```
function [trainedClassifier, validationAccuracy] =
 trainClassifier(trainingData)
% trainClassifier(trainingData)
%  returns a trained classifier and its accuracy.
%  This code recreates the classification model trained in
%  Classification Learner app.
%
%   Input:
%       trainingData: the training data of same data type as imported
%        in the app (table or matrix).
%
%   Output:
%       trainedClassifier: a struct containing the trained classifier.
%        The struct contains various fields with information about the
%        trained classifier.
%
%       trainedClassifier.predictFcn: a function to make predictions
%        on new data. It takes an input of the same form as this
 training
%        code (table or matrix) and returns predictions for the
 response.
%        If you supply a matrix, include only the predictors columns
 (or
%        rows).
%
%       validationAccuracy: a double containing the accuracy in
%        percent. In the app, the History list displays this
%        overall accuracy score for each model.
%
%  Use the code to train the model with new data.
%  To retrain your classifier, call the function from the command line
%  with your original data or new data as the input argument
 trainingData.
%
%  For example, to retrain a classifier trained with the original data
 set
%  T, enter:
%    [trainedClassifier, validationAccuracy] = trainClassifier(T)
%
%  To make predictions with the returned 'trainedClassifier' on new
 data T,
%  use
%    yfit = trainedClassifier.predictFcn(T)
%
%  To automate training the same classifier with new data, or to learn
 how
%  to programmatically train classifiers, examine the generated code.

% Auto-generated by MATLAB on 17-May-2016 15:13:50


% Extract predictors and response
```

```
% This code processes the data into the right shape for training the
% classifier.
inputTable = trainingData;
predictorNames =
 {'wellnum', 'wellName', 'x0', 'strain', 'ageDpf', 'nFish', 'minutesIncubation', '
predictors = inputTable(:, predictorNames);
response = inputTable.DMSO;
isCategoricalPredictor = [false, true, false, true, false, false,
 false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the
 classifier.
classificationTree = fitctree(...
    predictors, ...
    response, ...
    'SplitCriterion', 'gdi', ...
    'MaxNumSplits', 100, ...
    'Surrogate', 'off', ...
    'ClassNames', {'DMSO'; 'Haloperidol'});

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
treePredictFcn = @(x) predict(classificationTree, x);
trainedClassifier.predictFcn = @(x)
 treePredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables =
 {'wellnum', 'wellName', 'x0', 'strain', 'ageDpf', 'nFish', 'minutesIncubation', '
trainedClassifier.ClassificationTree = classificationTree;
trainedClassifier.About = 'This struct is a trained classifier
 exported from Classification Learner R2016a.';
trainedClassifier.HowToPredict = sprintf('To make predictions
 on a new table, T, use: \n  yfit = c.predictFcn(T) \nreplacing
 ''c'' with the name of the variable that is this struct, e.g.
 ''trainedClassifier''. \n \nThe table, T, must contain the variables
 returned by: \n  c.RequiredVariables \nVariable formats (e.g.
 matrix/vector, datatype) must match the original training data.
 \nAdditional variables are ignored. \n \nFor more information, see
 <a href="matlab:helpview(fullfile(docroot, ''stats'', ''stats.map''),
 ''appclassification_exportmodeltoworkspace'')">How to predict using
 an exported model</a>.');

% Extract predictors and response
% This code processes the data into the right shape for training the
% classifier.
inputTable = trainingData;
predictorNames =
 {'wellnum', 'wellName', 'x0', 'strain', 'ageDpf', 'nFish', 'minutesIncubation', '
predictors = inputTable(:, predictorNames);
response = inputTable.DMSO;
isCategoricalPredictor = [false, true, false, true, false, false,
 false, false, false];
```

```
% Perform cross-validation
partitionedModel =
 crossval(trainedClassifier.ClassificationTree, 'KFold', 5);

% Compute validation accuracy
validationAccuracy = 1 -
 kfoldLoss(partitionedModel, 'LossFun', 'ClassifError');

% Compute validation predictions and scores
[validationPredictions, validationScores] =
 kfoldPredict(partitionedModel);
```

Published with MATLAB® R2016a