Garrett Scholtes
Intelligent data Analysis
Homework #2 (Classification trees with banknote dataset)
Due Date: Tuesday, Sept. 27th, 9PM

All code used to complete this assignment is located at the end of this document

1. (10) Load the dataset in MATLAB as table named d1 and compute the covariance matrix of the data by using the command: "cm = cov(d1);". Write all the information that you can infer by interpreting the values in the covariance matrix cm.

In the covariance matrix, the (*i*, *j*)-th element correspond to the covariance between the attributes *i* and *j* in the *d1* table (i.e., covariance between all pairs of columns), that is, whether there is some dependence or relationship between the variables. Whether or not the covariance is positive will tell us if the attributes tend to be directly proportional (+) or inversely proportional (-)

For example, glancing at the scatter plots we made for the problems below, we can informally visualize whether or not the attributes have a direct or inverse relationship. The covariance values make this notion formal. Continue to the below plots for more details.

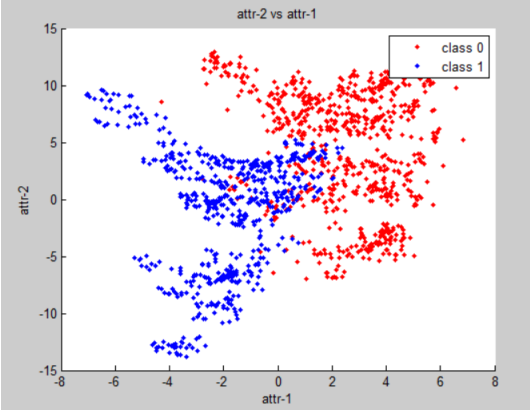
```
cm =

    8.0813    4.4051   -4.6663    1.6533   -1.0243
    4.4051   34.4457  -19.9051   -6.4900   -1.2974
   -4.6663  -19.9051   18.5764    2.8872    0.3340
    1.6533   -6.4900    2.8872    4.4143   -0.0245
   -1.0243   -1.2974    0.3340   -0.0245    0.2471
```

2. (6) Do a scatter plot of attribute-1 and attribute-2. Use different colors to mark the points from the two different classes. Write your interpretation of the separability of the two classes using attribute1 and attribute2, and also any other insights that you can obtain from this scatter plot.

Looking at the below figure, it appears that the classes are not linearly separable based on attributes 1 and 2 alone.

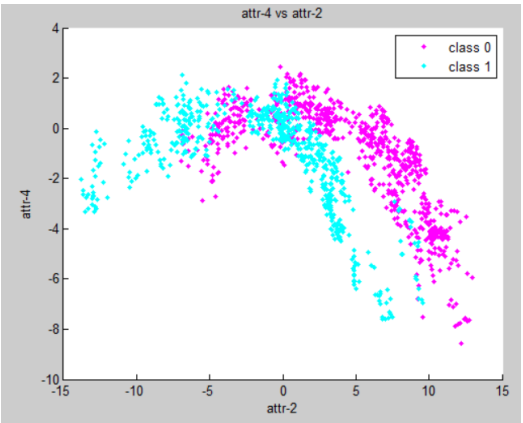
Going back to the covariance concept, it visually appears that class 1 tends to the lower left corner and class 0 tends to the upper right, making them have a direct relationship. This is consistent with the covariance(attr-1, attr-2) = 4.4051 (see marix above), which is in fact positive.



3. (6) Repeat #2 above for attribute2 and attribute4.

Attributes 2 and 4 also do not appear to be linearly separable by themselves.

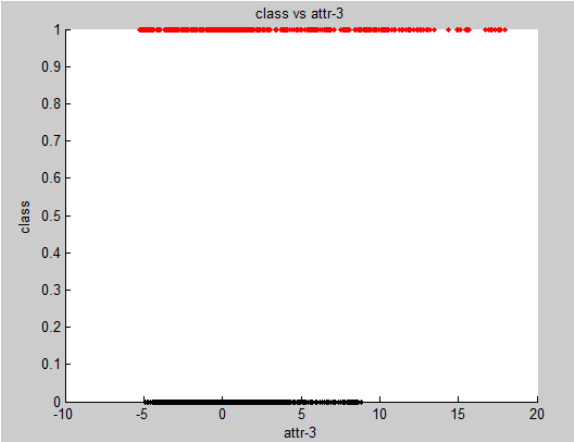
Noting that most of these data points occur at attr-2 > 0, it appears that the attributes beyond that point are *roughly* inversely proportional. This is verifiable with covariance(attr-2, attr-4) = -6.4900, which is in fact negative.



4. (6) Repeat #2 above for attribute3 and class label (attribute5).

Classes do not appear to be linearly separable in this case. **Note:** in this instance (and in problem #5), unlike the above cases, by linearly separable we mean there is no **vertical** line that separates the classes We only care about separating on attribute-3, not the class labels (which are discrete and presumably not known in advance in the real data).

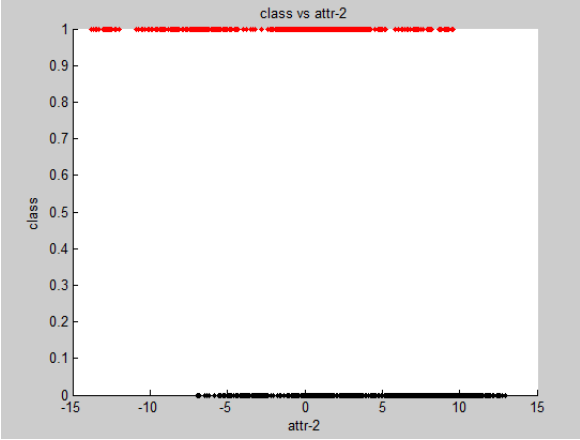
Data in class 1 tend to be more positive (hence covariance(attr-3, class) = 0.3340)



5. (6) Repeat #2 above for attribute2 and class label (attribute5).

Also not linearly separable (by vertical line)

Data in class 1 tend to be more negative than in class 2 (covariance(attr-2, class) = -1.2974)



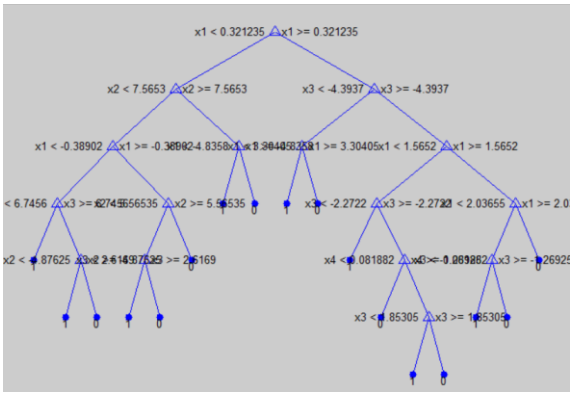
Note: With regards to all of the above data not being linearly separable with 1 or 2 attributes, it should be noted that this does not necessarily say anything about their separability in higher dimensional space.

6. (6) From the dataset randomly select 1000 records for testing and leave the rest for testing. These selections MUST be random. Use a random number generator function to generate 1000 random numbers between 1 and 1372, and then choose records with these indices for training. Show the code used for selecting the training set of records.

See code at bottom of document (look for the comment that says "Part 6")

7. (20) Use the training set to learn decision trees (use fitctree command) by varying the parameter that controls the minimum number of records in a leaf node. For this parameter use the values of 5, 25, and 50. For each generated tree show its graph schematic (print it from Matlab), and report: (i) number of leaf nodes, (ii) the longest path from the root to a leaf node, its population, and its class purity.

For each of the trees below, I am reporting on the *leftmost* leaf node at the *deepest* level for population and class purity (these trees have multiple leaf nodes that are on a longest path)



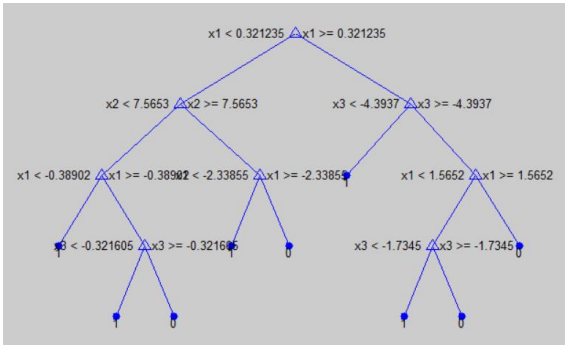
Min records at leaf = 5

Leaf nodes = 17

Depth = 6

Population = 12

Purity = 91.7%



Min records at leaf = 25

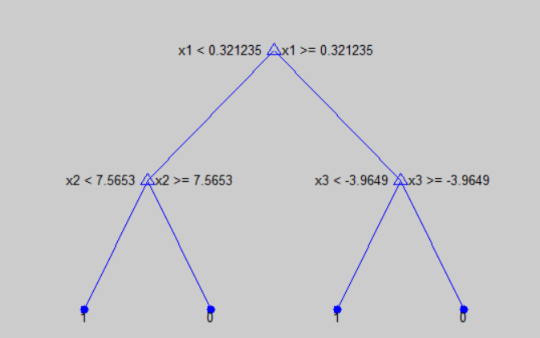
Leaf nodes = 9

Depth = 4

Population = 30

Purity = 93.3%

(continue to next page for 3rd tree)



Min records at leaf = 25

Leaf nodes = 4

Depth = 2

Population = 402

Purity = 93.0%

Note: the purities for the selected leaf nodes are not necessarily indicative to the performance of the tree. Other nodes had much less impressive purities (though some of those nodes also had smaller populations).
That being said, the confusion matricies below should provide a much better summary of the performance of each tree.

8. (40) Test each of the trees generated in #7 with the test dataset and generate (and show) the confusion matrix. For each tree compute the accuracy, precision, and recall values. Give an interpretation of these performance numbers individually and also in the context of changes to these numbers as the size of the decision tree changes.

Tree with min records = 5
Very performant. Note: 100% precision implies no false positives
Accuracy = 98.66%
Precision = 100%
Recall = 97.54%

confusion_5 =

198 5

0 169

Tree with min records = 25
Less, but still highly performant (with less depth, we should also expect better *time* to compute)
Accuracy = 92.74%
Precision = 94.44%
Recall = 92.12%

confusion_25 =

187 16

11 158

Tree with min records = 50
Decently performant
Accuracy = 89.25%
Precision = 90.55%
Recall = 89.66%

confusion_50 =

182 21

19 150

The accuracy (and precision, recall, etc) tend to go down as the decision tree size goes down (though computing and using them is likely faster – i.e., cheaper). This makes sense, because we expect each question (node) to increase the amount of information extracted from the data set (hence why we use metrics like *information gain* to analyze the utility of a question node). More questions in a tree should therefore yield more information (that is, up until a point – eventually there will be no more separability of data).

SOURCE CODE

```
% Garrett Scholtes
% 2016-09-24
% Homework #2
% Intelligent data analysis
%
% NOTE: Created with MATLAB R2013a
%       Using `ClassificationTree.fit` in lieu of `fitctree`

clear all;
close all;
% Close tree view windows
hiddenfigs = findall(0,'Type','figure', '-not', 'HandleVisibility', 'on');
close(hiddenfigs);

% Seed the random number generator for convenience
% Can be removed if desired
rng(2016);

% Part 1
% Load data + covariance matrix
d1 = xlsread('data_banknote_authentication.xlsx');

cm = cov(d1);

% Part 2
% Scatter plot attr-1 attr-2
D10 = d1(d1(:,5)==0,:);
D11 = d1(d1(:,5)==1,:);
scatter(D10(:,1),D10(:,2),'r.');
hold on;
scatter(D11(:,1),D11(:,2),'b.');
title('attr-2 vs attr-1');
xlabel('attr-1');
ylabel('attr-2');
legend('class 0', 'class 1');

% Part 3
% Scatter plot attr-1 attr-2
figure;
D20 = d1(d1(:,5)==0,:);
D21 = d1(d1(:,5)==1,:);
scatter(D20(:,2),D20(:,4),'m.');
hold on;
scatter(D21(:,2),D21(:,4),'c.');
title('attr-4 vs attr-2');
xlabel('attr-2');
ylabel('attr-4');
legend('class 0', 'class 1');

% Part 4
```

```
% Scatter plot attr-3 class

figure;
D30 = d1(d1(:,5)==0,:);
D31 = d1(d1(:,5)==1,:);
scatter(D30(:,3),D30(:,5),'k.');
hold on;
scatter(D31(:,3),D31(:,5),'r.');
title('class vs attr-3');
xlabel('attr-3');
ylabel('class');

% Part 5
% Scatter plot attr-2 class

figure;
D40 = d1(d1(:,5)==0,:);
D41 = d1(d1(:,5)==1,:);
scatter(D40(:,2),D40(:,5),'k.');
hold on;
scatter(D41(:,2),D41(:,5),'r.');
title('class vs attr-2');
xlabel('attr-2');
ylabel('class');

% close all;
% Part 6
% Select a training set of 1000 randomly selected records

% (1) Shuffle the index list [1, 2, 3, ... , size(d1)]
permutation = randperm(size(d1, 1));
% (2) The first 1000 entries of those shuffles produces
%     a random selection of indecies for the training set.
%     The last entries of the shuffle represent the test set.
TRAIN_SIZE = 1000;
train_set = d1(permutation(:,1:TRAIN_SIZE),:);
test_set = d1(permutation(:,TRAIN_SIZE+1:end),:);

% Part 7
% Create classification trees using the training data, while
% experimenting with different pruning levels.
train_attr = train_set(:,1:4);
train_class = train_set(:,5);

tree5 = ClassificationTree.fit(train_attr, train_class, 'MinLeaf', 5);
tree25 = ClassificationTree.fit(train_attr, train_class, 'MinLeaf', 25);
tree50 = ClassificationTree.fit(train_attr, train_class, 'MinLeaf', 50);

view(tree5, 'Mode', 'Graph');
view(tree25, 'Mode', 'Graph');
view(tree50, 'Mode', 'Graph');

% Part 8
```

```
% Testing the trees and confusion matricies report
test_attr = test_set(:,1:4);
test_class = test_set(:,5);

test_results_5 = tree5.predict(test_attr);
confusion_5 = confusionmat(test_class, test_results_5);

test_results_25 = tree25.predict(test_attr);
confusion_25 = confusionmat(test_class, test_results_25);

test_results_50 = tree50.predict(test_attr);
confusion_50 = confusionmat(test_class, test_results_50);
```