen that there is a %
% complete match between resubstitution predict and the general predict. %
% So, further in the script resubstition fuction will be used for %
% evaluating the loss or error of a model on train set. %
```

Plot misclassification of the baseline model as a function of the % number of trained trees in the ensemble. %

```
figure
plot(loss(Mdl1,x_train,y_train,'mode','cumulative'))
xlabel('Number of trees')
ylabel('Train classification error')
```



Check and assign a variable to the method through which the baseline % model was trained. %

```
Model1_Method = Mdl1.Method

% After executing this section of code mutiple times, it was observed that
% for most of runs of this script, LogitBoost or AdaBoostM1 methods were %
% chosen most of the time by the sotware to train the model. %
```

```
Model1_Method =
```

```
'LogitBoost'
```

Bagging reduces variance when compared to boosting. So, the Bag method % will be used to trained the model. %

```
% Fit train data into second ensemble algorithm with Bag method. %
% First iteration of improvisation over the baseline model. %

rng(1);
tic
Mdl2 =fitcensemble(train_data,'two_year_recid','Method','Bag');
toc
```

```
Elapsed time is 0.954030 seconds.
```

Calculate error and accuracy of the Model for train data using the % function for resubstitution loss. %

```
trainErrorm2 = resubLoss(Mdl2);
trainaccuracym2 = 1 -trainErrorm2;
```

From workspace it can be seen that the accuracy of the second model has % improved with Bag Method. Next paramater to try to improve the model % further is the number of learning cycles. % Check the number of learning cycles in which the second model was % trained. %

```
Mdl2.ModelParameters.NLearn
```

```
ans =

   100
```

Fit train data into third ensemble algorithm for different values of % learining cycles. % Second iteration to improve the model. %

```
% Putting different values of learning cycles into a vector. %
numNL = [50 200 500 1000 1500 2000];

% Calculate training accuracy for all the different values of learning %
% cycles by using a for loop. %

% Create a zero vector that will be filled with training accuracies of all
% the runs of the loop, after the execution of the loop. %
trainaccuracym3 = zeros(1,6);

% Create a for loop for the said purpose. %

for i=1:length(numNL);
    rng(1);
    Mdl3 = fitcensemble(train_data,'two_year_recid','Method','Bag','NumLearningCycles', i);
    trainaccuracym3(i) = 1 - resubLoss(Mdl3);
end

trainaccuracym3
```

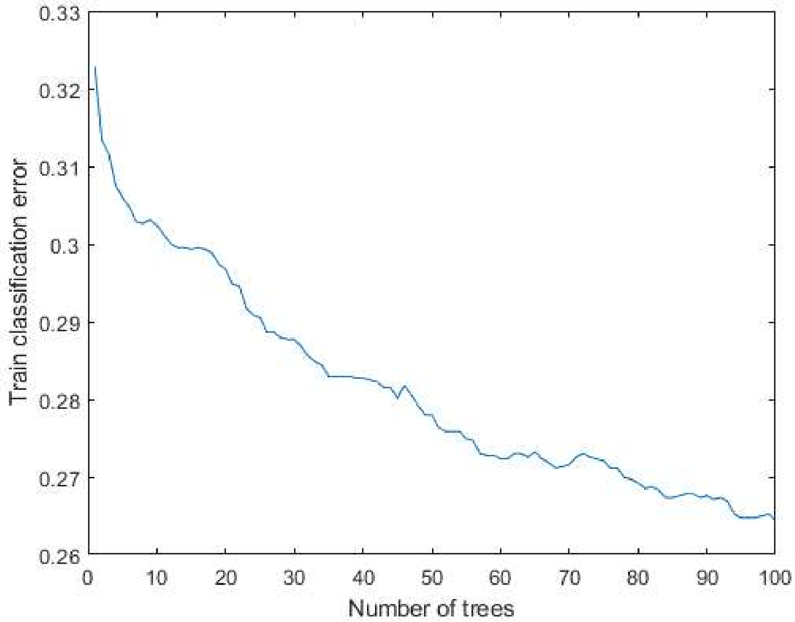
```
trainaccuracym3 =

    0.7130    0.7336    0.7412    0.7469    0.7523    0.7580
```

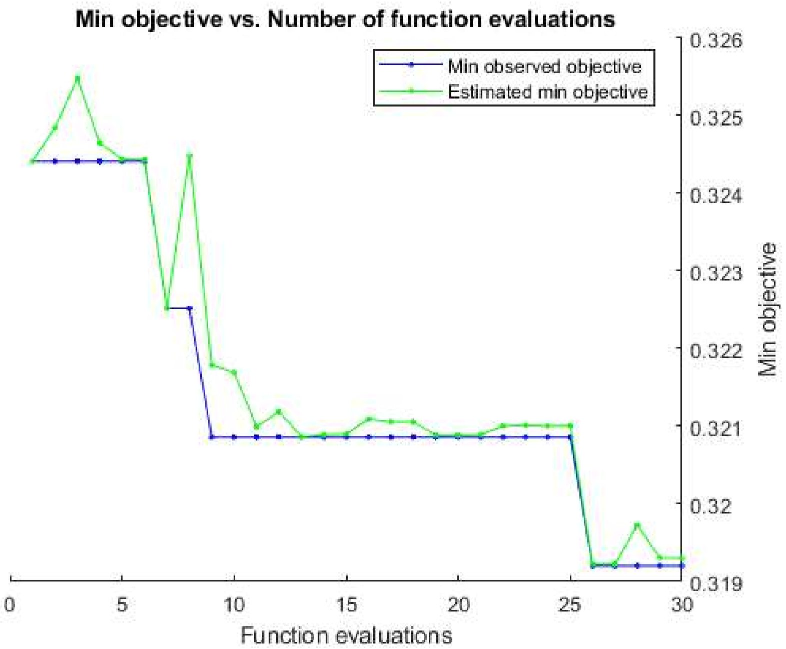
From workspace it can be seen that the accuracies of the third model % for different values of number of learning cycles, were not better than % the accuracy achieved by the second model.%

```
% Third iteration to improve the model. %
% Fit train data into fourth ensemble algorithm, in which the %
% hyperparameters are optimized with the inbuilt argument available for %
% fitcensemble. %

rng(1);
Mdl4 = fitcensemble(train_data,'two_year_recid','Method','Bag','OptimizeHyperparameters','auto');
```

```
% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

```
|===================================================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Method | NumLearningC-|    LearnRate | MinLeafSize |
|      | result |             | runtime     | (observed)  | (estim.)    |              | ycles        |              |             |
|===================================================================================================================================|
|    1 | Best   |     0.32441 |      9.1618 |     0.32441 |     0.32441 |   AdaBoostM1 |          407 |      0.64945 |           1 |
|    2 | Accept |     0.32607 |     0.74973 |     0.32441 |     0.32484 |  GentleBoost |           38 |    0.0091503 |          69 |
|    3 | Accept |     0.32607 |      8.1337 |     0.32441 |     0.32548 |   AdaBoostM1 |          416 |    0.0024906 |          17 |
|    4 | Accept |     0.36206 |     0.92587 |     0.32441 |     0.32465 |     RUSBoost |           20 |     0.012796 |         791 |
|    5 | Accept |      0.3256 |      1.3223 |     0.32441 |     0.32444 |  GentleBoost |           51 |     0.041137 |          77 |
|    6 | Accept |     0.35804 |     0.80205 |     0.32441 |     0.32444 |   LogitBoost |           24 |     0.070176 |           1 |
|    7 | Best   |     0.32252 |      1.4798 |     0.32252 |     0.32252 |          Bag |           25 |            - |          45 |
|    8 | Accept |     0.32631 |       1.797 |     0.32252 |     0.32449 |          Bag |           25 |            - |          41 |
|    9 | Best   |     0.32086 |      25.108 |     0.32086 |     0.32179 |          Bag |          498 |            - |          20 |
|   10 | Accept |     0.36491 |     0.40945 |     0.32086 |     0.32169 |   AdaBoostM1 |           10 |     0.021216 |        1603 |
|   11 | Accept |     0.32607 |      1.0984 |     0.32086 |     0.32099 |  GentleBoost |           28 |    0.0022414 |        1868 |
|   12 | Accept |     0.33128 |      4.2945 |     0.32086 |     0.32119 |          Bag |           98 |            - |         335 |
|   13 | Accept |     0.47052 |     0.56663 |     0.32086 |     0.32087 |          Bag |           10 |            - |        1908 |
|   14 | Accept |     0.33625 |     0.58764 |     0.32086 |     0.32089 |          Bag |           10 |            - |         122 |
|   15 | Accept |      0.3327 |     0.42594 |     0.32086 |      0.3209 |   AdaBoostM1 |           10 |     0.026316 |           4 |
|   16 | Accept |     0.32394 |     0.46095 |     0.32086 |     0.32109 |   AdaBoostM1 |           10 |      0.68739 |          42 |
|   17 | Accept |      0.3256 |     0.82383 |     0.32086 |     0.32106 |          Bag |           10 |            - |           6 |
|   18 | Accept |     0.32939 |     0.46064 |     0.32086 |     0.32106 |   AdaBoostM1 |           10 |    0.0020099 |         159 |
|   19 | Accept |     0.32607 |      1.0118 |     0.32086 |     0.32088 |  GentleBoost |           37 |     0.098878 |         415 |
|   20 | Accept |     0.32726 |      1.3397 |     0.32086 |     0.32089 |  GentleBoost |           34 |      0.06318 |           2 |
|===================================================================================================================================|
| Iter | Eval   | Objective   | Objective   | BestSoFar   | BestSoFar   |       Method | NumLearningC-|    LearnRate | MinLeafSize |
|      | result |             | runtime     | (observed)  | (estim.)    |              | ycles        |              |             |
|===================================================================================================================================|
|   21 | Accept |     0.34004 |     0.76934 |     0.32086 |     0.32089 |          Bag |           10 |            - |           1 |
|   22 | Accept |     0.32418 |      9.7642 |     0.32086 |       0.321 |          Bag |          227 |            - |         105 |
|   23 | Accept |     0.33199 |     0.52758 |     0.32086 |     0.32101 |   AdaBoostM1 |           10 |    0.0056627 |           1 |
|   24 | Accept |     0.32205 |     0.58654 |     0.32086 |       0.321 |   LogitBoost |           24 |      0.63232 |        1980 |
|   25 | Accept |     0.35733 |     0.80538 |     0.32086 |       0.321 |   LogitBoost |           24 |    0.0033013 |         268 |
|   26 | Best   |      0.3192 |      2.8866 |      0.3192 |     0.31923 |   LogitBoost |          114 |      0.29438 |        2044 |
|   27 | Accept |     0.34573 |     0.46434 |      0.3192 |     0.31923 |   LogitBoost |           12 |       0.2975 |          17 |
|   28 | Accept |     0.35733 |     0.43575 |      0.3192 |     0.31973 |   LogitBoost |           10 |     0.010264 |           5 |
|   29 | Accept |     0.32607 |     0.84213 |      0.3192 |     0.31931 |  GentleBoost |           28 |     0.006032 |          10 |
|   30 | Accept |     0.32915 |      1.3946 |      0.3192 |     0.31931 |     RUSBoost |           20 |      0.14638 |           1 |

__________________________________________________________
Optimization completed.
MaxObjectiveEvaluations of 30 reached.
Total function evaluations: 30
Total elapsed time: 118.0837 seconds
Total objective function evaluation time: 79.4366

Best observed feasible point:
      Method      NumLearningCycles    LearnRate    MinLeafSize
    __________    _________________    _________    ___________

    LogitBoost           114            0.29438        2044

Observed objective function value = 0.3192
Estimated objective function value = 0.31931
Function evaluation time = 2.8866

Best estimated feasible point (according to models):
      Method      NumLearningCycles    LearnRate    MinLeafSize
    __________    _________________    _________    ___________

    LogitBoost           114            0.29438        2044

Estimated objective function value = 0.31931
Estimated function evaluation time = 2.7962
```





Calculate error and accuracy of the Model for train data using the % function for resubstitution loss. %

```
trainErrorm4 = resubLoss(Mdl4);
trainaccuracym4 = 1 -trainErrorm4;
```

From workspace it can be seen that the accuracy of the fourth model was also not better than the second model, in spite of optimizing the % hyperparameters. The inbuilt fucntion used to do this gives results of % hyperparameters optimization of both observed values and estimated % values. %

```
% Assign the observed values of hyperparameters optimization to a variable.

bestHyperparameters = Mdl4.HyperparameterOptimizationResults.XAtMinObjective

% After executing this section of code mutiple times, it was observed that
% for most of runs of this script, LogitBoost or AdaBoostM1 methods were %
% resulted after hyperparameters optimization. %

% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

```
bestHyperparameters =

  1×4 table
```

Method	NumLearningCycles	LearnRate	MinLeafSize
LogitBoost	114	0.29438	2044

Fourth iteration to improve the model. % Fit train data into fifth ensemble algorithm, in which Bag is continued % to be the method for the reason mentioned in the section with code for % second training model. The parameters of leaf size and number of learning cycles were assigned best the values hyperparameter optimization model. %

```
rng(1)
templm5 = templateTree("MinLeafSize",bestHyperparameters.MinLeafSize);
Mdl5 = fitcensemble(train_data,'two_year_recid','Method','Bag',...
    'NumLearningCycles',bestHyperparameters.NumLearningCycles);

% Code reference: %
% Statistics and Machine Learning Toolbox™ User's Guide %
% Revision March 2021, R2021a, Chapter 18 %
```

Calculate error and accuracy of the Model for train data using the % function for resubstitution loss. %

```
trainErrorm5 = resubLoss(Mdl5);
trainaccuracym5 = 1 -trainErrorm5;
```

After executing the section of code where the fifthe model was trained, % multiple times, it was seen that the accuracy of the fifth model was also not better than the second model for most of the runs, in spite of % using the best values observed in hyperparameters optimization. %

```
% Fifth iteration to improve the model. %
% Fit train data into fifth ensemble algorithm, in which the predictor
% selection argument is changed from the default CART, keeping rest of the
% parameters same as the second model. %

rng(1)
templm6 = templateTree("PredictorSelection",'curvature');
Mdl6 = fitcensemble(train_data,'two_year_recid','Method','Bag');
```

Calculate error and accuracy of the Model for train data using the % function for resubstitution loss. %

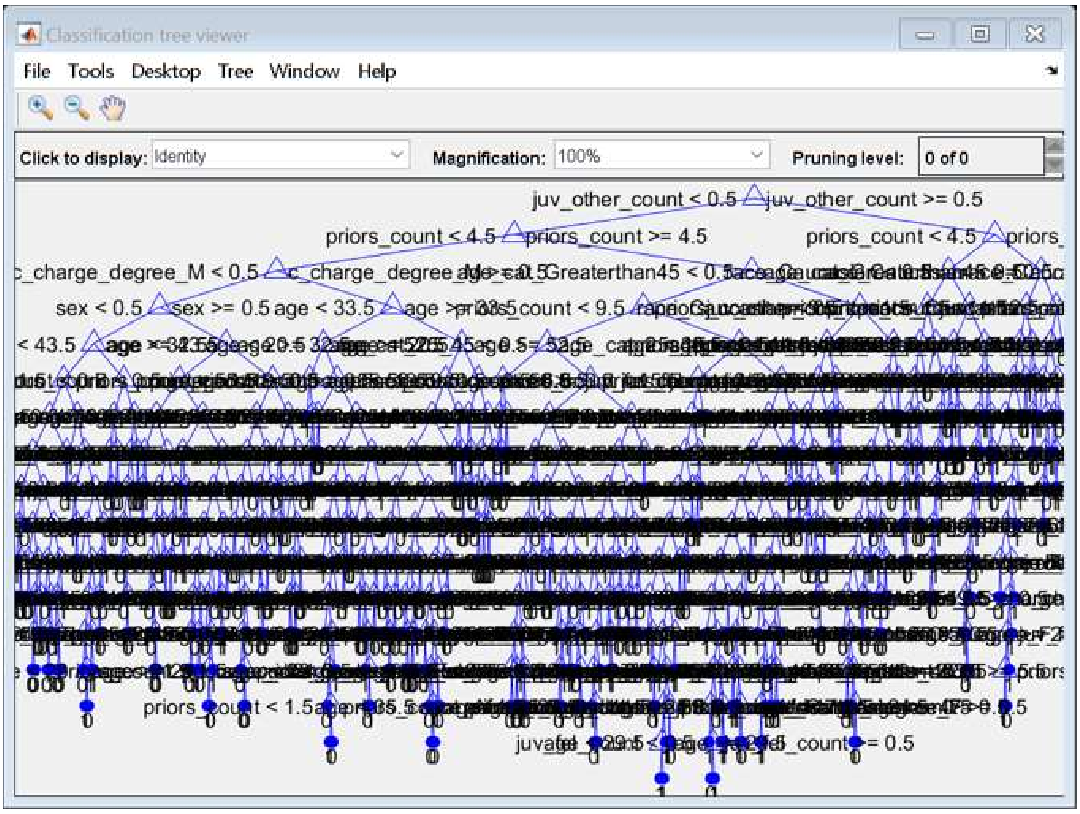
```
trainErrorm6 = resubLoss(Mdl6);
trainaccuracym6 = 1 -trainErrorm6;
```

From workspace it can be seen that the accuracy of the sixth model was also not better than the second model, but mostly same as the second model. %

```
% Although the fifth model gave highest accuracy in few of the runs, it is
% not consistently the best model and hence cannot be considered as the
% best model. %

% It can be concluded that the second model with just the method as bag %
% and rest of the parameters same as the baseline model can be considered %
% as the best model. %

% Cross validate the best model to understand the generalized behavior of
% the model. %
CVMdl2 = crossval(Mdl2);
cvfitm2 = kfoldPredict(CVMdl2);

% Calculate loss and accuracy of cross validated model for train data. %
cvlossm2 = kfoldLoss(CVMdl2);
cvaccuracym2 = 1 - cvlossm2;
```

Plot the baseline model. %

```
view(Mdl2.Trained{1},"Mode","graph");
```

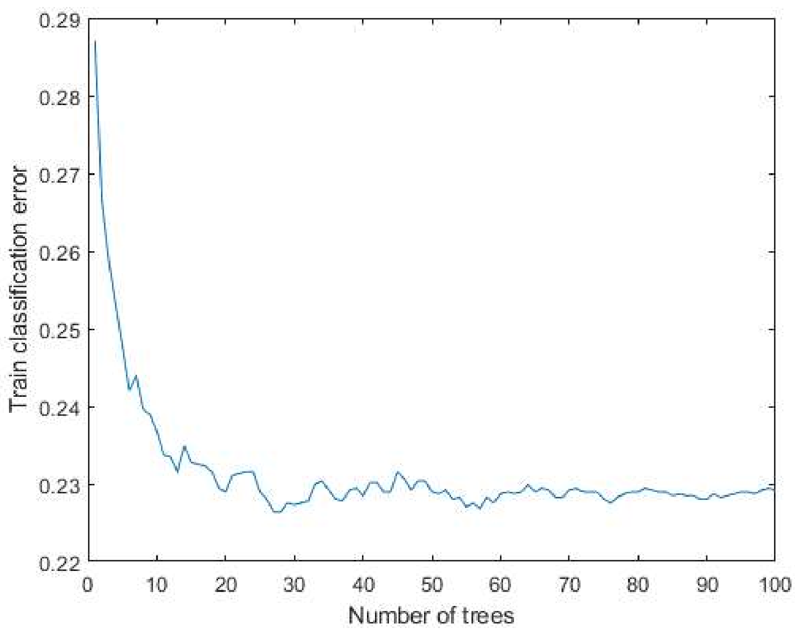


Calculate the Out of the bag loss for the best model. %

```
ooblossm2 = oobLoss(Mdl2);
```

Plot misclassification of the best model as a function of the % number of trained trees in the ensemble. %

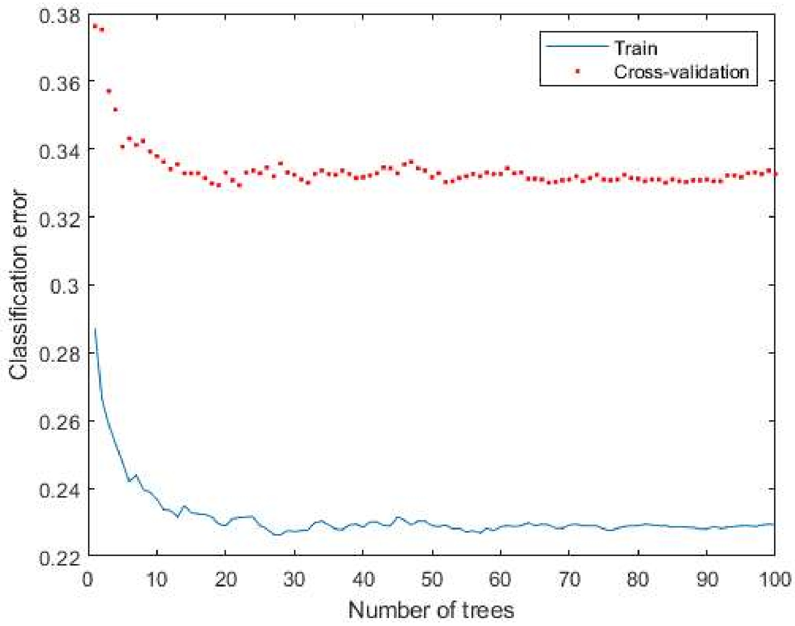
```
figure
plot(loss(Mdl2,x_train,y_train,'mode','cumulative'))
xlabel('Number of trees')
ylabel('Train classification error')
```



Plot misclassification of the best model and cross validation loss as % a function of the number of trained trees in the ensemble. %

```
figure
plot(loss(Mdl2,x_train,y_train,'mode','cumulative'))
hold on
plot(kfoldLoss(CVMdl2,'mode','cumulative'),'r.')
```

```
hold off
xlabel('Number of trees')
ylabel('Classification error')
legend('Train','Cross-validation','Location','NE')
```



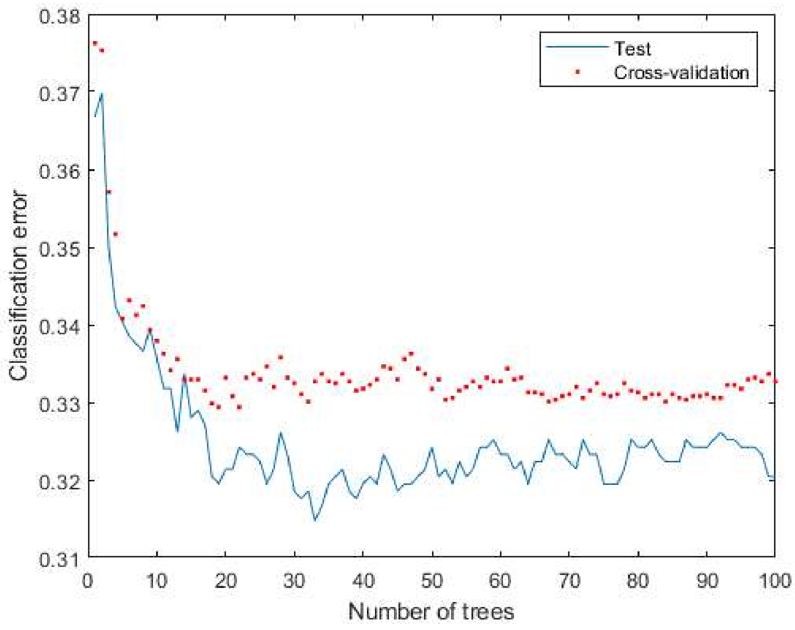
Predict Response for test data using the best Model. %

```
rng(1);
tic
yfittest = predict(Mdl2,x_test);
toc

% Calculate error and accuracy of the Model for test data using the %
% comparision Vector. %

vtest = (yfittest == y_test.two_year_recid);
testError = 1- sum(vtest)/size(vtest,1);
testaccuracy = sum(vtest)/size(vtest,1);
```

```
Elapsed time is 0.065944 seconds.
```

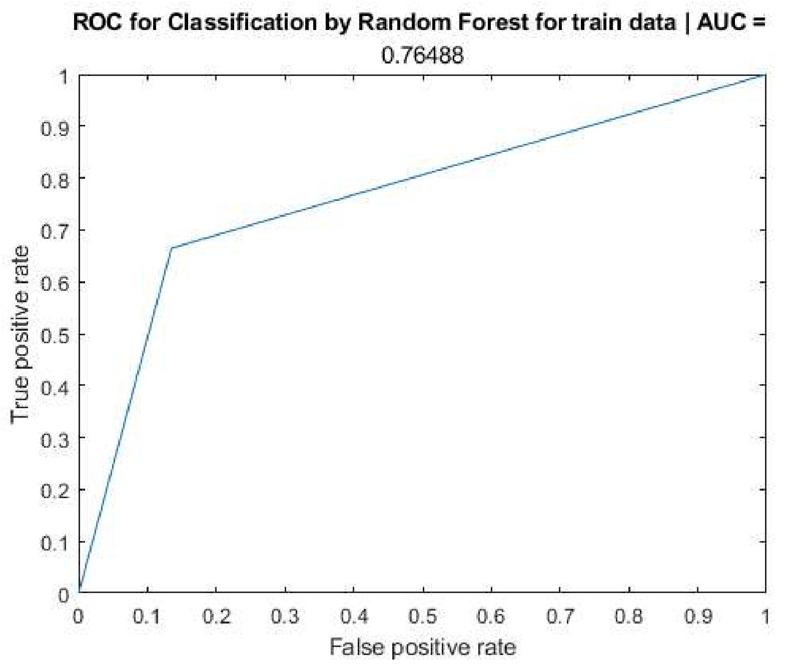
Plot misclassification of the best model and cross validation loss as % a function of the number of trained trees in the ensemble for test data. %

```
figure
plot(loss(Mdl2,x_test,y_test,'mode','cumulative'))
hold on
plot(kfoldLoss(CVMdl2,'mode','cumulative'),'r.')
hold off
xlabel('Number of trees')
ylabel('Classification error')
legend('Test','Cross-validation','Location','NE')
```



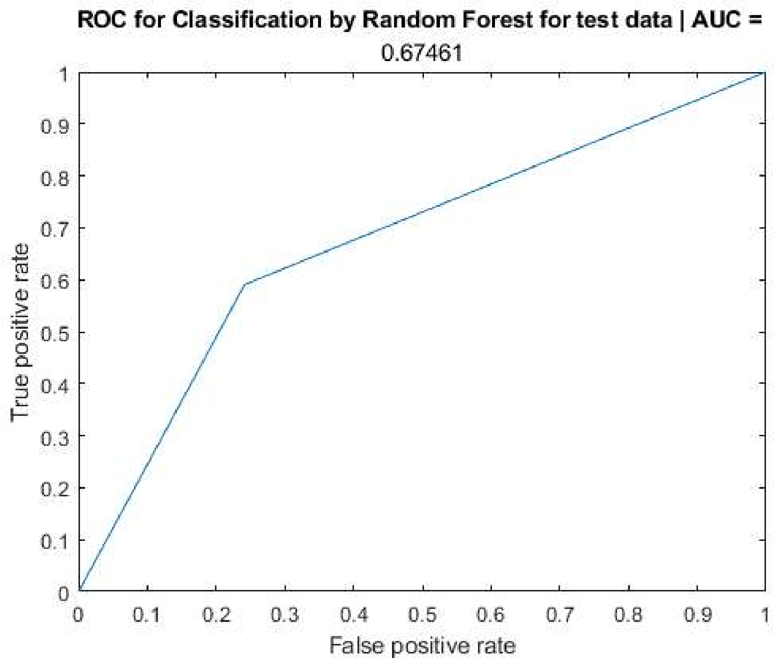
Check AUC of the best model for train data. %

```
yfittrain = predict(Mdl2,x_train);
[Xtr,Ytr,Ttr,AUCtr] = perfcurve(y_train.two_year_recid,yfittrain,'1');

% Plot ROC of the model for train data. %
plot(Xtr,Ytr)
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification by Random Forest for train data | AUC = ',AUCtr)
```



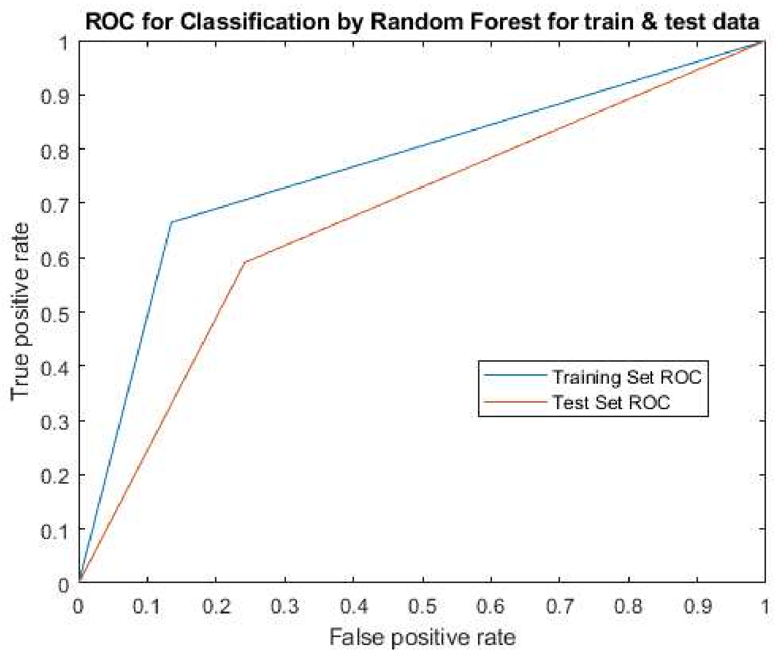
Check AUC of the best model for test data. %

```
[Xte,Yte,Tte,AUCte] = perfcurve(y_test.two_year_recid,yfittest,'1');

% Plot ROC of the model for test data. %
plot(Xte,Yte)
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification by Random Forest for test data | AUC = ',AUCte)
```



Plot ROC of the model for both train and test data for comparison. %

```
plot(Xtr,Ytr)
hold on
plot(Xte,Yte)
legend('Training Set ROC', 'Test Set ROC',Location='best')
xlabel('False positive rate')
ylabel('True positive rate')
title('ROC for Classification by Random Forest for train & test data')
hold off
```

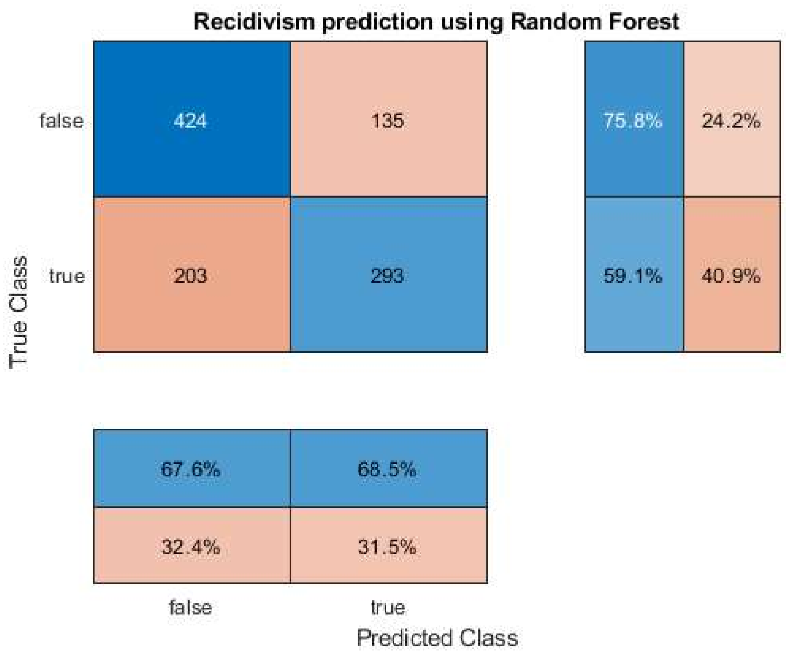


Metrics for Model performance for test data. %

```
% Convert table of original and predicted values of response variable for %
% test data into a logical array, to use further. %
y_test_ar = table2array(y_test);
y_test_lg = logical(y_test_ar);
yfittest_lg = logical(yfittest);

% Plot and assign Confusion Matrix for test data. %
conchart = confusionchart(y_test_lg,yfittest_lg);
conchart.Title = 'Recidivism prediction using Random Forest'
conchart.RowSummary = 'row-normalized'
conchart.ColumnSummary = 'column-normalized'
```

```
% Assing variable to Confusion Matrix for test data. %
confmat = confusionmat(y_test_lg,yfittest_lg);
confmat;

% Assigning variables to Components of Confusion Matrix viz., %
% True Negative, True Positive, False Negative, and False Poistive. %
TN = confmat(1,1);
TP = confmat(2,2);
FN = confmat(2,1);
FP = confmat(1,2);

% Calculating other Model Evaluation Metrics for test data. %
Sensitivity = (TP/(TP + FN));
Specificity = (TN/(TN + FP));
Precision = (TP/(TP + FP));

% Formula Reference: %
% https://en.wikipedia.org/wiki/Confusion_matrix %
% Code Reference: %
% https://uk.mathworks.com/help/stats/confusionchart.html?s_tid=doc_ta %
```

```
conchart =

  ConfusionMatrixChart (Recidivism prediction using Rand…) with properties:

    NormalizedValues: [2×2 double]
         ClassLabels: [2×1 logical]

  Use GET to show all properties


conchart =

  ConfusionMatrixChart (Recidivism prediction using Rand…) with properties:

    NormalizedValues: [2×2 double]
         ClassLabels: [2×1 logical]

  Use GET to show all properties


conchart =

  ConfusionMatrixChart (Recidivism prediction using Rand…) with properties:

    NormalizedValues: [2×2 double]
         ClassLabels: [2×1 logical]

  Use GET to show all properties
```



END %%

Published with MATLAB® R2021a