IE 6318 Data Mining and Analytics

Homework 2

Classification Using KNN & Decision Tree

Submitted by

Sarika Kshatriya

UTA ID:1001483313

[Sarika.kshatriya@mavs.uta.edu](mailto:Sarika.kshatriya@mavs.uta.edu)

1. KNN:

Implementation of the KNN classifiers to the IRIS dataset using different KNN parameter settings:

K = 3, 5, 7 for K-nearest neighbors, and r = 1, 2, 5 for the distance order of Minkowski Distance. For each parameter setting of K and r, performed the classification experiment using 5-fold cross validation.

**Matlab Code:**

Classification Using KNN main program

%Classification Using KNN

clear all;

close all;

clc

N=5;

load Iris.csv;

class= [1,2,3];

Y =[class];

X = Iris (:,1:4);

Y = Iris (:,5);

% set K = 3, 5, 7 for K-nearest neighbors

%and r = 1, 2, 5 for the distance order

for k = 3:2:7

for r = 1:2:5

[accuracy, confusionMatrix] = cm(N, k, r);

end

end

Function to Calculate Confusion Matrix and Accuracy:

% Calculate confusion matrix and accuracy with the given parameters

function [accuracy, confusionMatrix] = cm(N, k, r)

load Iris.csv;

class= [1,2,3];

Y =[class];

X = Iris (:,1:4);

Y = Iris (:,5);

accuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

class = unique(Y);

for i = 1:size(class,1)

foldSize = [];

temp = X(Y==class(i),:);

classIndex = find(Y==class(i));

classCount = numel(classIndex);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(classCount),:);

quotient = floor(classCount/N);

remainder = mod(classCount, N);

rng('shuffle');

extra = randsample(1:N,remainder);

foldSize = ones(N,1) \* quotient;

foldSize(extra) = foldSize(extra) + 1;

%Cumulative Sum

foldSizeCum = cumsum(foldSize);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCum(j-1)+1), ':' , num2str(foldSizeCum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 1:size(class,1)

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 1:size(class,1)

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

% prdict using KNN function

PredData = knn(trainData, trainDataLabel, testData, k, r);

%disp(k);

%Accuracy

accuracy = sum(PredData == testLabel)/length(testLabel);

accuracyMatrix = [accuracyMatrix; accuracy];

end

%Calculate accuracy in percentage

accuracy = mean(accuracyMatrix)\*100;

disp("Accuracy is:"+ accuracy);

%Calculate Confusion-matrix

confusionMatrix = confusionmat (testLabel,PredData);

disp("For k= "+k+" and r="+r+" with the Confusion Matrix - ");

disp(confusionMatrix);

end

Function for KNN Classifier

function PredData = knn(trainData, trainDataLabel, testData, k, r)

class = unique(trainDataLabel);

PredData = [];

for iC = 1:length(class)

cl = class(iC);

idx = find(trainDataLabel==cl);

eval(['dclass', num2str(iC), '=trainData(idx,:);']);

end

for i = 1:size(testData,1)

t = testData(i,:);

myKnnDist = [];

for iC = 1:length(class)

cl = class(iC);

A = eval(['dclass', num2str(cl)]);

s = size(A,1);

B=repmat(t,s,1);

%call minkowski distance functio

mdist= minkowskiDistance(A,B,r);

sortData = sort(mdist);

MyKnnDist = mode(sortData(1:k)); %find most repeated

myKnnDist = [myKnnDist;MyKnnDist,cl];

end

[vmin,imin] = min(myKnnDist(:,1));

p = myKnnDist(imin,2);

PredData = [PredData; p];

end

end

Function for Minkoski Distance:

function [MD] = minkowskiDistance(A,B,r)

MD=nthroot(sum((abs(A-B)).^r,2),r);

end

**Output of above program:Accuracy in percentage**

Accuracy is:95.3333

For k= 3 and r=1 with the Confusion Matrix -

10 0 0

0 9 1

0 0 10

Accuracy is:95.3333

For k= 3 and r=3 with the Confusion Matrix -

10 0 0

0 10 0

0 0 10

Accuracy is:96.6667

For k= 3 and r=5 with the Confusion Matrix -

10 0 0

0 10 0

0 0 10

Accuracy is:96

For k= 5 and r=1 with the Confusion Matrix -

10 0 0

0 8 2

0 1 9

Accuracy is:95.3333

For k= 5 and r=3 with the Confusion Matrix -

10 0 0

0 10 0

0 2 8

Accuracy is:94

For k= 5 and r=5 with the Confusion Matrix -

10 0 0

0 9 1

0 1 9

Accuracy is:92.6667

For k= 7 and r=1 with the Confusion Matrix -

10 0 0

0 10 0

0 0 10

Accuracy is:96

For k= 7 and r=3 with the Confusion Matrix -

10 0 0

0 9 1

0 0 10

Accuracy is:93.3333

For k= 7 and r=5 with the Confusion Matrix -

10 0 0

0 9 1

0 1 9

1. Report the classification accuracies for the 9 KNN parameter settings.

Output if use k=9

Accuracy is:91.3333

For k= 9 and r=1 with the Confusion Matrix -

10 0 0

0 9 1

0 3 7

Accuracy is:96

For k= 9 and r=3 with the Confusion Matrix -

10 0 0

0 10 0

0 2 8

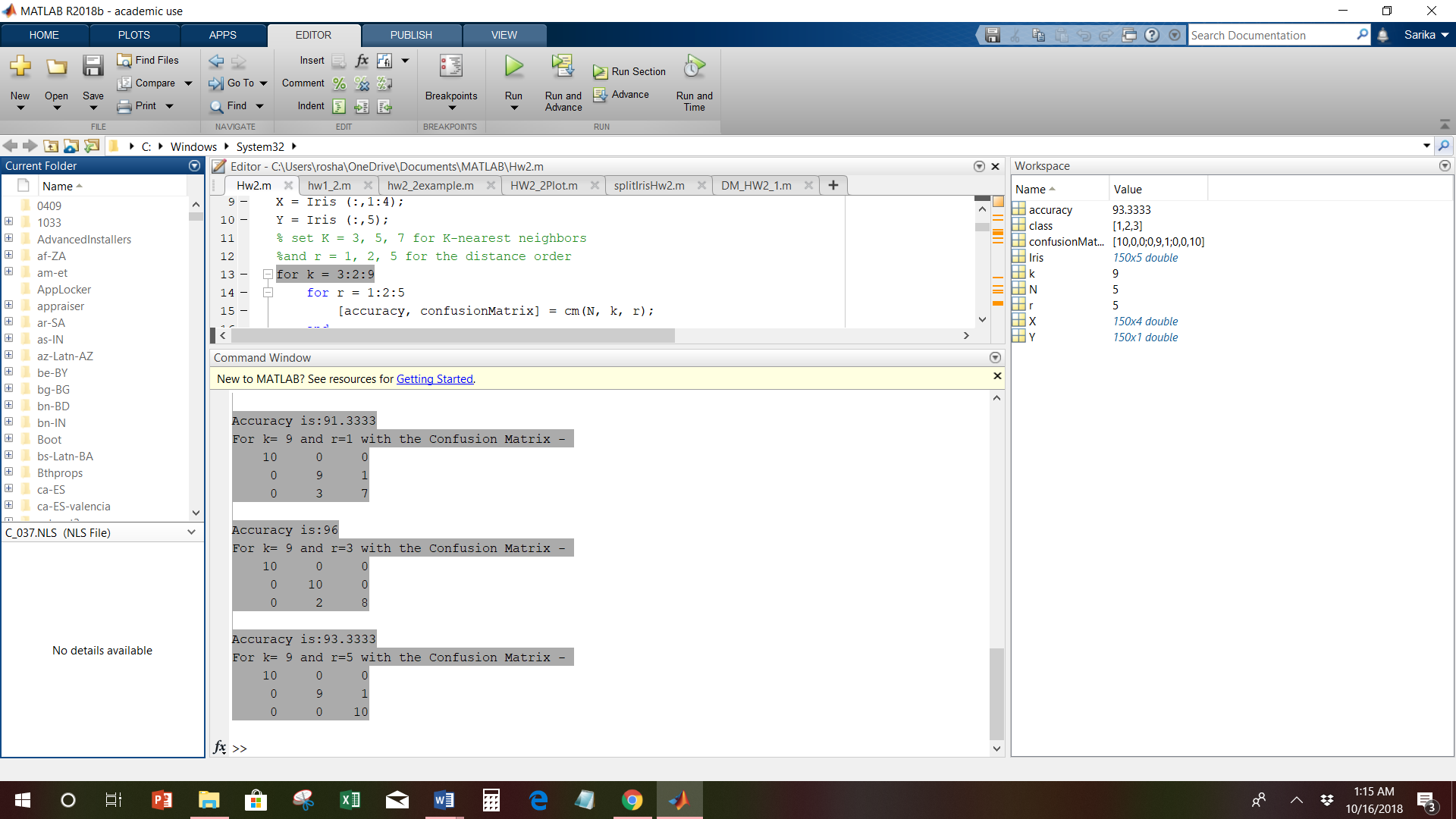
Accuracy is:93.3333

For k= 9 and r=5 with the Confusion Matrix -

10 0 0

0 9 1

0 0 10



1. find the best KNN parameter setting that generates the highest accuracy, report the corresponding confusion matrix for the best KNN parameter setting.

From the above Outputs the highest accuracy is 96.667 generated by k=3 and r=5

**Accuracy is:96.6667**

**For k= 3 and r=5 with the Confusion Matrix -**

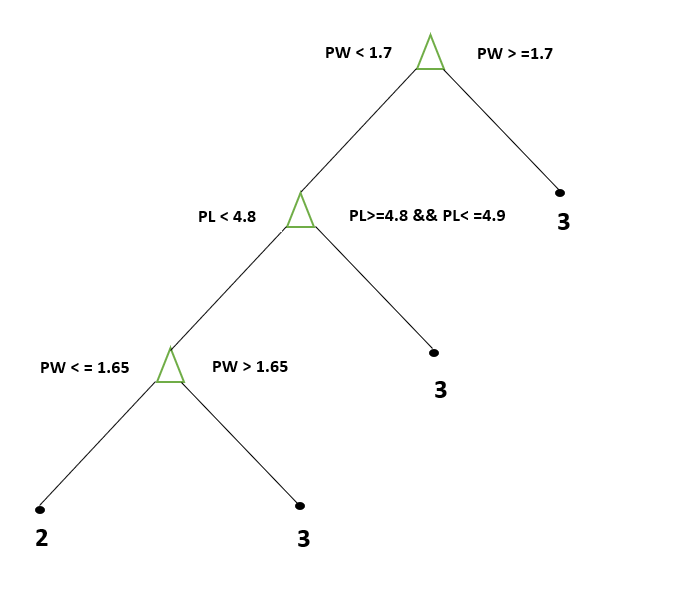
10 0 0

0 10 0

0 0 10

2.Decision Tree

Simple decision tree using the attributes of petal width(PW) and petal length (PL) to classify two types of iris flowers: Versicolor(2) and Virginica(3).



Classify the 100 iris samples of Versicolor or Virginica using 5 fold cross validation:

**Matlab Code:**

%Decision tree using the attributes of petal width and petal length

%to classify two types of iris flowers: Versicolor and Virginica.

clear all;

close all;

clc

N = 5; % N-fold cross validation

[accuracy, confusionMatrix] = cm(N);

%---------------------------------------------------------%

Function to Calculate Confusion Matrix and Accuracy:

% Calculate confusion matrix and accuracy for N fold

function [accuracy, confusionMatrix] = cm(N)

%N=5;

load Iris.csv;

class= [1,2,3];

X = Iris (51:end,3:4);

Y = Iris (51:end,5);

accuracyMatrix = []; confusionMatrix = [];

% 5 fold

%Divide the data into 5 sets in each class

%class = unique(Y);

for i = 2:3

foldSize = [];

temp = X(Y==class(i),:);

classIndex = find(Y==class(i));

classCount = numel(classIndex);

%Shuffle the dataset randomly.

rng('shuffle');

temp = temp(randperm(classCount),:);

quotient = floor(classCount/N);

remainder = mod(classCount, N);

rng('shuffle');

extra = randsample(1:N,remainder);

foldSize = ones(N,1) \* quotient;

foldSize(extra) = foldSize(extra) + 1;

%Cumulative Sum

foldSizeCum = cumsum(foldSize);

%Split the dataset into 5 groups

for j = 1:N

if j==1

eval(['nfold.class' , num2str(i) , '.fold', num2str(j) , ...

'=temp(', num2str(1), ':', num2str(foldSizeCum(1)), ',:);']);

else

eval(['nfold.class' , num2str(i), '.fold', num2str(j) , ...

'=temp(', num2str(foldSizeCum(j-1)+1), ':' , num2str(foldSizeCum(j)), ',:);']);

end

end

end

%For each unique group:

%Take the group as a hold out or test data set

%Take the remaining groups as a training data set

for i = 1:N

testData = []; testLabel = []; trainData = []; trainDataLabel = [];

%test data

for j = 2:3

eval(['testData = [testData; nfold.class' , num2str(j), '.fold', num2str(i), '];']);

eval(['testLabel = [testLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(i), ',1),1)];']);

end

%train data

for j = 2:3

for h = 1:N

if h ~= i

eval(['trainData = [trainData; nfold.class' , num2str(j), '.fold', num2str(h), '];']);

eval(['trainDataLabel = [trainDataLabel; j \* ones(size(nfold.class', num2str(j), '.fold', num2str(h), ',1),1)];']);

end

end

end

%disp(testLabel);

% disp(testData);

len=length(testData);

%disp(len);

%---------------------------------------------------------%

% prdict using decision tree function

PredData(k)= myFunction(trainData,trainDataLabel,testData)

end

%disp(PredData);

%Accuracy

accuracy = sum(PredData == testLabel)/length(testLabel);

%---------------------------------------------------------%

% accuracyMatrix for all folds

AccuracyMatrix = [];

eval(['confusionMatrix.fold', num2str(i), ' = confusionmat(testLabel, PredData);']);

if i == 1

confusionMatrix.allFolds = confusionMatrix.fold1;

else

eval(['confusionMatrix.allFolds = confusionMatrix.allFolds + confusionMatrix.fold', num2str(i),';']);

end

AccuracyMatrix = [AccuracyMatrix; accuracy];

disp(confusionMatrix.allFolds);

confusionchart(confusionMatrix.allFolds);

end

%---------------------------------------------------------%

%Scatter plot for testdata and predicted labels

figure

gscatter(testData(:,1),testData(:,2), PredData,'rg');

hold on;

xrange = [15 60];

yrange = [5 25];

%Assumed the boundary on Petal Length is 4.8, and

%the decision boundary on Petal Width is 1.7.

hline = refline([0 1.7]);

hline.Color = 'r';

hold on

line([4.8 4.8], [0 1.7])

title('Red:Versicolor Green:Virginica');

xlabel('Petal Length')

ylabel('Petal Width')

%---------------------------------------------------------%

%Calculate Confusion-matrix

confusionMatrix = confusionmat (testLabel,PredData);

figure

disp(confusionMatrix);

confusionchart(confusionMatrix);

%---------------------------------------------------------%

%call confusion statistics function which returns specificity,confusion matrix,accuracy and sensitivity form predicted data and test data labels.

stats=confusionmatStats(PredData,testLabel);

end

Decision tree classification function. The function has two inputs: petal width and petal length, and one output: type of flower Versicolor(2) or Virginica(3).

function flowerType =myFunction(trainData,trainDataLabel,testData)

flowerType =[];

len=length(trainData);

for k=1:len

trainPL=trainData(k:k,1);

trainPW=trainData(k:k,2);

if trainPW < 1.7

if trainPL < 4.8

if trainPW < 1.65

getFlowerType2=trainDataLabel(k);

elseif trainPW >= 1.65

getFlowerType3 = trainDataLabel(k)

end

elseif trainPL >= 4.8 && trainPL <=4.9

getFlowerType2=trainDataLabel(k);

else

getFlowerType3=trainDataLabel(k);

end

else

getFlowerType3=trainDataLabel(k);

end

end

testlen=length(testData);

for t=1:testlen

pl=testData(t:t,1);

pw=testData(t:t,2);

if pw < 1.7 %the decision boundary on Petal Width is 1.7

if pl< 4.8 %the binary decision boundary on Petal Length is 4.8

if pw< 1.65

f=getFlowerType2;

elseif pw>=1.65

f=getFlowerType3;

end

elseif pl>=4.8 && pl<=4.9

f=getFlowerType2;

else

f=getFlowerType3;

end

else

f=getFlowerType3;

end

flowerType = [flowerType; f];

end

end

ConfusionmatStats function:

function stats = confusionmatStats(group,grouphat)

% INPUT

% group = true class labels

% grouphat = predicted class labels

%

% OR INPUT

% stats = confusionmatStats(group);

% group = confusion matrix from matlab function (confusionmat)

%

% OUTPUT

% stats is a structure array

% stats.confusionMat

% Predicted Classes

% p' n'

% \_\_\_|\_\_\_\_\_|\_\_\_\_\_|

% Actual p | | |

% Classes n | | |

%

%

% TP: true positive, TN: true negative,

% FP: false positive, FN: false negative

%

field1 = 'confusionMatrix';

if nargin < 2

value1 = group;

else

[value1,gorder] = confusionmat(group,grouphat);

end

disp(field1);

disp(confusionmat(group,grouphat));

numOfClasses = size(value1,1);

totalSamples = sum(sum(value1));

[TP,TN,FP,FN,accuracy,sensitivity,specificity] = deal(zeros(numOfClasses,1));

for class = 1:numOfClasses

TP(class) = value1(class,class);

tempMat = value1;

tempMat(:,class) = []; % remove column

tempMat(class,:) = []; % remove row

TN(class) = sum(sum(tempMat));

FP(class) = sum(value1(:,class))-TP(class);

FN(class) = sum(value1(class,:))-TP(class);

end

for class = 1:numOfClasses

accuracy(class) = (TP(class) + TN(class)) / totalSamples;

sensitivity(class) = TP(class) / (TP(class) + FN(class));

specificity(class) = TN(class) / (FP(class) + TN(class));

% precision(class) = TP(class) / (TP(class) + FP(class));

% f\_score(class) = 2\*TP(class)/(2\*TP(class) + FP(class) + FN(class));

end

field2 = 'accuracy'; value2 = accuracy;

disp(field2);

disp(value2);

field3 = 'sensitivity'; value3 = sensitivity;

disp(field3);

disp(value3);

field4 = 'specificity'; value4 = specificity;

disp(field4);

disp(value4);

stats = struct(field1,value1,field2,value2,field3,value3,field4,value4);

end

Output of above Program is:

Confusion Matrix :

9 1

0 10

Confusion Matrix Chart:



accuracy

0.9500

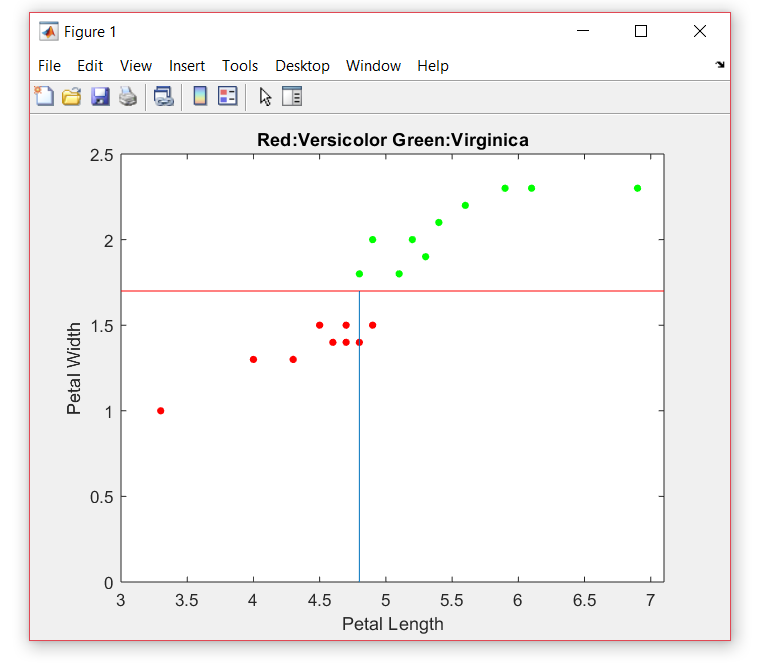
sensitivity

1.0000

specificity

0.9091

Scatter plot for test data and predicted labels(20 samples):



Confusion Matrix for all folds:

9 1

0 10

18 2

0 20

28 2

0 30

37 3

0 40

47 3

0 50

47 3

0 50

Confusion Matrix chart for all folds:



Scatter Plot for all 100 Samples:

Plot using the attributes of petal width and petal length to classify two types of iris flowers: Versicolor and Virginica, as shown below. Assume the binary decision boundary on Petal Length is 4.8, and the decision boundary on Petal Width is 1.7.

**Matlab Code:**

clear all;

close all;

clc

load Iris.csv;

%Scatter plot for petal width and petal length

%to classify two types of iris flowers: Versicolor and Virginica

Class= [1,2,3];

X = Iris (51:end,1:4);

Y = Iris (51:end,5);

Petal\_length = Iris (51:end,3);

Petal\_width = Iris (51:end,4);

gscatter(Petal\_length,Petal\_width, Y,'rg');

hold on

xrange = [15 60];

yrange = [5 25];

%Assumed the boundary on Petal Length is 4.8, and

%the decision boundary on Petal Width is 1.7.

hline = refline([0 1.8]);

hline.Color = 'r';

hold on

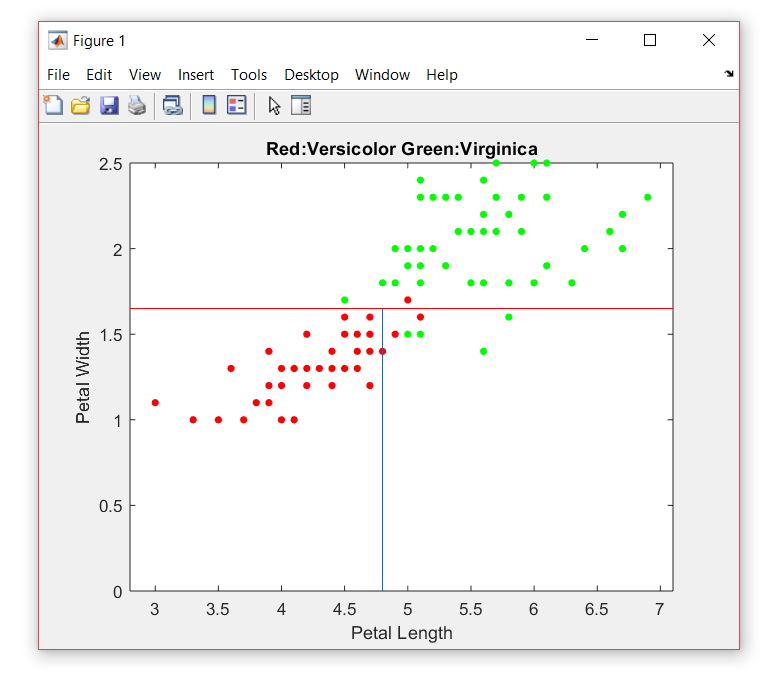
line([4.8 4.8], [0 1.7])

title('Red:Versicolor Green:Virginica');

xlabel('Petal Length')

ylabel('Petal Width')

N = size(Iris,1);



References:

<https://in.mathworks.com/help/> <https://in.mathworks.com/help/stats/examples/classification.html>

<https://machinelearningmastery.com/k-fold-cross-validation/>

<https://in.mathworks.com/matlabcentral/fileexchange/46035-confusionmatstats-group-grouphat>

<https://stackoverflow.com/questions/2314850/help-understanding-cross-validation-and-decision-trees>

Lecture3\_Classification Basics