**Assignment 1**

Zezhou Li

10405265

1. In this assignment, I am going to use Matlab to train the data and get the classification decision tree and then implement the decision tree to the test data and get the classification results, finally compare them with the actual results to see the accuracy.

In Matlab, there is already a function called *fitctree()* which receives parameters such as the attributes data and the classification results from the training data. Then it gives us the classification decision tree after training the data. We can then use function *predict()* to classify the test data with the decision tree.

First, import the data from the given file, since the format of the file is ‘.*csv’*, thus I use an open source function to import them in Matlab without importing by hand, everything I will do will be automatically done by the program, thus it will be useful in the future.

Since the import function is an open source function, I am not going to show it here, I will put it into the attached file.

Here is the code importing the data:

|  |
| --- |
| %--- Read in training and test data ---%  TrainingFileName = 'train.csv';  TestFileName = 'test.csv';  ResultFileName = 'result.csv';  TrainingData = ReadinData(TrainingFileName);  TestData = ReadinData(TestFileName);  ResultData = ReadinData(ResultFileName); |

After importing the files, I have to transform some of the vectors into the type of *Double* since two vectors of attributes *‘Age’* and *‘Embarked’* are string vectors. Here for the *‘Age’* attribute, if the element is *‘male’*, I will set it to 1, if it is *‘female’*, I will then set it to 0. I put this function into a file with the name *‘change2num‘*.

Here is the code:

|  |
| --- |
| function Y = change2num(X,attr)  t = size(attr,2);  if t==2  for i=1:size(X,1)  if strcmp(X{i},attr{1})  X{i} = 1; % male = 1  else if strcmp(X{i},attr{2})  X{i} = 0; % female = 0  else  X{i} = NaN;  end  end  end  else  for i=1:size(X,1)  if strcmp(X{i},attr{1})  X{i} = 1; % S = 1  else if strcmp(X{i},attr{2})  X{i} = 2; % C = 2  else if strcmp(X{i},attr{3})  X{i} = 3; % Q = 3  else  X{i} = NaN;  end  end  end  end  end  Y = cell2mat(X);  end |

Since there are three files of data need to read, thus I wrote a function to do the importing and transforming at the same time.

Here is the code:

|  |
| --- |
| function Y = ReadinData(Filename)  data = csvimport(Filename);  Y = [];  N = size(data,2);  %--- Seperate training data into vectors ---%  if N == 8  Y = [Y,cell2mat(data(2:end,1))];  Y = [Y,cell2mat(data(2:end,2))];  Y = [Y,change2num(data(2:end,3),{'male','female'})];  Y = [Y,cell2mat\_preprocess(data(2:end,4))];  Y = [Y,cell2mat(data(2:end,5))];  Y = [Y,cell2mat(data(2:end,6))];  Y = [Y,cell2mat(data(2:end,7))];  Y = [Y,change2num(data(2:end,8),{'S','C','Q'})];  end  if N == 7  Y = [Y,cell2mat(data(2:end,1))];  Y = [Y,change2num(data(2:end,2),{'male','female'})];  Y = [Y,cell2mat\_preprocess(data(2:end,3))];  Y = [Y,cell2mat(data(2:end,4))];  Y = [Y,cell2mat(data(2:end,5))];  Y = [Y,cell2mat\_preprocess(data(2:end,6))];  Y = [Y,change2num(data(2:end,7),{'S','C','Q'})];  end  if N == 2  Y = cell2mat(data(2:end,2));  end    end |

In the code, there is a function called *‘cell2mat\_preprocess’*, it is to deal with the situation that the element in the vector is empty and could not be transformed to type Double using Matlab function *‘cell2mat’*.

Here is the code:

|  |
| --- |
| function Y = cell2mat\_preprocess(X)  ix = cellfun(@isempty,X);  X(ix)={'nan'};  Y = cellfun(@str2num,X);  end |

After normalizing the data, then it is time to fit them into function ‘*fitctree’* and get the classification decision tree.

Here is the code:

|  |
| --- |
| %--- Use fitctree function to train the decision tree ---%  %--- View the tree structure ---%  %--- Calculate resubstitution error ---%  DTTree = fitctree(TrainingData(:,2:end),TrainingData(:,1));  view(DTTree,'Mode','graph');  resuberror = resubLoss(DTTree);  disp(['resubstitution error before pruning is:',num2str(resuberror)]); |

Here besides function ‘*fitctree’*, I also use function ‘*view’* to see the tree, then it gives me the first result of my decision tree. Here is the plot of the first decision tree:

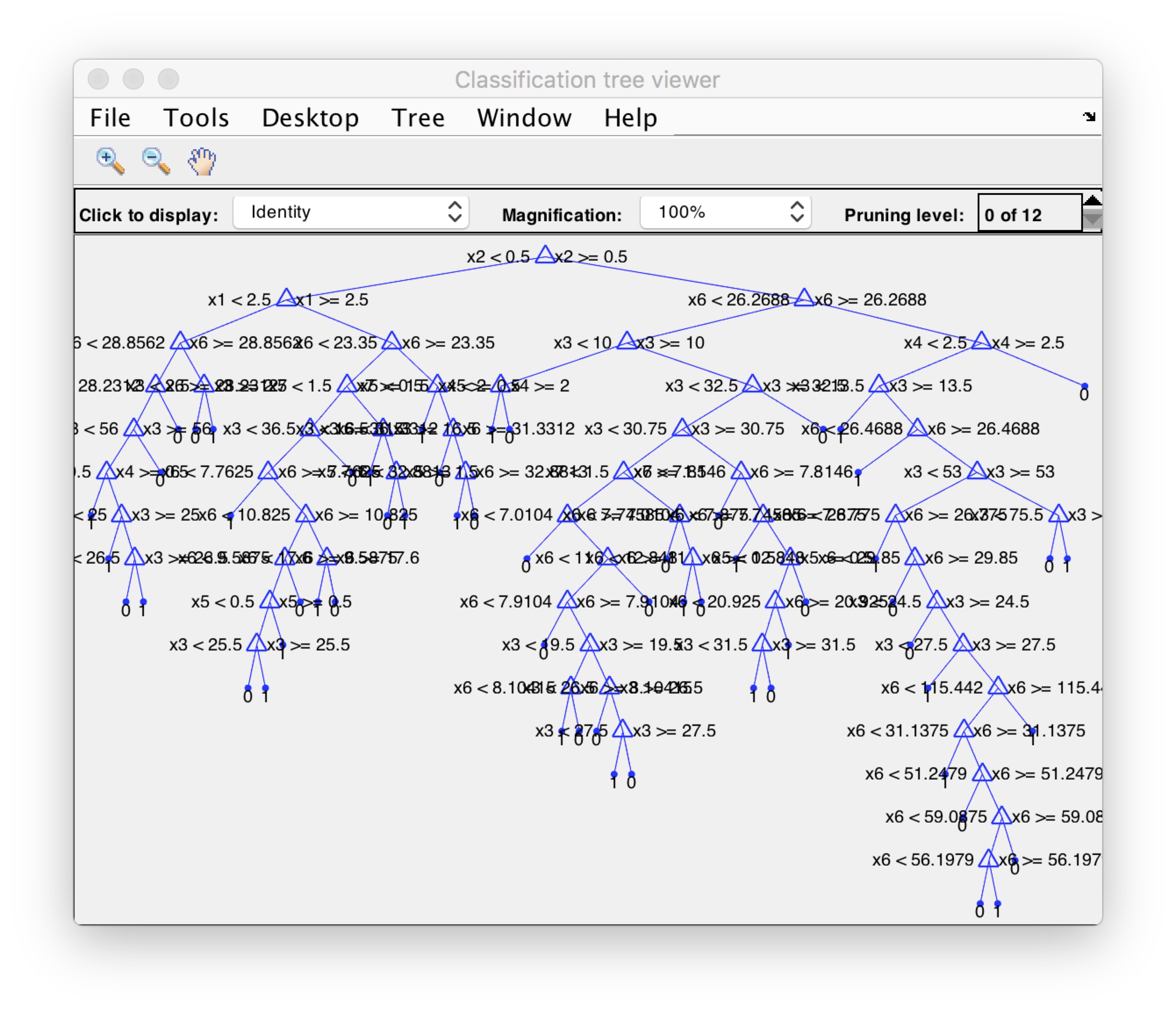


Figure 1. First Decision Tree

We can see that this tree is extremely big, it has so many branches which means that it may overfits the data later when I do the test. Here I use this tree to predict the result of the test data to see its accuracy.

Since prediction is used whenever I correct the decision tree, for convenience, I wrote a function file to realize it.

Here is the code:

|  |
| --- |
| function Y = makePrediction(DTTreeSet, TestSet)  Y = [];  N = size(DTTreeSet,1);  for i = 1:N  Y = [Y,predict(DTTreeSet{i}, TestSet)];  end  end |

After comparing with the actual result, I got the first prediction accuracy: 82.25%.

This accuracy is not good enough and I guess that it most likely because of the decision tree I made was too large which needs to be pruned. As a result, the second step I am going to do is to prune the tree and see the accuracy.

There is also a function ‘prune’ in Matlab which is used to prune the decision by the level you set in the function. Here as I read in the Matlab official document, the usual level is set to be 10. So I choose level 10 at the first time pruning the tree.

Here is the code:

|  |
| --- |
| %--- Prune the decision tree and get an optimal decision tree ---%  %--- View the tree structure ---%  %--- Calculate resubstitution error ---%  DTTreePrune = prune(DTTree,'Level',10);  view(DTTreePrune,'Mode','graph');  resuberror\_prune = resubLoss(DTTreePrune);  disp(['resubstitution error after pruning is:',num2str(resuberror\_prune)]); |

I also view the pruned decision tree after that:

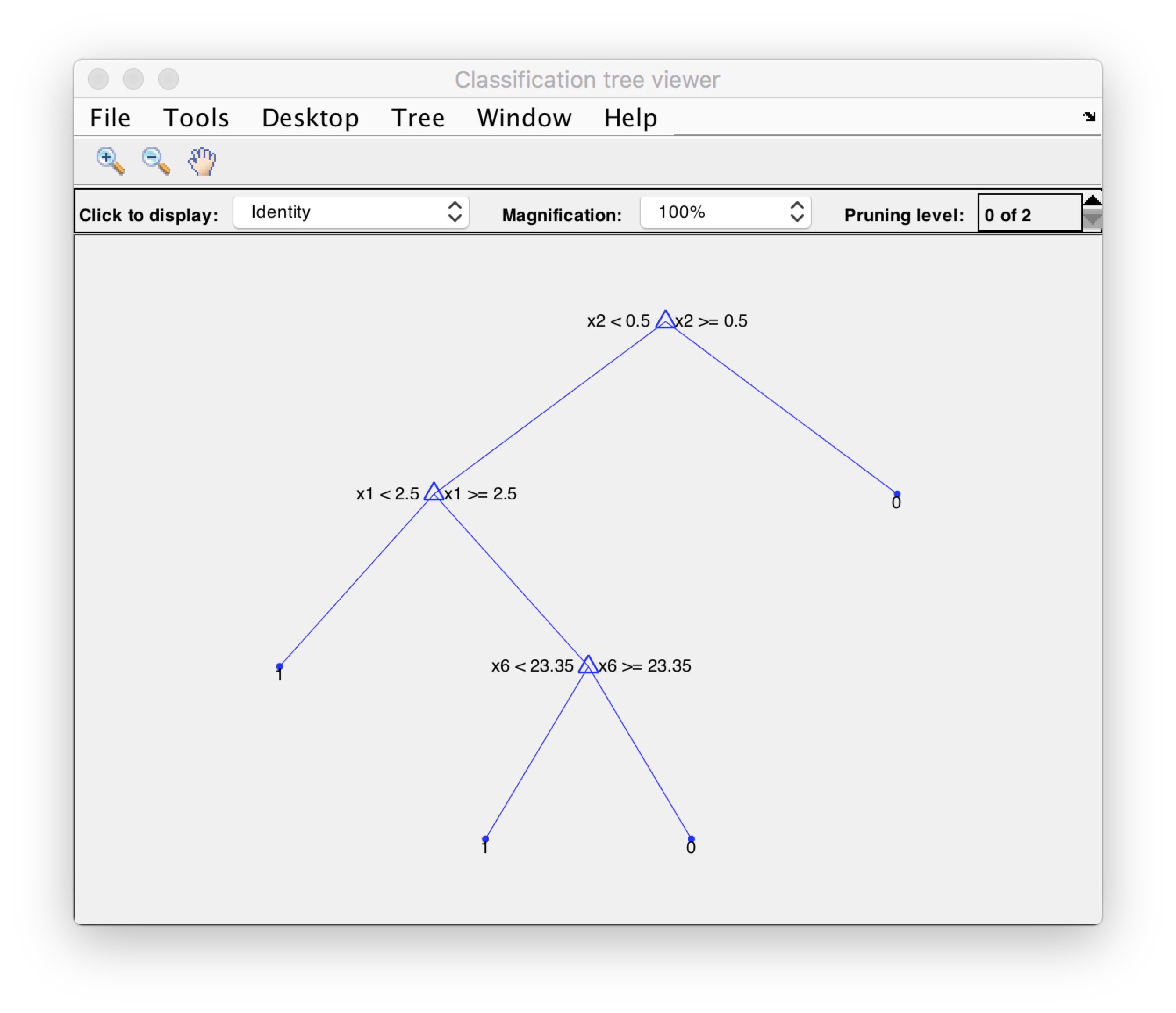


Figure 2. Pruned Tree with level 10

From the plot we can see that the tree is pruned so much and left only few branches, then I want to know how well it can predict the test data, so again I use the function like before to see the accuracy, this time it is: 98.56%.

Awesome!

After pruning the complicated decision tree, with so few branches, I can get a very high accuracy as 98.56%.

After achieving this target, can we stop? Maybe not!

I am going to further prune the tree to see if a much simpler tree can do better? Thus I set the prune level to be 11. Let’s see the pruned decision tree now:

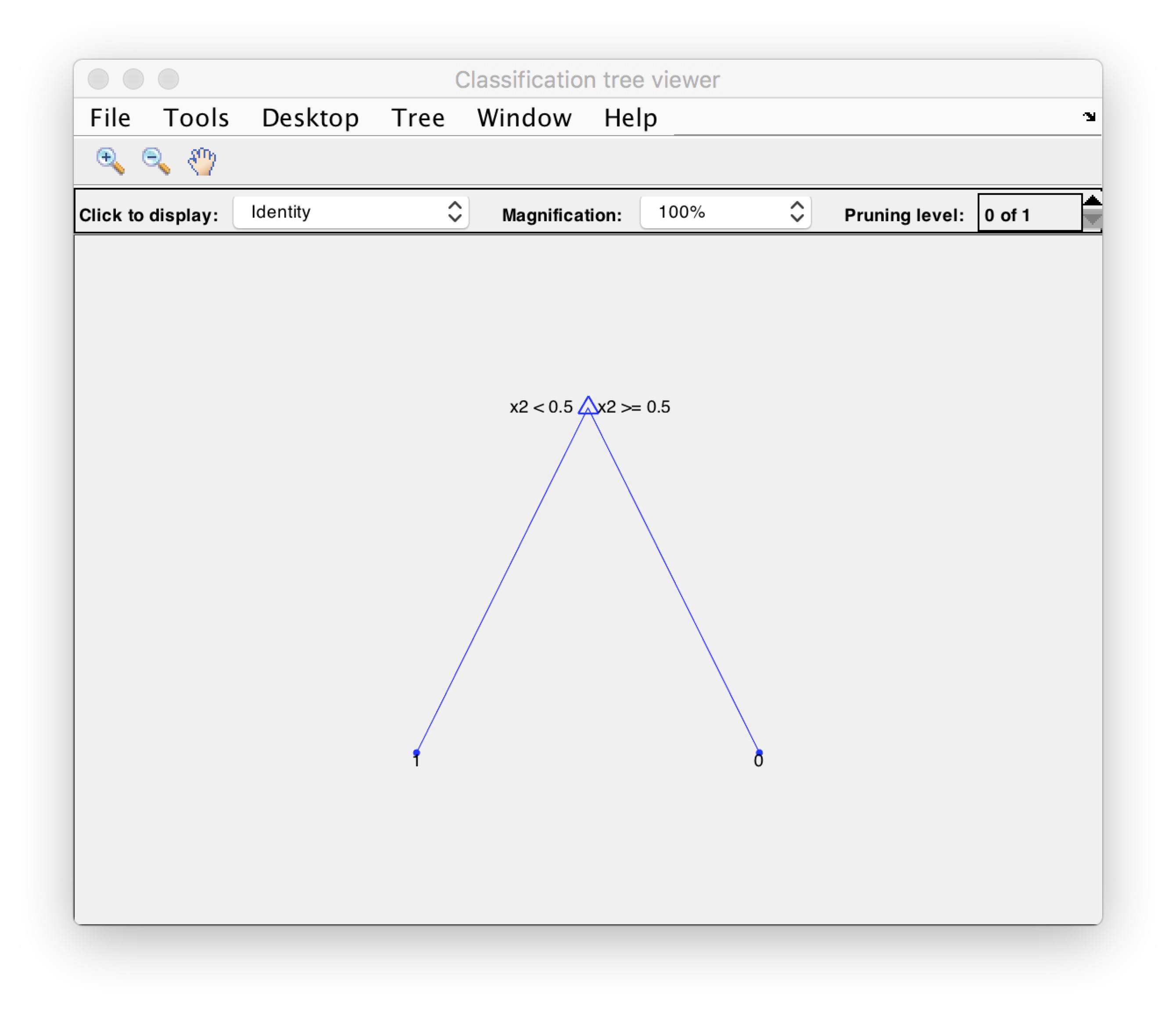


Figure 3. Pruned Tree with level 11

Now we can see the tree is very simple and cannot be further pruned. It only use the root attribute to classify the data. Let’s see now the accuracy: 100%!

So, it got every predicted value the same as the actual result. Thus, I suppose that the actual result is derived from the second attribute.