A guided tour to Machine Learning using MATLAB®

Introduction

- This document guides you through several tutorials, papers, and resources related to Machine Learning (with emphasis on image and vision tasks) using **MATLAB.**
- It assumes no prior exposure to Machine Learning or MATLAB.
- It is structured as a step-by-step guide. It is best that you follow it in the intended sequence.

Part 1- Accessing MATLAB

You are expected to have frequent access to a computer running MATLAB and some of its toolboxes (notably, the Image Processing, Computer Vision System, Statistics and Machine Learning, Neural Network, and Fuzzy Logic toolboxes) for your assignments and projects.

Here are some options to consider:

1. Purchase your own copy of the student version of MATLAB. It costs $99, is fully functional, and comes with several toolboxes. For more details, go to: http://www.mathworks.com/academia/student_version/

2. Access via the Engineering Cloud (https://tsg.eng.fau.edu/software/matlab/).
   Instructions on how to access the Engineering Cloud are posted here:
   https://tsg.eng.fau.edu/software/vmware-remote-desktop-access/
   Once connected, select the "All Engineering Students" pool of windows desktops.
   Once logged in to windows, click on Start -> All Programs -> MATLAB 2016a -> MATLAB 2016a. Be sure to save any files to your Z: drive.
   If you have issues connecting or starting MATLAB, please contact help@eng.fau.edu.

3. Consider a free alternative to MATLAB. The most popular is GNU Octave (http://www.gnu.org/software/octave/), which has been used by many Machine Learning professors, students, and researchers. They are "work in progress" and are not 100% compatible with MATLAB. So please use them at your own risk.

Part 2- Learning the basics of MATLAB

1. Take the **MATLAB Onramp** training available at https://matlabacademy.mathworks.com/R2016a/portal.html?course=gettingstarted (see separate document for step-by-step instructions).

2. (OPTIONAL) Watch the 46-min "Introduction to MATLAB" video: www.mathworks.com/videos/introduction-to-matlab-81592.html
   Don't forget to download the associated source code: http://www.mathworks.com/matlabcentral/fileexchange/49570-introduction-to-matlab--february-2015-

Part 3- Learning the basics of Machine Learning in MATLAB

1. Read the "Introducing Machine Learning" e-book (available on Canvas).

2. Read "Supervised Learning Workflow and Algorithms" https://www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html

3. (OPTIONAL) Watch the 35-min "Machine Learning Made Easy" video: www.mathworks.com/videos/machine-learning-with-matlab-100694.html
   Don't forget to download the associated source code: http://www.mathworks.com/matlabcentral/fileexchange/50232-machine-learning-made-easy

4. (OPTIONAL) Watch the 41-min "Machine Learning with MATLAB" video: http://www.mathworks.com/videos/machine-learning-with-matlab-81984.html
   Don't forget to download the associated source code: http://www.mathworks.com/matlabcentral/fileexchange/42744-machine-learning-with-matlab

Part 4- Classification using decision trees in MATLAB

Inspired by the steps at
https://www.mathworks.com/help/stats/classification-trees-and-regression-trees.html

1. Run the example file **dtIntro.m**, paying attention to the following aspects:
    a. How to load a dataset (in this case, it's already available in **.mat** format)
    b. How to create a decision tree, view it, and use it to make a prediction using unseen data
    c. How to compute resubstitution error of the resulting classification tree
    d. How to compute cross-validation accuracy
    e. How to select the appropriate tree depth
    f. How to prune the tree

2. Run the example file **dtIris.m**, paying attention to the following aspects:
    a. How to load a dataset (in this case, it's already available in **.mat** format)
    b. How to plot different views of the dataset (whenever feasible) in order to better understand the data
    c. How to create a decision tree, view it, and use it to make a prediction using unseen data
    d. How to compute resubstitution error of the resulting classification tree
    e. How to compute cross-validation accuracy

Part 5- Linear Regression in MATLAB

1. Run the examples in the **'Stanford'** subfolder. They are from Andrew Ng's "Machine Learning" course (MOOC) – Stanford University – Fall 2011.
    a. **ex1.m** shows linear regression for one variable
    b. **ex1_multi.m** shows linear regression with multiple variables. It also introduces *feature normalization* and the *normal equation* method (an alternative to gradient descent)

2. Run the example **IntroToLinearRegression.m** in the **'Mathworks'** subfolder.
    a. It is based on https://www.mathworks.com/help/matlab/data_analysis/linear-regression.html. It builds and compares two simple linear regression models and introduces the coefficient of determination.

3. Run the examples in the **'Regression_Demos'** subfolder. They are also available at: http://www.mathworks.com/matlabcentral/fileexchange/35789-new-regression-capabilities-in-r2012a
    a. (OPTIONAL, but recommended) Watch the associated webinar / video: https://www.mathworks.com/videos/regression-analysis-with-matlab-new-statistics-toolbox-capabilities-in-r2012a-81869.html
    b. Explore the examples following this sequence: StraightLine.m, CurvesSurfaces.m, and NonLinear.m. (Skip the Housing.m, Model.m and the GLMs.m examples)
    c. Don't be intimidated or discouraged by the rich amount of information available in some MATLAB objects, e.g., LinearModel.

Part 6- Logistic Regression in MATLAB

1. Run the examples in the **'Stanford'** subfolder. They are from Andrew Ng's "Machine Learning" course (MOOC) – Stanford University – Fall 2011.
    a. **ex2.m** shows logistic regression
    b. **ex2_reg.m** shows the use of additional polynomial features and the impact of regularization on logistic regression with multiple variables. Don't forget to change the value of the regularization parameter, *lambda*, in line 90, and run that section every time you do so. Notice how the resulting decision boundary changes as a result of changes in *lambda*.

Part 7- The Classification Learner App

Goal: Learn how to use the MATLAB Classification Learner App to perform 3-class classification on the Fisher's Iris dataset.

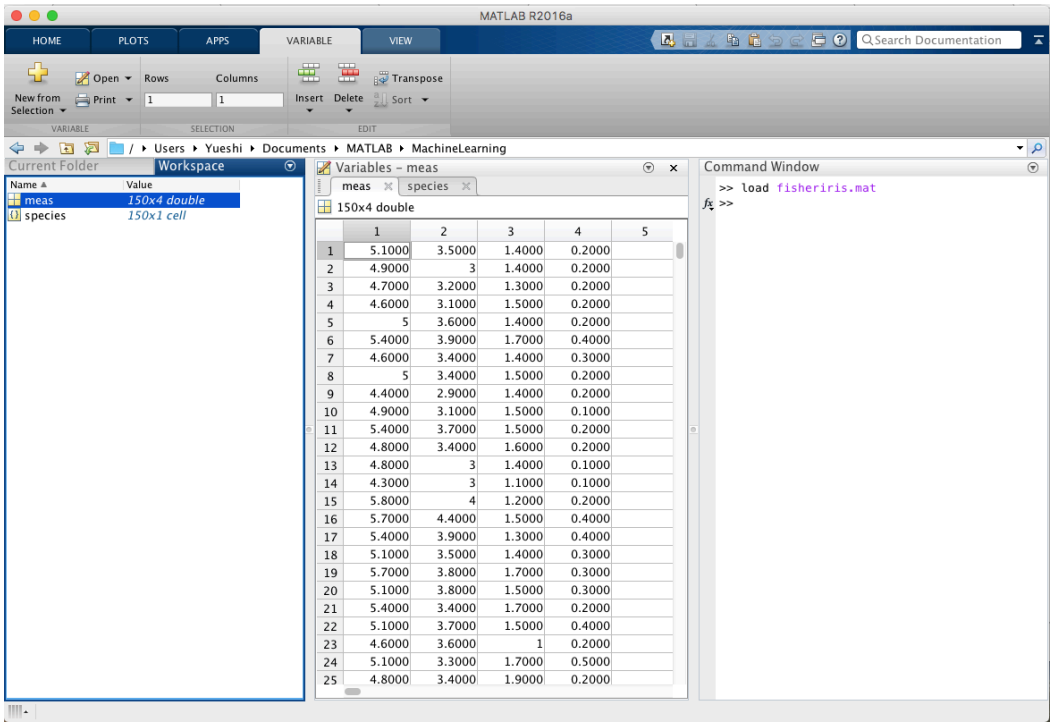
1. Dataset:

   In this example, we will use the **Fisher's Iris dataset**.
   This is a sample dataset included in the MATLAB Statistics and Machine Learning Toolbox.
   You can find all sample datasets at:
   https://www.mathworks.com/help/stats/_bq9uxn4.html

2. View the dataset:

   **This step is not necessary, just to give an idea how this dataset looks like*
   To load a dataset into the MATLAB workspace, type: *load filename*
   In this particular example, we will type in **Command Window**: *load fisheriris.mat*

   You can view the datasets loaded to the workspace by double clicking the matrix name under **Workspace** window.
   (Please notice that you may have a different window layout than the screenshot below)
   In this example, **meas** is a 150*4 double matrix. There are 150 rows each row represents one instance. There are 4 columns store attribute information (col1: sepal length in cm; col2: sepal width in cm; col3: petal length in cm; col4: petal width in cm).
   The class for each instance is stored in a separate 150*1 cell called "**species**". In this case, the first 50 instances belong to class *Setosa*, the following 50 belong to class *Versicolor* and the last 50 belong to class *Virginica*.



3. Prepare the data.

We need to first load the *fisheriris* data set and create a table of measurement predictors (or features) using variables from the data set to use for a classification.

Type the following command after the Command Window prompt:

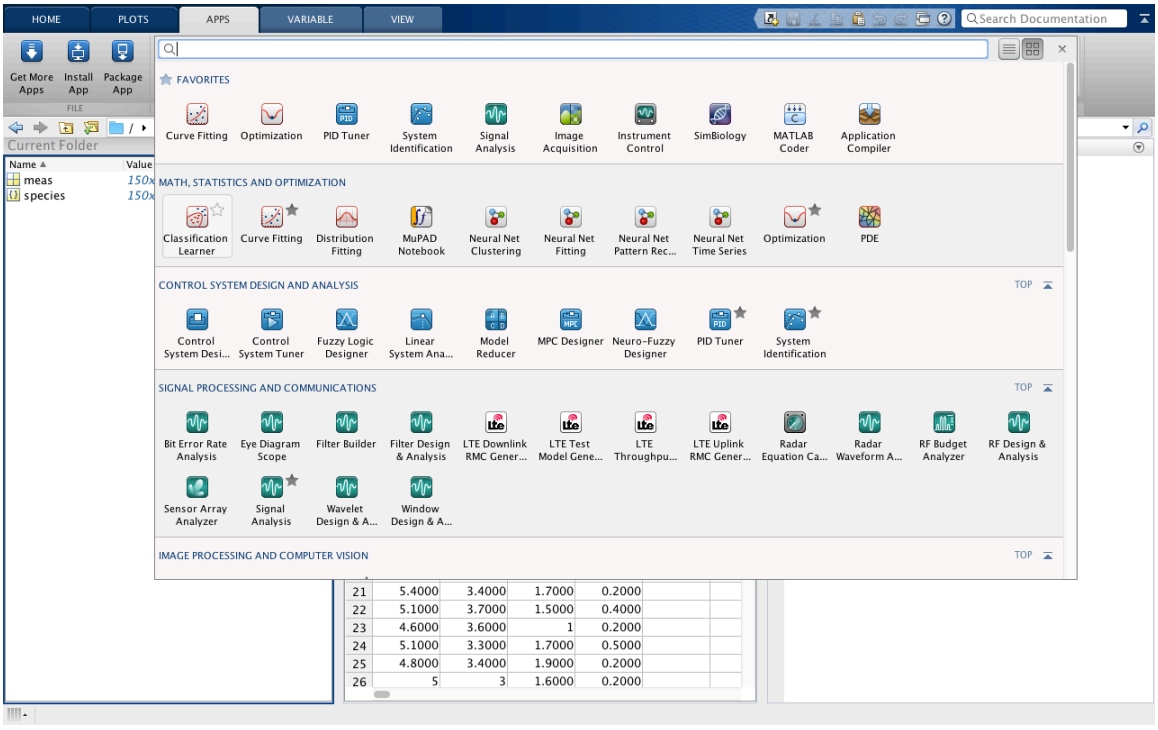
```
fishertable = readtable('fisheriris.csv');
```

4. Start the Classification Learner App.

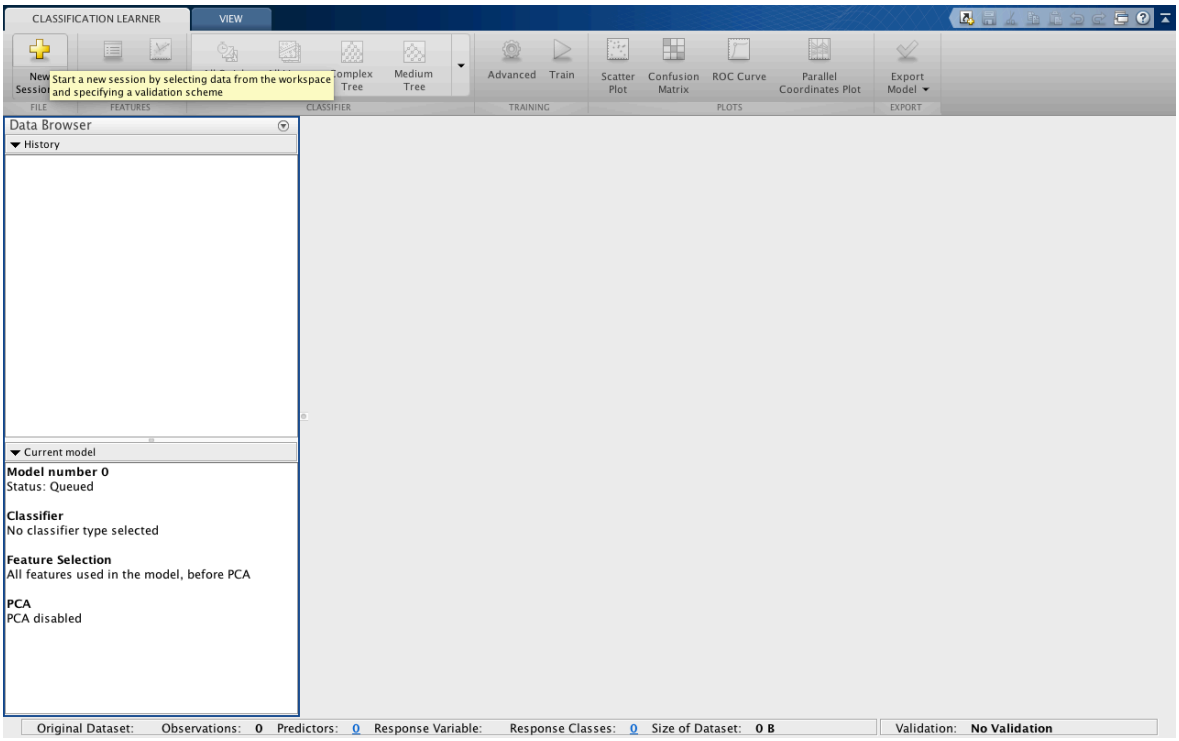
There are two ways of doing this:

a) MATLAB Toolstrip: On the **APPS** tab, under Math, Statistics and Optimization, click the app icon (see screenshot below).

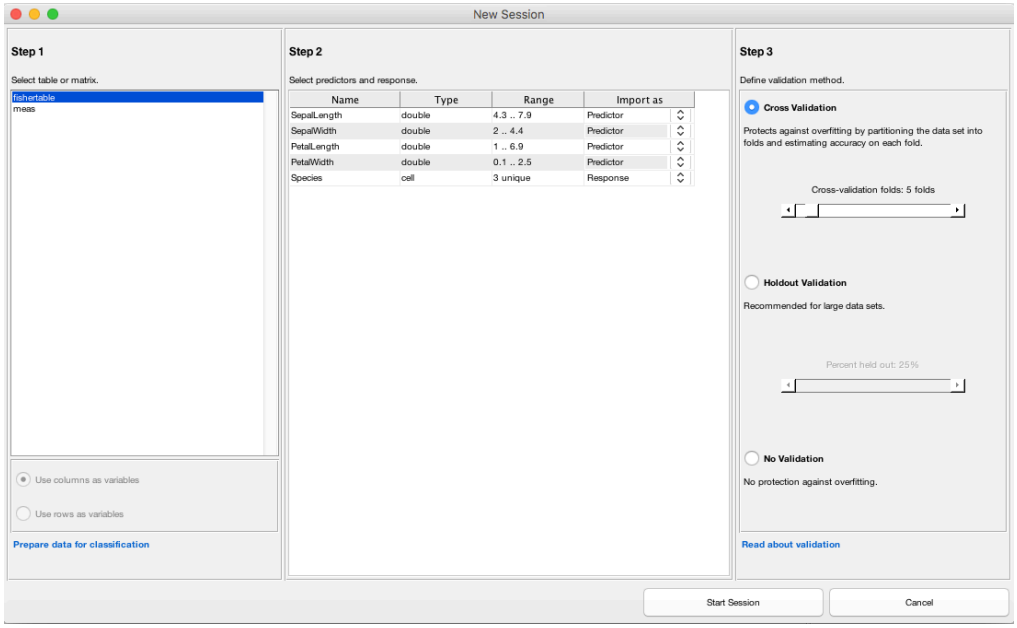
b) MATLAB command prompt: type *classificationLearner*



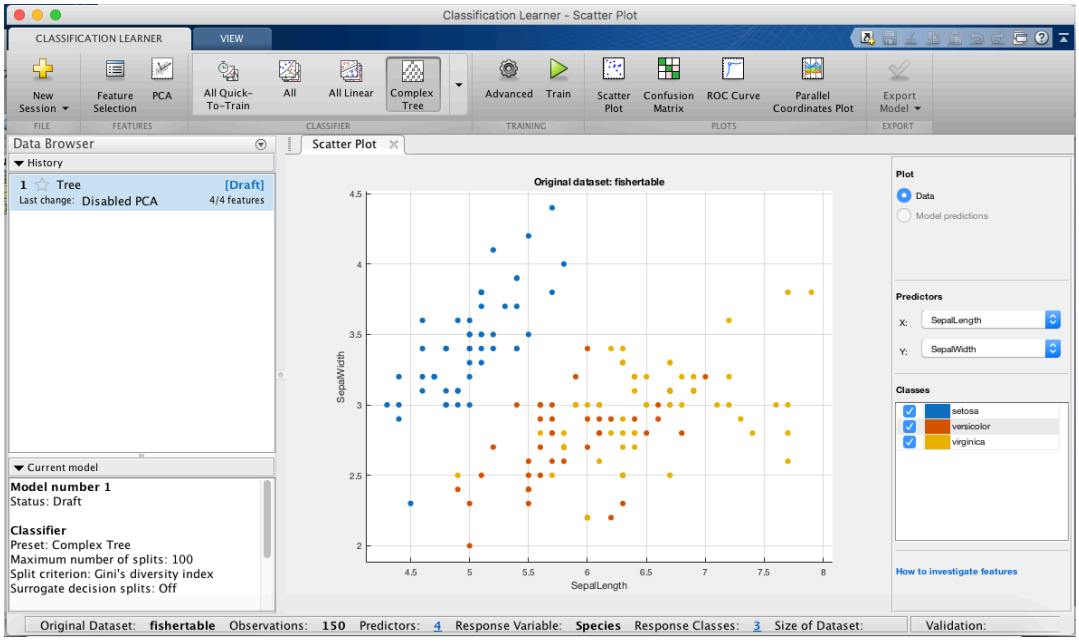
5. On the **Classification Learner** tab, in the **File** section, click **New Session**. (see screenshot below)

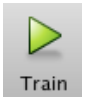


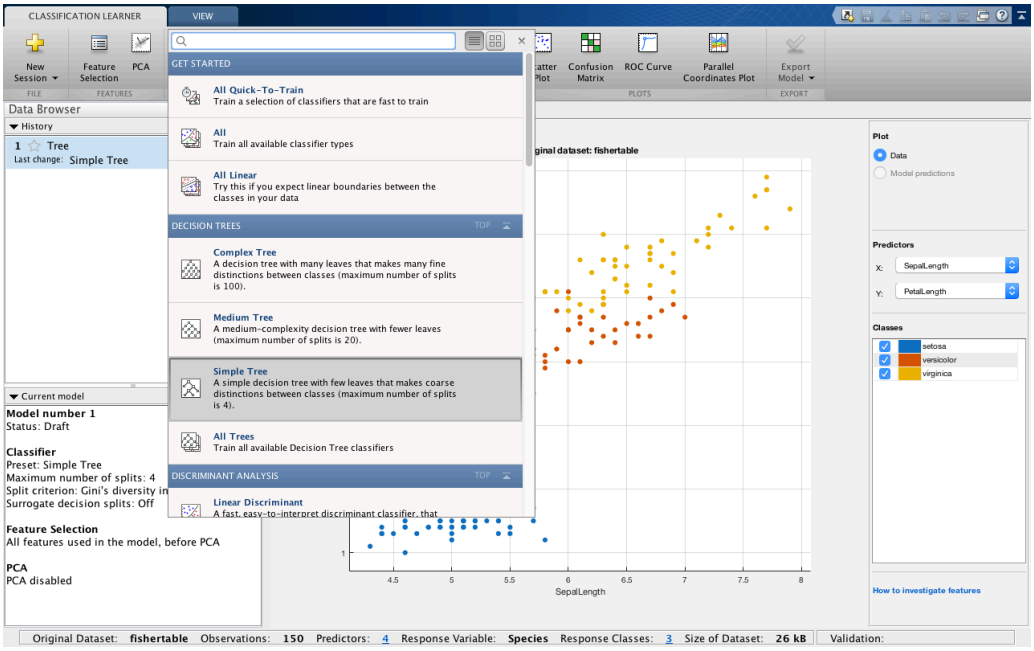
6. In the New Session dialog box, select the table *fishertable* from the workspace list.
   Note: If you did optional step 2, you may find *meas* in the dialog as well; **make sure the *fishertable* is selected.**
   Observe that the app has selected response and predictor variables based on their data type. Petal and sepal length and width are predictors, and species is the response that you want to classify. For this example, do not change the selections.



7. Accept the default validation option (5-fold cross-validation) and continue by clicking **Start Session**. You will see the session like following screenshot.



8. Choose a classification model. In this case, we shall use a simple decision tree.
   To create a classification tree model, on the Classification Learner tab, in the **Classifier section**, click the down arrow to expand the gallery and click **Simple Tree**. Then disable the 'Use Parallel' button (if it's set to ON) and click **Train**. 

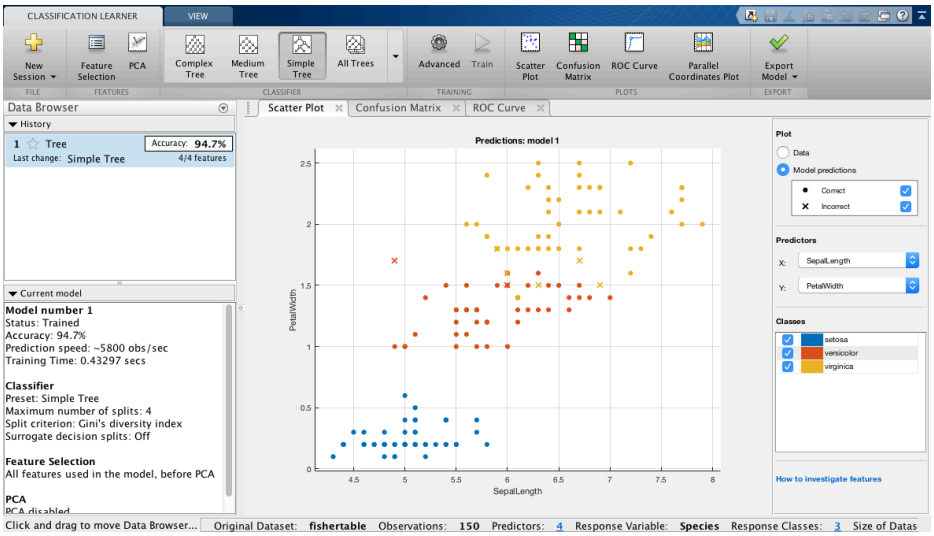


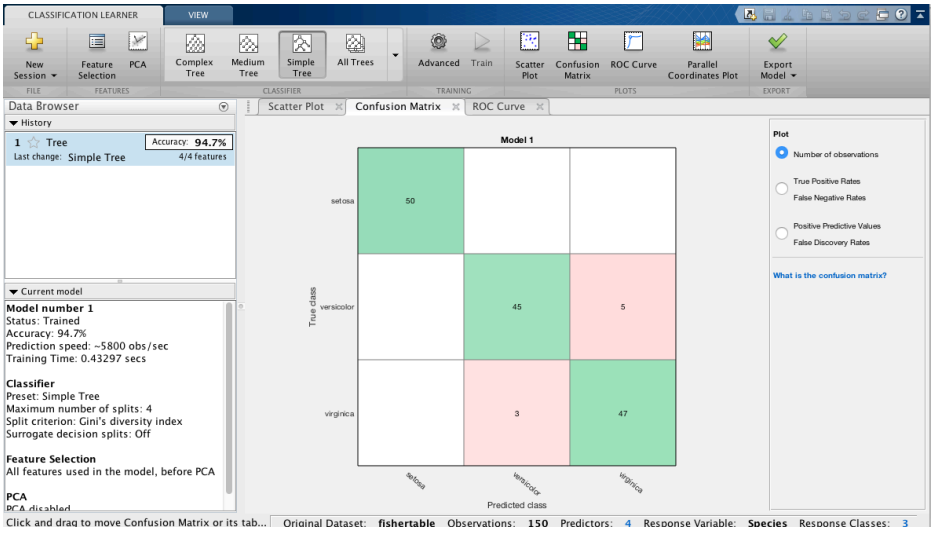
9. Examine results
   The Simple Tree model is now in the History list. The model validation score is in the **Accuracy** box. *This number may be slightly different in your case.*

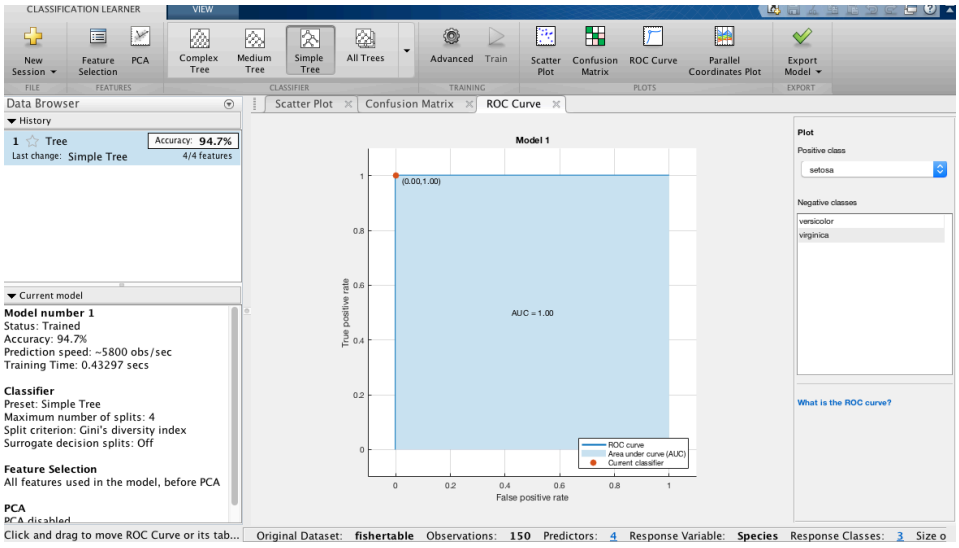
   Examine the **scatter plot**. An X indicates misclassified points. The blue points (setosa species) are all correctly classified, but some of the other two species are misclassified. Under Plot, switch between the Data and Model Predictions options. Observe the color of the incorrect (X) points. Alternatively, while plotting model predictions, to view only the incorrect points, clear the Correct check box.

   On the Classification Learner tab, in the **Plots** section, click **Confusion Matrix** or **ROC Curve** to generate **Confusion Matrix** or **ROC Curve,** respectively. Each plot will open on a separate tab. See representative screenshots on the next page.

   Experiment with changing the settings in each Plot section to fully examine how the currently selected classifier performed in each class.


CLASSIFICATION LEARNER
VIEW
New Session
Feature Selection
PCA
Complex Tree
Medium Tree
Simple Tree
All Trees
Advanced
Train
Scatter Plot
Confusion Matrix
ROC Curve
Parallel Coordinates Plot
Export Model
Data Browser
History
1 Tree
Accuracy: 94.7%
Last change: Simple Tree
4/4 features
Current model
Model number 1
Status: Trained
Accuracy: 94.7%
Prediction speed: ~5800 obs/sec
Training Time: 0.43297 secs
Classifier
Preset: Simple Tree
Maximum number of splits: 4
Split criterion: Gini's diversity index
Surrogate decision splits: Off
Feature Selection
All features used in the model, before PCA
PCA
PCA disabled
Scatter Plot
Confusion Matrix
ROC Curve
Predictions: model 1
PetalWidth
SepalLength
Plot
Data
Model predictions
Correct
Incorrect
Predictors
X:
SepalLength
Y:
PetalWidth
Classes
setosa
versicolor
virginica
How to investigate features
Click and drag to move Data Browser...
Original Dataset: fishertable
Observations: 150
Predictors: 4
Response Variable: Species
Response Classes: 3
Size of Datas

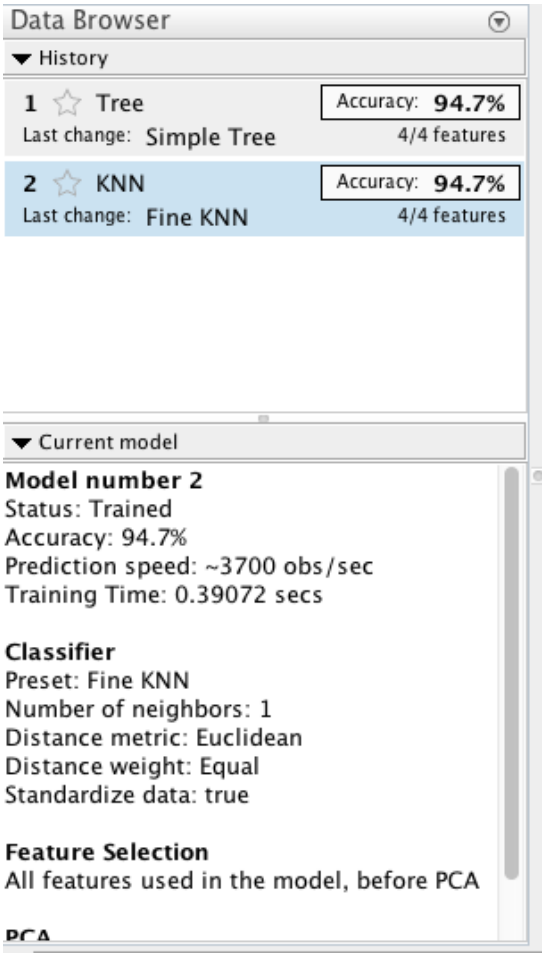

CLASSIFICATION LEARNER
VIEW
Data Browser
History
1 Tree
Accuracy: 94.7%
Last change: Simple Tree
4/4 features
Current model
Model number 1
Status: Trained
Accuracy: 94.7%
Prediction speed: ~5800 obs/sec
Training Time: 0.43297 secs
Classifier
Preset: Simple Tree
Maximum number of splits: 4
Split criterion: Gini's diversity index
Surrogate decision splits: Off
Feature Selection
All features used in the model, before PCA
PCA
PCA disabled
Scatter Plot
Confusion Matrix
ROC Curve
Model 1
setosa
versicolor
virginica
True class
Predicted class
50
45
5
3
47
Plot
Number of observations
True Positive Rates
False Negative Rates
Positive Predictive Values
False Discovery Rates
What is the confusion matrix?
Click and drag to move Confusion Matrix or its tab...
Original Dataset: fishertable
Observations: 150
Predictors: 4
Response Variable: Species
Response Classes: 3


CLASSIFICATION LEARNER
VIEW
Data Browser
History
1 Tree
Accuracy: 94.7%
Last change: Simple Tree
4/4 features
Current model
Model number 1
Status: Trained
Accuracy: 94.7%
Prediction speed: ~5800 obs/sec
Training Time: 0.43297 secs
Classifier
Preset: Simple Tree
Maximum number of splits: 4
Split criterion: Gini's diversity index
Surrogate decision splits: Off
Feature Selection
All features used in the model, before PCA
PCA
PCA disabled
Scatter Plot
Confusion Matrix
ROC Curve
Model 1
(0.00,1.00)
AUC = 1.00
True positive rate
False positive rate
ROC curve
Area under curve (AUC)
Current classifier
Plot
Positive class
setosa
Negative classes
versicolor
virginica
What is the ROC curve?
Click and drag to move ROC Curve or its tab...
Original Dataset: fishertable
Observations: 150
Predictors: 4
Response Variable: Species
Response Classes: 3

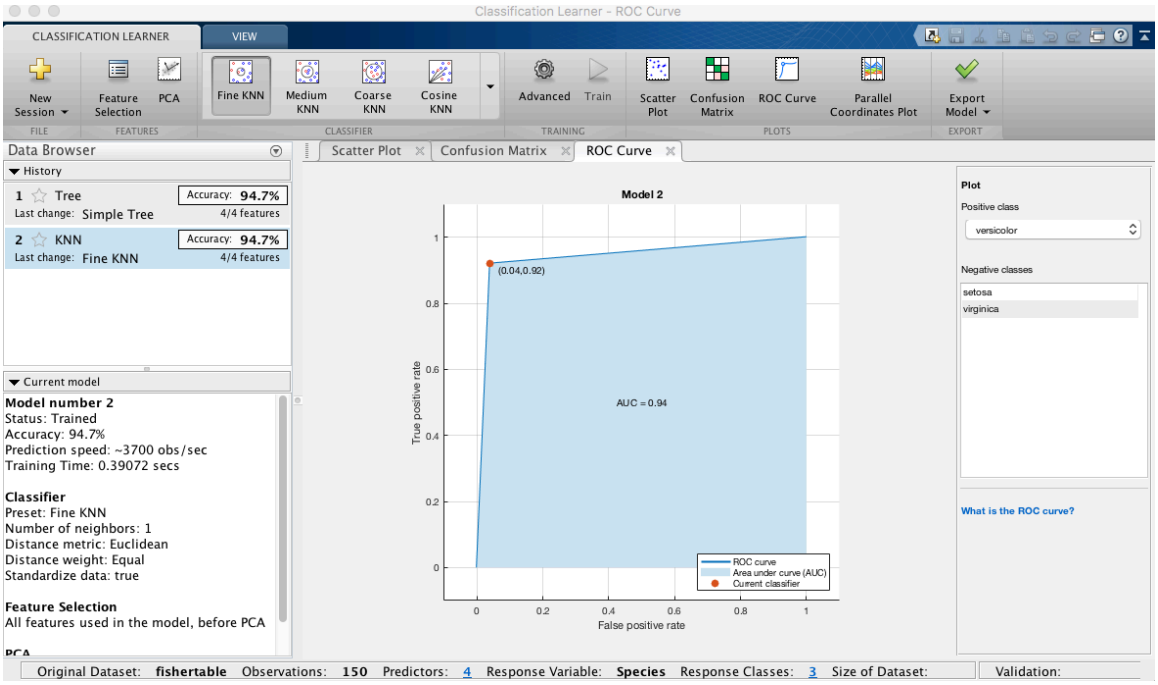
10. Choose another model.

    You can train different models to compare to the decision tree, by choosing other models in the **Classifier** section. In this example, I chose Fine KNN[1]. Click **Fine KNN**, and then click **Train**.

    After training, you can see the Fine KNN in the History list. You can click each model in the History list to view and compare the results. *The accuracy value may be slightly different in your case.*



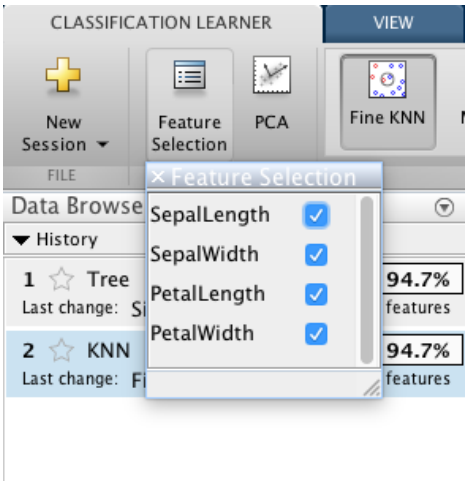
[1] Technically, you don't know what a kNN classifier is, since we haven't covered it in class (yet). But that's on purpose! My goal is to show that you can pick other classifiers, train them, and 'play' with their parameters rather easily, even if you don't quite know what is "inside the box".



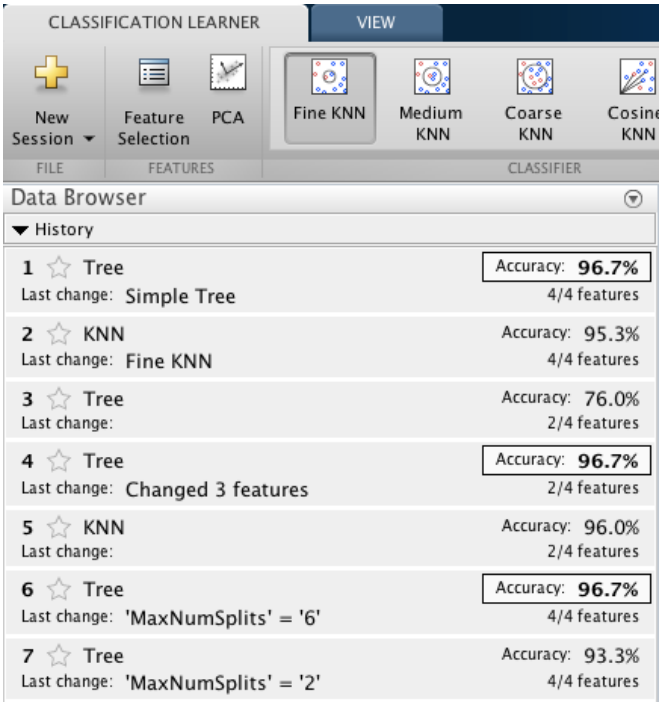
11. Try using different attributes

    To try to improve the model, *try using different features in the model*. See if you can improve the model by removing features with low predictive power.

    On the Classification Learner tab, in the Features section, click **Feature Selection**. You can remove a feature by uncheck the box beside it.

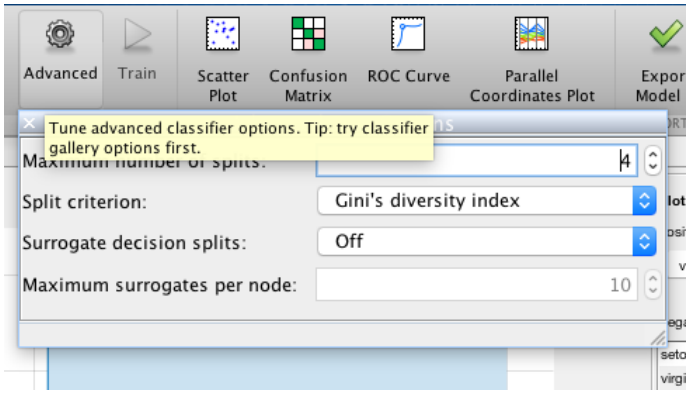


After performing Feature Selection, a new model will appear on the left-hand side of the app. You should then train it and compare the accuracy results (as well as confusion matrix, ROC curve, AUC, etc.) against the previously trained models. MATLAB will indicate the best model so far by highlighting the highest accuracy values.
See screenshot below (obtained after trying 5 variants of decision trees and 2 variants of kNN).



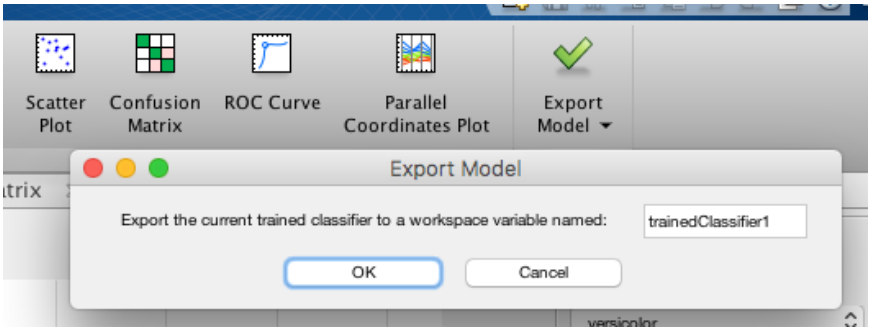
12. Advanced classifier options

    To learn about model settings, choose a model in the History list and view the advanced settings. The options in the Classifier gallery are preset starting points, and you can change further settings. On the Classification Learner tab, in the Training section, click Advanced. For decision trees, consider changing the Maximum Number of Splits setting (which controls tree depth), then train a new model by clicking Train. View the settings for the selected trained model in the Current model pane, or in the Advanced dialog box.



13. Export trained model

To export the best trained model to the workspace, on the **Classification Learner** tab, in the **Export** section, click **Export Model**. In the Export Model dialog box, click **OK** to accept the default variable name *trainedClassifier* or change to another name.



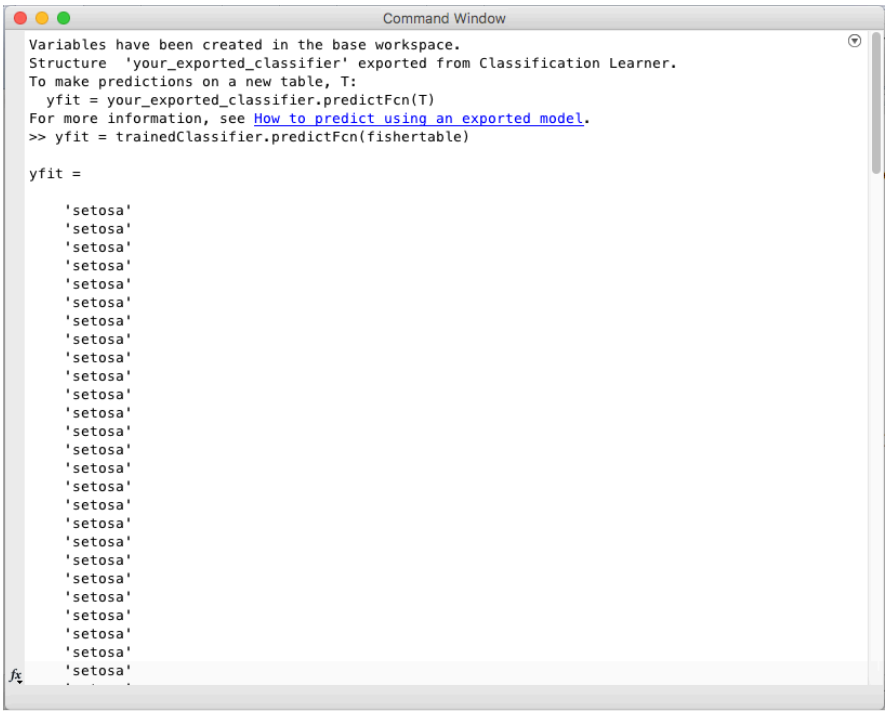
If the exported model is a decision tree called trainedTreeClassifier, use the following line on MATLAB command window to view the resulting model (tree):

```
view(trainedTreeClassifier.ClassificationTree,'mode','graph');
```

You can use the exported classifier to make predictions on new data. For example, to make predictions for the *fishertable* data in your workspace, enter:

```
yfit = trainedClassifier.predictFcn(fishertable)
```

The output *yfit* contains a class prediction for each data point. See the Command Window screenshot below:

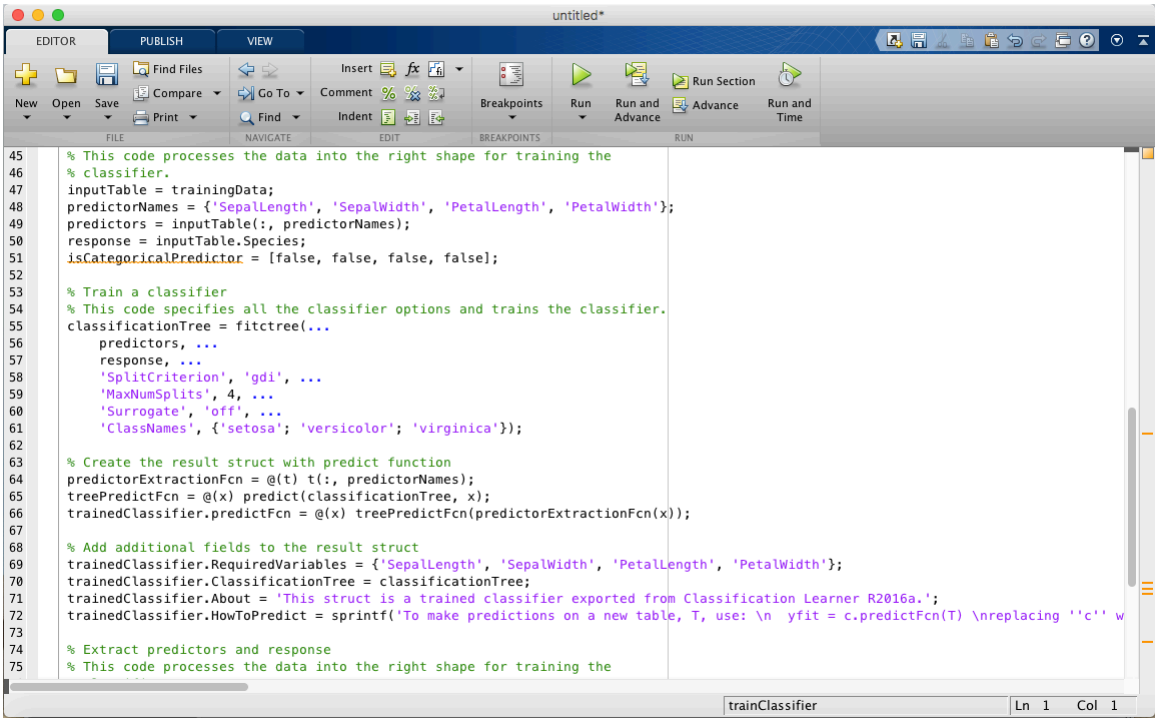


14. Generate code

If you want to automate training the same classifier with new data, or learn how to programmatically train classifiers, you can generate code from the app. To generate code for the best trained model, on the **Classification Learner** tab, in the **Export** section, click **Export Model > Generate Code**.

The app generates code from your model and displays the file in the **MATLAB Editor**.

See screenshot below for the generated code in Editor:



Mathworks provide many nice detailed examples, alternatively, you can refer to these links:

https://www.mathworks.com/help/stats/train-decision-trees-in-classification-learner-app.html

http://www.mathworks.com/help/stats/train-logistic-regression-classifiers-in-classification-learner-app.html