```
function [trainedClassifier, validationAccuracy] =
 trainClassifier(trainingData)
% [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% Returns a trained classifier and its accuracy. This code recreates the
% classification model trained in Classification Learner app. Use the
% generated code to automate training the same model with new data, or to
% learn how to programmatically train models.
%
%  Input:
%      trainingData: A table containing the same predictor and response
%       columns as those imported into the app.
%
%  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.
%
%      validationAccuracy: A double containing the accuracy as a
%       percentage. In the app, the Models pane 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 T2,
% use
%   yfit = trainedClassifier.predictFcn(T2)
%
% T2 must be a table containing at least the same predictor columns as used
% during training. For details, enter:
%   trainedClassifier.HowToPredict

% Auto-generated by MATLAB on 17-Nov-2022 11:52:18


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = trainingData;
predictorNames = {'SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth'};
predictors = inputTable(:, predictorNames);
response = inputTable.Species;
isCategoricalPredictor = [false, false, false, false];
```

```
% Train a classifier
% This code specifies all the classifier options and trains the classifier.
classificationDiscriminant = fitcdiscr(...
    predictors, ...
    response, ...
    'DiscrimType', 'linear', ...
    'Gamma', 0, ...
    'FillCoeffs', 'off', ...
    'ClassNames', categorical({'setosa'; 'versicolor'; 'virginica'}));

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

% Add additional fields to the result struct
trainedClassifier.RequiredVariables =
 {'PetalLength', 'PetalWidth', 'SepalLength', 'SepalWidth'};
trainedClassifier.ClassificationDiscriminant = classificationDiscriminant;
trainedClassifier.About = 'This struct is a trained model exported from
 Classification Learner R2022a.';
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. ''trainedModel''. \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
% model.
inputTable = trainingData;
predictorNames = {'SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth'};
predictors = inputTable(:, predictorNames);
response = inputTable.Species;
isCategoricalPredictor = [false, false, false, false];

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

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

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

Not enough input arguments.
```

```
Error in E16 (line 46)
inputTable = trainingData;
```

Published with MATLAB® R2022a