ESE589 Course Project III

Decision Tree Induction Algorithm  
December 5, 2019  
Professor Alex Doboli  
Kenneth Ng (110211727)  
Kurt Butler (109587387)

# Abstract

In this project, we implement the Decision Tree Induction algorithm using Python. We discuss our implementation of the algorithm, and we evaluate its performance using the benchmark data sets supplied by the UCI Machine Learning Repository [1]. Our implementation of the algorithm was successful across 10 data sets and three information metrics. We consider the performance of our algorithm in each metric to solutions that implement support vector machine (SVM) or artificial neural network (ANN) technologies using standard open-source libraries. We assess the strengths and weaknesses of each algorithm, as observed in our experiments, and we propose considerations for further study and future implementations.

Contents

[Abstract 1](#_Toc26439005)

[Introduction 3](#_Toc26439006)

[Design of the Software 3](#_Toc26439007)

[Figure 1. Example Decision Tree 3](#_Toc26439008)

[Information Gain Measure 4](#_Toc26439009)

[Gain Ratio Measure 4](#_Toc26439010)

[Gini Index Measure 4](#_Toc26439011)

[Algorithm Implementation 4](#_Toc26439012)

[Table 1: All Electronics Customer Database 5](#_Toc26439013)

[Figure 2: First Iteration of Decision Tree Generation for *AllElectronics* Database 5](#_Toc26439014)

[Figure 3: Complete Decision Tree for *AllElectronics* Database 6](#_Toc26439015)

[Experiments 6](#_Toc26439016)

[Table 2: Gain Ratio DTI Test Statistics 7](#_Toc26439017)

[Table 3: Info Gain DTI Test Statistics 7](#_Toc26439018)

[Table 4: Gini Index DTI Test Statistics 7](#_Toc26439019)

[Table 5: SVM Test Statistics 8](#_Toc26439020)

[Table 6: Artificial Neural Network Test Statistics 8](#_Toc26439021)

[Discussion 8](#_Toc26439022)

[Plot 1: Execution Time of DTI Algorithms 9](#_Toc26439023)

[Plot 2: Algorithm Accuracy Measurements 10](#_Toc26439024)

[Plot 3: Algorithm Sensitivity Measurements 10](#_Toc26439025)

[Plot 4: Algorithm Specificity Measurements 11](#_Toc26439026)

[Plot 5: DTI Algorithms Normalized Time vs No. Attributes 12](#_Toc26439027)

[Plot 6: DTI Algorithms Memory Usage 12](#_Toc26439028)

[Conclusion 13](#_Toc26439029)

[References 13](#_Toc26439030)

# Introduction

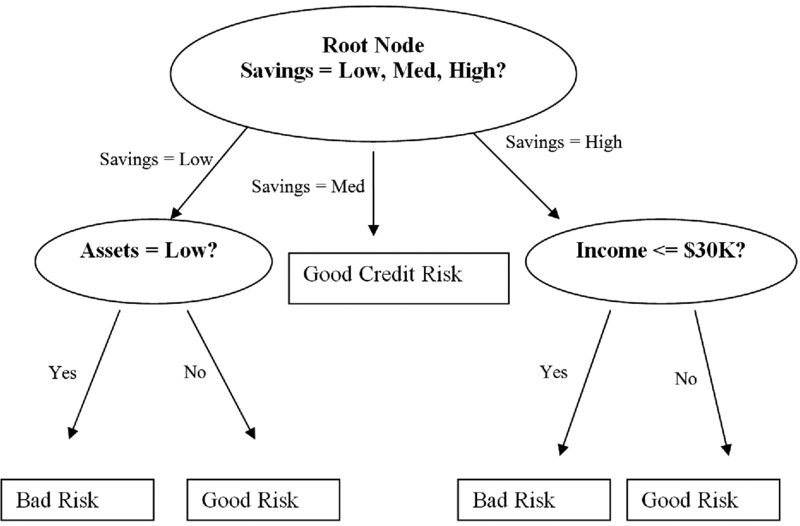
Most machine learning problems fall into one of three categories: classification, clustering and regression. While the latter two support a wealth of applications across science and engineering, neither as are well-recognized as the classification problem. Indeed, many of the most famous problems in machine learning, such as facial recognition, consumer analysis and voice control technology admit solutions that are conveniently expressed in the language of classification problems.

Often it is desirable to have simple rules for classifying data, implemented in such a way that is human interpretable. The Decision Tree Induction algorithm aims to address these concerns by recursively constructing a flowchart for classification. Given a data point vector, the algorithm sequentially observes each attribute of the vector to determine its classification in a deterministic manner. The advantages of this approach include its simplicity, efficiency and interpretability after the classifier is trained sufficiently.

# Design of the Software

The Decision Tree Algorithm is used for classification by making decisions based on different criteria provided by the data. To visualize the algorithm the following example will be taken from Rajesh S. Brid’s article on decision trees. Decision trees will have a tree data structure with nodes that contain either an attribute or a classification result. In Figure 1, each bubble contains an attribute: {savings, assets, income} with the splitting criteria {[low, med, high], [low?], [$30K?]}. Each rectangle contains a classification result {Good Risk, Bad Risk}.

### Figure 1. Example Decision Tree[3]



In choosing which attribute to use as the next node, different selection metrics can be used to determine the most useful attribute. There are two types of attributes: discrete and continuous. The discrete attributes have distinct splits and can be grouped or left alone. The continuous attributes have no defined split; therefore, the algorithm should determine the most optimal split that gives the highest metric and use the split as a comparison with the other attributes. For example, referencing Figure 1, the income attribute would determine that $30k would be the highest metric. In our implementation, we only use a binary split for continuous data using a threshold value, but in principle you can discretize the data set by partitioning the real number line into an arbitrary number of intervals. The three metrics that are used for comparison are information gain, Gini index, and gain ratio and we briefly sketch their usage below.

## Information Gain Measure

Information gain minimizes the information needed to classify a given set and reflects the least randomness [2]. The expected information needed for classification is given by the entropy of the data set after partitioning by class. We consider a repartition of the data set and compute it’s entropy by a weighted sum of entropies of each partition (where the weight is the percentile size of the partition). The information gain is then defined to be the difference of the original entropy with the repartitioned entropy. The repartitioning is facilitated by a branching condition at a given node, and we maximize the gain to produce the best classification.

## Gain Ratio Measure

The gain ratio method computes the entropy of a splitting condition by considering the magnitude of the Info Gain after being normalized by the so-called *split info*. The split info is the blind entropy of the partitioning given by the branching condition, without consideration of the classifications. At ratios near 0, the measure becomes unstable, so we only consider gain ratio splits that exceed the average of previously examined tests [2].

## Gini Index Measure

The Gini Index measures the impurity or inequality in a data set [2], based on a binary split. The Gini index of a set measures the probability of not duplicating classifications in a data set, for any class. The Gini measure of a split is the weighted sum of the Gini indices of each partition set, with weights given by the fraction of the total no. of samples that they occupy. Then, the reduction in impurity is given by the difference in the Gini measure before and after implementing the split criterion. By minimizing impurity, we partition the data set into more homogenized subsets, ideally corresponding to one class per subset.

# Algorithm Implementation

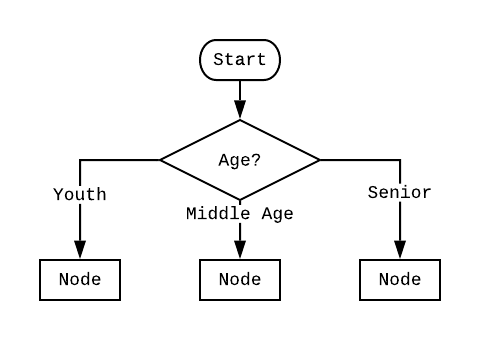
Using a small sample to test the algorithm, we consider the operation of the algorithm. Using Example 8.1 of [2], we create a test data set. We show the original data set in Table 1. We generate a decision tree using the Info Gain measure and allocate 20% of the data set for testing. We note that the algorithm performs with a nominal 100% accuracy, but due to the small sample size we are not concerned with the numerical performance of the algorithm in this case. In this instance, and of data sets of similarly low complexity, we can interpret the decision tree directly to make inferences about the system, and we consider this idea in more detail below.

### Table 1: All Electronics Customer Database[2]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RID | Age | Income | Student | Credit Rating | Buys computer? | |
| 1 | youth | high | no | fair | no |
| 2 | youth | high | no | excellent | no |
| 3 | middle aged | high | no | fair | yes |
| 4 | senior | medium | no | fair | yes |
| 5 | senior | low | yes | fair | yes |
| 6 | senior | low | yes | excellent | no |
| 7 | middle aged | low | yes | excellent | yes |
| 8 | youth | medium | no | fair | no |
| 9 | youth | low | yes | fair | yes |
| 10 | senior | medium | yes | fair | yes |
| 11 | youth | medium | yes | excellent | yes |
| 12 | middle aged | medium | no | excellent | yes |
| 13 | middle aged | high | yes | fair | yes |
| 14 | senior | medium | no | excellent | no |

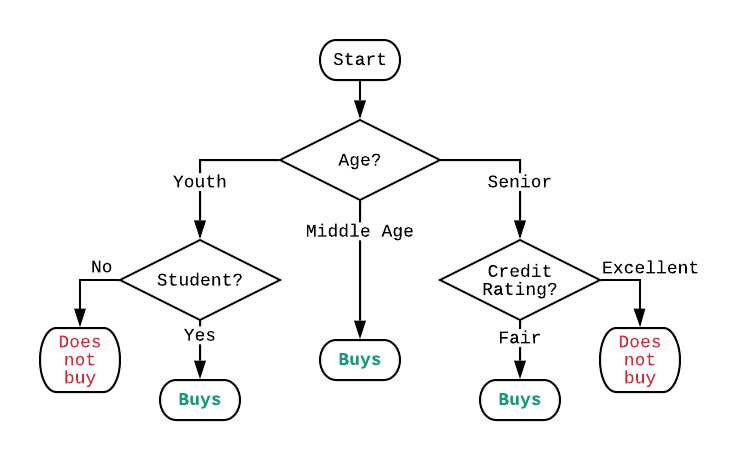
Given that there are four nontrivial attributes in this data set (excluding RID and the classification), we expect four layers in the decision tree. We parse the root object of the tree to see its children. Root is a **Node** object, with three major properties for determining a split: Attribute, Children and Criterion. We observe that the root object has an attribute ‘Age’. The root object has criterion [1,2,3], corresponding to the three attribute values youth, middle age and senior, respectively encoded as 1,2, and 3. Thus, one branch of the tree considers youth entries only, and the other branch considers middle age and senior on equal ground. Taking the first children corresponds the youth option. We show the first iteration of this procedure in Figure 2.

### Figure 2: First Iteration of Decision Tree Generation for *AllElectronics* Database



On the second level of the procedure, the algorithm considers a new attribute to split on. Based on which branch was taken at the first level, we observe different attributes being selected for each child in the second. Indeed, each branch produces selects a different partition, with different entropies. For each node in the second level, we repeat this procedure to generate the full decision tree. In Figure 3, we show the result of the recursion.

### Figure 3: Complete Decision Tree for *AllElectronics* Database



Due to the low complexity of the data set, we observe that the algorithm only requires two layers of decision making to achieve full accuracy. With more samples, the algorithm would be forced to make more difficult splits, and the actual entropies of each split would be nontrivial. Drawing this diagram becomes exponentially more complicated with additional layers, and so we yield at this point.

We notice that income level was determined to be not an important feature at any branch, and indeed this is plausible. Based on additional data provided by other attributes (such as credit rating), it is always possible that we incorporate redundant information into our ontology and in this case, the algorithm did not need to include it. With more complicated or large data sets, that redundancy may unveil more subtle trends in the data, so its absence in the sample’s decision tree maybe only represent a special simplification of the more general statistics of the population. This phenomenon of limited data restricting our ability to make inferences is an example of the problem of *censorship* in statistics.

# Experiments

Using the UCI Machine Learning Repository [2], we consider 10 test data sets to evaluate our implementation of each algorithm. Because each data set is attributed to different contributors, we will refer to each data set using its name within the UCI database.

We begin by considering the success of the Decision Tree Induction algorithm using each of the three information-metrics in Tables 2 to 4. For each data set, we detail the no. instances (samples) and attributes of data vectors. We also include a variable Test %, which gives what percentage of the dataset was used for testing and evaluation, but since we use the same percentage for each measure, we only show this variable in the first table. For the DTI tests, we consider the execution time, memory usage, accuracy, specificity and sensitivity performance metrics.

### Table 2: Gain Ratio DTI Test Statistics

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Data Set Name** | **No. Samples** | **No. Attr.** | **Test %** | **Execution Time (s)** | **Memory Usage (kB)** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Breast Cancer Coimbra | 116 | 10 | 0.2 | 1.040 | 150,499 | 0.667 | 0.545 | 0.769 |
| 2 | Credit Approval | 690 | 15 | 0.2 | 5.145 | 147,816 | 0.819 | 0.767 | 0.859 |
| 3 | Autistic Spectrum Disorder Screening | 292 | 21 | 0.2 | 2.620 | 146,981 | 0.763 | 0.677 | 0.857 |
| 4 | Chronic Kidney Disease | 400 | 25 | 0.2 | 1.975 | 146,985 | 0.926 | 0.947 | 0.875 |
| 5 | Connectionist Bench | 208 | 60 | 0.2 | 9.440 | 147,460 | 0.738 | 0.762 | 0.714 |
| 6 | Glass | 214 | 11 | 0.2 | 0.772 | 147,161 | 0.814 | 0.778 | 0.824 |
| 7 | Horse Colic | 368 | 27 | 0.2 | 1.469 | 147,010 | 0.817 | 0.933 | 0.467 |
| 8 | Adult | 48842 | 14 | 0.2 | 24.410 | 152,318 | 0.789 | 0.941 | 0.319 |
| 9 | Mushroom | 8124 | 22 | 0.2 | 11.295 | 148,034 | 0.814 | 0.506 | 0.924 |
| 10 | Soybean-Large | 307 | 35 | 0.2 | 0.870 | 149,660 | 0.952 | 1.000 | 0.944 |

### Table 3: Info Gain DTI Test Statistics

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Data Set Name** | **No. Samples** | **No. Attr.** | **Execution Time (s)** | **Memory Usage (kB)** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Breast Cancer Coimbra | 116 | 10 | 0.850 | 148,480 | 0.625 | 0.583 | 0.769 |
| 2 | Credit Approval | 690 | 15 | 3.530 | 147,145 | 0.775 | 0.700 | 0.833 |
| 3 | Autistic Spectrum Disorder Screening | 292 | 21 | 1.060 | 147,268 | 1.000 | 1.000 | 1.000 |
| 4 | Chronic Kidney Disease | 400 | 25 | 1.440 | 147,616 | 0.926 | 0.947 | 0.875 |
| 5 | Connectionist Bench | 208 | 60 | 4.715 | 147,087 | 0.738 | 0.667 | 0.810 |
| 6 | Glass | 214 | 11 | 0.620 | 147,849 | 0.791 | 0.778 | 0.794 |
| 7 | Horse Colic | 368 | 27 | 0.985 | 147,329 | 0.817 | 0.956 | 0.400 |
| 8 | Adult | 48842 | 14 | 21.495 | 154,251 | 0.803 | 0.909 | 0.474 |
| 9 | Mushroom | 8124 | 22 | 11.020 | 150,667 | 0.828 | 0.581 | 0.917 |
| 10 | Soybean-Large | 307 | 35 | 0.840 | 146,989 | 0.952 | 1.000 | 0.944 |

### Table 4: Gini Index DTI Test Statistics

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Data Set Name** | **No. Samples** | **No. Attr.** | **Execution Time (s)** | **Memory Usage (kB)** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Breast Cancer Coimbra | 116 | 10 | 0.875 | 147,812 | 0.792 | 0.400 | 0.895 |
| 2 | Credit Approval | 690 | 15 | 7.800 | 148,169 | 0.797 | 0.712 | 0.861 |
| 3 | Autistic Spectrum Disorder Screening | 292 | 21 | 3.655 | 147,853 | 0.729 | 0.613 | 0.857 |
| 4 | Chronic Kidney Disease | 400 | 25 | 6.305 | 147,325 | 0.864 | 0.895 | 0.792 |
| 5 | Connectionist Bench | 208 | 60 | 18.269 | 148,197 | 0.571 | 0.619 | 0.524 |
| 6 | Glass | 214 | 11 | 1.275 | 147,108 | 0.767 | 0.667 | 0.794 |
| 7 | Horse Colic | 368 | 27 | 2.595 | 147,370 | 0.800 | 0.911 | 0.467 |
| 8 | Adult | 48842 | 14 | 23.561 | 156,631 | 0.776 | 0.952 | 0.231 |
| 9 | Mushroom | 8124 | 22 | 23.150 | 147,919 | 0.815 | 0.529 | 0.917 |
| 10 | Soybean-Large | 307 | 35 | 3.919 | 150,774 | 0.984 | 1.000 | 0.981 |

We include the statistical results of the SVM and ANN implementations separately, with emphasis on classification performance. We record the accuracy, sensitivity and specificity of the SVM and ANN implementations in Tables 5 and 6.

### Table 5: SVM Test Statistics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Data Set Name** | **No. Samples** | **No. Attr.** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Breast Cancer Coimbra | 116 | 10 | 0.657 | 0.529 | 0.778 |
| 2 | Credit Approval | 690 | 15 | 0.816 | 0.778 | 0.846 |
| 3 | Autistic Spectrum Disorder Screening | 292 | 21 | 0.852 | 0.786 | 0.913 |
| 4 | Chronic Kidney Disease | 400 | 25 | 0.934 | 0.949 | 0.905 |
| 5 | Connectionist Bench | 208 | 60 | 0.619 | 0.621 | 0.618 |
| 6 | Glass | 214 | 11 | 0.800 | 0.714 | 0.841 |
| 7 | Horse Colic | 368 | 27 | 0.789 | 0.921 | 0.481 |
| 8 | Adult | 48842 | 14 | 0.805 | 0.870 | 0.601 |
| 9 | Mushroom | 8124 | 22 | 0.785 | 0.521 | 0.880 |
| 10 | Soybean-Large | 307 | 35 | 0.978 | 1.000 | 0.975 |

### Table 6: Artificial Neural Network Test Statistics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Data Set Name** | **No. Samples** | **No. Attr.** | **Accuracy** | **Sensitivity** | **Specificity** |
| 1 | Breast Cancer Coimbra | 116 | 10 | 0.743 | 0.588 | 0.889 |
| 2 | Credit Approval | 690 | 15 | 0.855 | 0.844 | 0.863 |
| 3 | Autistic Spectrum Disorder Screening | 292 | 21 | 0.795 | 0.786 | 0.804 |
| 4 | Chronic Kidney Disease | 400 | 25 | 0.909 | 0.924 | 0.881 |
| 5 | Connectionist Bench | 208 | 60 | 0.698 | 0.517 | 0.853 |
| 6 | Glass | 214 | 11 | 0.692 | 0.524 | 0.773 |
| 7 | Horse Colic | 368 | 27 | 0.833 | 1.000 | 0.444 |
| 8 | Adult | 48842 | 14 | 0.805 | 0.932 | 0.403 |
| 9 | Mushroom | 8124 | 22 | 0.864 | 0.747 | 0.905 |
| 10 | Soybean-Large | 307 | 35 | 0.989 | 0.909 | 1.000 |

# Discussion

Given our results recorded in the previous section, we analyze the performance of our algorithms, beginning with execution time. We record the execution time for our three implementations of Decision Tree Induction in Plot 1. We note that consistently the Gini Index metric is the worst in time complexity, followed by Gain Ratio and Info Gain. In comparison to Info Gain, Gain Ratio performs more arithmetic operations, leading to slightly worse scaling.

### Plot 1: Execution Time of DTI Algorithms

Next, we consider the accuracy measurement of all algorithms. In Plot 2, we show the measured accuracy for all three DTI implementations, as well as the SVM and ANN versions. Because we used binary classification at the output, the prediction task was occasionally more difficult than usual. In large data sets, the ANN performed marginally better than other metrics. We note that in certain data sets that included may numerical values, such as Connectionist Bench, the performance was low in general, likely as a result of preprocessing which over simplified the data set. Interestingly, this effect doesn’t appear to be compensated by the large no. of attributes.

### Plot 2: Algorithm Accuracy Measurements

We consider the sensitivity of each implementation in Plot 3. We define sensitivity to be . The significance of this quantity is that it tells us how often a classification of “1” is correct. As such, high accuracy data sets often have similarly high sensitivities. Some examples, such as Mushroom, show lower sensitivity than other datasets of similar accuracy. We attribute this to the success of this data set in the specificity measurement, of Plot 4. Overall, most algorithms performance similarly on each data, with no obvious best option.

### Plot 3: Algorithm Sensitivity Measurements

We consider the specificity of each implementation in Plot 4. We define specificity to be . The significance of this quantity is that it tells us how often a classification of “0” is correct. Overall, most algorithms performed similarly. Surprisingly, the adult data set showed low specificity despite its large size, and hence large training set. We anticipate that the predicted classifications were biased towards “1”.

### Plot 4: Algorithm Specificity Measurements

We consider the execution time versus the no. of attributes in a data set in Plot 5. We notice that we see that the DTI algorithms do not scale much in time per sample until larger no. of attributes are considered. It is possible that this follows an exponential model, but the regression produces insignificant predictability in this case.

### Plot 5: DTI Algorithms Normalized Time vs No. Attributes

Finally, in Plot 6 we visualize the memory usage of each DTI implementation. We notice a very large constant offset in these measurements, since each implementation imports the same libraries for every test. There is a weak dependence on data set size, as seen in the Adult data set, but these fluctuations are miniscule in comparison to the overall offset of the measurements.

### Plot 6: DTI Algorithms Memory Usage

# Conclusion

We implemented the Decision Tree Induction algorithm using three different attribution selection measures and compared its efficiency to a typical SVM or ANN implementation. In general, execution time increased with the no. of attributes in use and data set size. The memory usage only varied slightly across tests. We observed that accuracy, sensitivity and specificity of the algorithms vary based on the data being assessed. Overall, each implementation performed similarly. Further improvements upon our implementations would further reduce time complexity, include in-code preprocessing and would provide more direct visualizations of the decision tree (i.e. generated by the script).

# References

1. D. Dua, C. Graff. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
2. J. Han, M. Kamber, J. Pei. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Waltham, MA: Morgan Kaufmann Publishers.
3. Brid, Rajesh S. (2018). *Decision Trees: A simple way to visualize a decision tree*, Grey Atom, Medium Corporation.