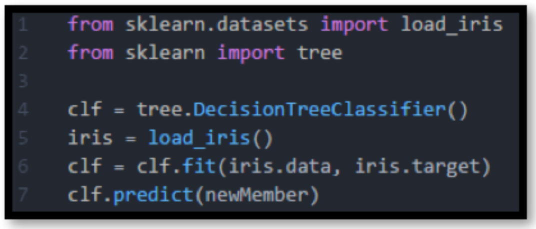
# Visualization of Decision Trees

Ather Qureshi
aqures43@uwo.ca
The University of Western Ontario, CANADA

**Abstract.** Classification is a powerful and useful tool that is aided by machine learning. However, the computations and mathematics underlying this tool are very complex and difficult to conceptualize. This paper describes a tool that visualizes the decision tree classifier. This tool can aid the programmer in understanding the process as well as the results of the algorithm. It is compatible with any dataset that has numerical features and a discrete number of classes. Its zeitgeist is to build a mental model of the decision tree for the user as well as be a light weight and easy to use classifier program.

## Introduction and Background

Sometimes, it is all too easy to utilize machine learning in programs. Programming libraries such as scikit-learn, Weka, Tensorflow, R-caret offer many easy to use machine and deep learning packages in which the programmer can utilize powerful algorithms without mathematical and framework intuition (Wale Akinfaderin, 2016). For instance, as seen in figure 1, one can create and utilize a decision tree classifier using python's scikit-learn library in only six lines of code and do so without any knowledge at all how this classifier works. It is like the programmer is working with a black box that takes input and output. The user can have no knowledge of what happens within the box.



```
from sklearn.datasets import load_iris
from sklearn import tree

clf = tree.DecisionTreeClassifier()
iris = load_iris()
clf = clf.fit(iris.data, iris.target)
clf.predict(newMember)
```

**Figure 1:** Utilizing the Python Library learn to create a Decision Tree Classifier (Pedregosa, 2011).

This allows the programmer to easily incorporate the power of machine learning in their programs with ease. However, many issues arise if the programmer has rudimentary knowledge of the machine learning tools they use. First, an unfortunate side effect is that programmer may not be able to diagnose and resolve bugs that inevitably arrive when they use the machine learning tools. Second, the programmer may not be able to express confidence in the results of the tools since it is difficult to trust the results of the algorithm if one does not understand the algorithm itself. This issue is amplified when the results of a machine learning algorithm are used in prime applications such as genomics or systems biology (Chunmei Liu, 2013). Third, it is not the case that a programmer can select any arbitrary machine learning algorithm for their dataset. The programmer must "select the right algorithm which includes considering accuracy, training time, model complexity, number of parameters and number of features" (Wale Akinfaderin, 2016). Only when all these requirements are satisfied, can a programmer effectively utilize machine learning tools.

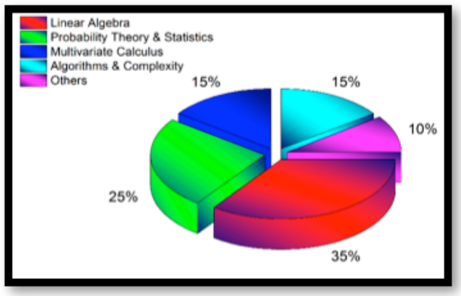


**Figure 2:** Mix of mathematical skills one IBM researcher deems necessary for utilizing machine algorithms. (Wale Akinfaderin, 2016)

But, learning the complexities of machine learning algorithms requires an elevated level of mathematical knowledge not typical of all software engineers. The task of understanding machine learning algorithms is a complex cognitive activity. This is because the task involves the use of complex psychological processes and the presence of complex conditions (Ericcson & Hastie, 1994). Also, both perception and memory are involved in making sense of the mathematical concepts underlying machine learning and the task involves many variables that exhibit a high level of interdependence (Sedig & Parsons, 2013). Furthermore, IBM data science researchers hypothesize that to obtain meaningful results from machine learning, one requires a mix of mathematical knowledge such as linear algebra, probability and statistics, multivariate calculus, algorithms and complexity as well as many other concepts (Wale Akinfaderin, 2016). Figure 2 shows a visual representation of this mix. This all points toward the task of learning machine learning algorithms as *hard and reflexive*. Programmers will have difficulty accessing the information space (the algorithms, and the result of the data) and combining this information with other sources of information to reach conclusions.

I argue that it is not necessary for one to delve into such low level mathematical proofs to correctly employ machine learning tools. Visual representations of the machine learning algorithms can act as a perceptible form of the encoded information space. And thus, the representations can act as a mental interface that can connect the human mind to the information space (Sedig & Parsons, 2013). Users can understand the inner-workings of the algorithms by engaging with a visual representation. Through this interface, the user can grasp the concepts underlying the machine learning algorithms which will in turn, inspire trust in the results of the algorithm as well as serve as a valuable tool in debugging. This paper discusses the use of a visual representation to showcase the algorithm of prediction in decision tree classifiers. This visual representation tool is called decision visualizer.

## Decision Trees

This paper describes a tool used to visualize a popular machine learning algorithm called decision tree classifiers. Before one can discuss classifier and decision trees, one must many key concepts that are part of the discussion of these concepts. To start, what is a decision tree? A Decision tree is a widely used form of representing algorithms and knowledge. They naturally represent identification and testing algorithms that specify the next test to perform based on the results of the previous tests (Chikalov & Lakhmi , 2011). Specifically, each internal node is a test on a feature, and each branch represents the outcome of a test. Starting at the root node, traverse downward (or away from the root node) until an external node is reached. The label on the resulting external node is the recommended action. Figure 3 showcases a knowledge decision tree that can be used for selecting the right machine learning algorithm. The tree takes the user from a general case to a specific case based on the decisions at each node.

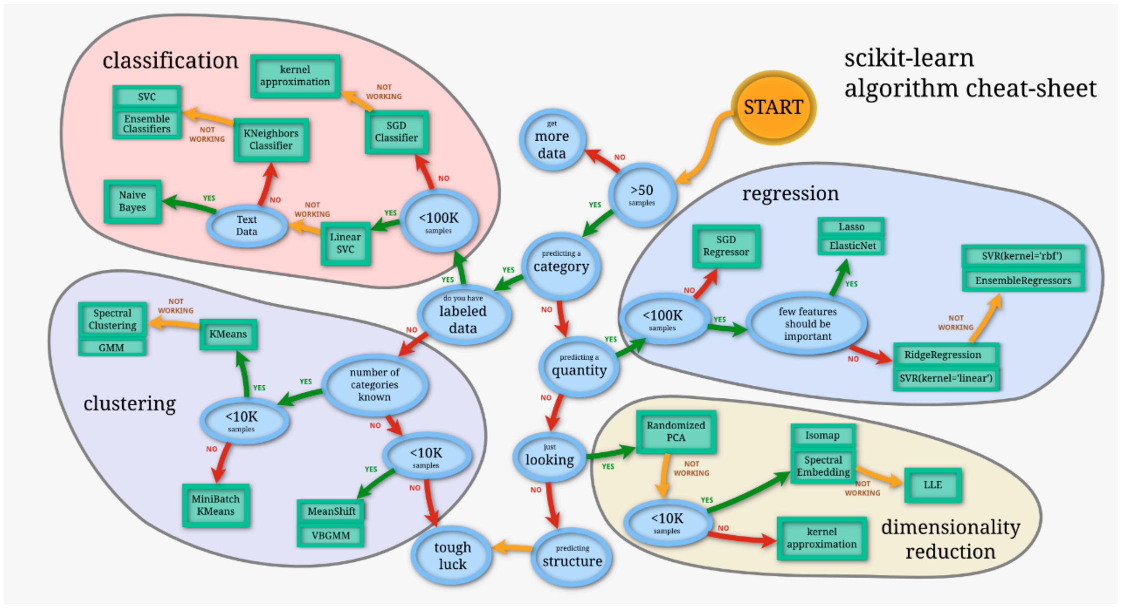


**Figure 3:** Decision Tree showcasing which machine learning algorithms one must use given their dataset (Pedregosa, 2011).

## Datasets

| Members/Samples | Class | Feature (Color) | Feature (Texture) |
|---|---|---|---|
| 1 | Apple | Red | Rough |
| 2 | Banana | Yellow | Mushy |
| 3 | ? | Maroon | Rough |

**Figure 4:** Example of a small dataset and a common classification problem

A **dataset** is a collection of members and can used as a training set for a machine learning classifier. A **member** is a new observation and is usually a single row in a dataset. They are also called samples. A **class** is a category. It is a grouping for a member. For instance, apples, bananas are all part of the class fruit and not of the class vegetable. A **feature** is a characteristic of a member. It is also called an attribute. A **value** is essentially the values of the member's features. (Mitchell, 1997). Figure 4 shows a small dataset. The classes are Apple and Banana. The features are color and texture. There are 2 unique members in the dataset. Member 3 is a new member in which the target class is not known. This member can be classified via a classifier algorithm.

**Classifiers**

The concept of arriving from a generic category to a more specific category is precisely the purpose of a classifier. A classifier identifies to which category an object belongs to (Kubat, 2015). In machine learning, classification is the problem of identifying to which of a set of classes a new member belongs, on the basis of a training set of data containing members whose class membership is known. There are several types of classification algorithms (see figure 3). But in this paper, we will focus on the decision tree classifier.

After a machine learning classifier learns from a test dataset, it can then be used to predict the class of new members. In figure 4, the classifier could be given the task of predicting the target class of member number 3. The accuracy of the prediction for a classifier depends on the quality and quantity of the test data, as well as the classifier algorithm used (Mitchell, 1997).

## Decision Tree Classifier

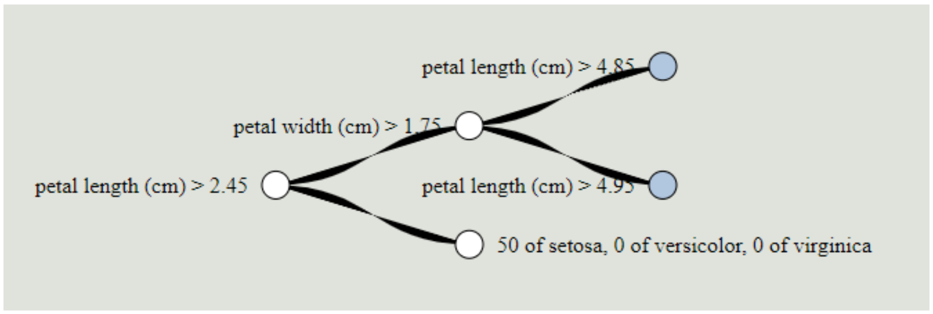


**Figure 5**: Screen shot of Decision Visualization on the Iris Data Set Decision Tree

In short, a decision tree classifier learns from a test data set, and generate a decision tree that will predict the target class of new members (Pedregosa, 2011). There are many different complex algorithms that can generate this tree but that is not the scope of this paper. In the tool described in this paper, the python sci-kit decisionTreeClassifier module is used. This specific version of the decision tree classifier uses an optimized version of the CART algorithm (see section on time complexity for efficiency).

Figure 5 showcases a typical decision tree. Once a decision tree is generated, all a program or user must do is compare the features of its new member to the key features in the tree, and simply traverse until a target class is identified. Each internal node tests a feature, each branch corresponds to the feature value, and each leaf node assigns a classification. Decision tree classifiers only work on datasets that have members described by feature-value pairs, and have discrete classes for each member (Mitchell, 1997). See section on Decision Visualizer - Iris Dataset for an example of how to traverse the tree.

**Time Complexity**:

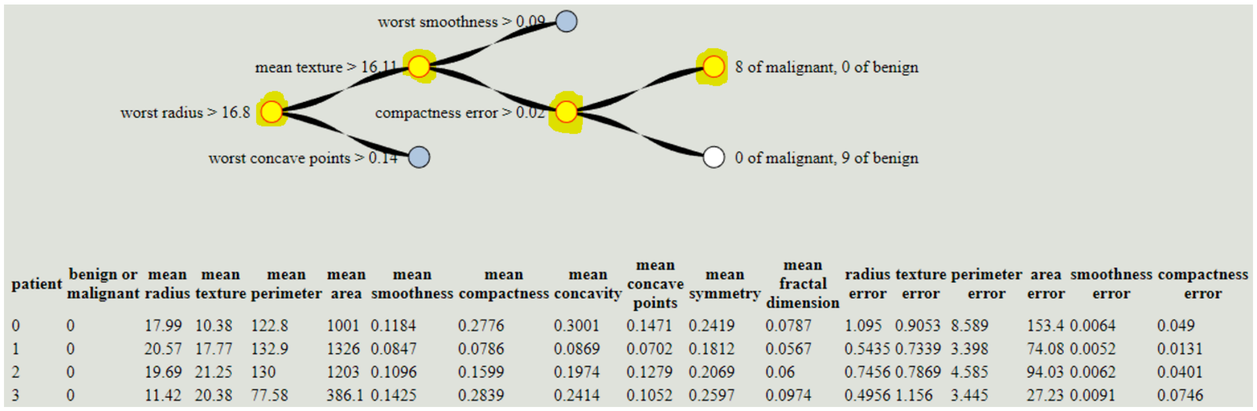
**Generation** of the tree depends on the algorithm used. In decision visualizer, the python libraries scikit's decisionTreeClassifier module is used. It generates the tree using an optimized version of the CART algorithm. The algorithm has a time complexity of $O(n_{features} n_{members} \log(n_{members}))$ on an average case (Pedregosa, 2011). This computation will only have to be done once and the output is stored in a negligible amount of memory. Generation of the tree must be done before prediction(s).

**Prediction** by a decision tree classifier will in the worst case traverse the height of the tree. The *height* of the tree is equal to log($n$) where $n$ is the number of nodes in the tree. At each node, there is a test on an attribute that will determine which branch to take. This is a constant time or O(*1*) operation. Thus, each prediction has a time complexity is equal to O(log($n$)).

This remainder of this paper will introduce a tool that will showcase a decision tree visualizing program.

## Decision Visualizer

Decision Visualizer is a tool that can both generate a decision tree and visualize these trees in an interactive way. The tool will first create a decision tree using python's sci-kit learn tree library. Then, it will visualize the decision tree onto a webpage using the d3 library. In the visualization, users can unwrap further branches of the tree by clicking on internal nodes. They can also enter in new member information and see how the decision tree predicts the class of the object. Similarly, they can also click on members of the test dataset and the decision tree will reappear showing the path of the classification on the tree. The nodes with an outline/stroke of red are the path in the tree that the member took in classifying.



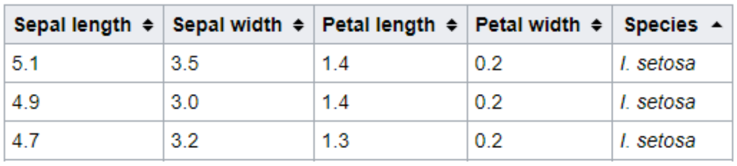
| patient | benign or malignant | mean radius | mean texture | mean perimeter | mean area | mean smoothness | mean compactness | mean concavity | mean concave points | mean symmetry | mean fractal dimension | radius error | texture error | perimeter error | area error | smoothness error | compactness error |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 17.99 | 10.38 | 122.8 | 1001 | 0.1184 | 0.2776 | 0.3001 | 0.1471 | 0.2419 | 0.0787 | 1.095 | 0.9053 | 8.589 | 153.4 | 0.0064 | 0.049 |
| 1 | 0 | 20.57 | 17.77 | 132.9 | 1326 | 0.0847 | 0.0786 | 0.0869 | 0.0702 | 0.1812 | 0.0567 | 0.5435 | 0.7339 | 3.398 | 74.08 | 0.0052 | 0.0131 |
| 2 | 0 | 19.69 | 21.25 | 130 | 1203 | 0.1096 | 0.1599 | 0.1974 | 0.1279 | 0.2069 | 0.06 | 0.7456 | 0.7869 | 4.585 | 94.03 | 0.0062 | 0.0401 |
| 3 | 0 | 11.42 | 20.38 | 77.58 | 386.1 | 0.1425 | 0.2839 | 0.2414 | 0.1052 | 0.2597 | 0.0974 | 0.4956 | 1.156 | 3.445 | 27.23 | 0.0091 | 0.0746 |

**Figure 6**: Snapshot of decision visualizer on the Breast Cancer Wisconsin (Diagnostic) Data Set from the UCI Machine Learning Repository. (Lichman, 2013). Nodes that are highlighted and stroked with red are the classification path taken.

The purpose of this tool is to make sense of the calculations required to classify members. The user will be able to see the tested attributes that will partition their data and how it affects the classification of items in their dataset with the power of visualization. They can understand what features are key in classifying their data, and thus will have more trust in the results of the algorithm. This tool effectively removes some of the black box the decision tree classifier algorithm without bombarding the user with mathematical complexity.

The tool has 2 different iterations. The first iteration is custom made for the Fisher iris dataset gathered from the UCI Machine Learning Repository (Lichman, 2013). The other iteration is a generic visualizer that can work on any numerical dataset, that has feature-value pairs and discrete classes.

### Decision Visualizer - Iris Dataset



| Sepal length ⬍ | Sepal width ⬍ | Petal length ⬍ | Petal width ⬍ | Species ▲ |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | *I. setosa* |
| 4.9 | 3.0 | 1.4 | 0.2 | *I. setosa* |
| 4.7 | 3.2 | 1.3 | 0.2 | *I. setosa* |

**Figure 7:** 3 members from the Iris Dataset (Lichman, 2013).

Fisher's Iris multivariate dataset was used for this implementation of decision visualizer (Lichman, 2013). It is a relatively simple dataset since it only has four features. This dataset represents plants. It has 4 unique features which are sepal length, sepal width, petal length, and petal width. These are all measured in centimeters. The members are classified into 3 different classes: Setosa, Virginica, and Versicolor. These classes represent the species of each member. A visual representation of the dataset can be seen below in Figure 8. The color of each member also indicates the class of each member. The sepal length and width of each member is signified via its location in the scatterplot. The Y axis is sepal length and the X axis is sepal width. The petal length and width are represented by the length and width of each point respectively.

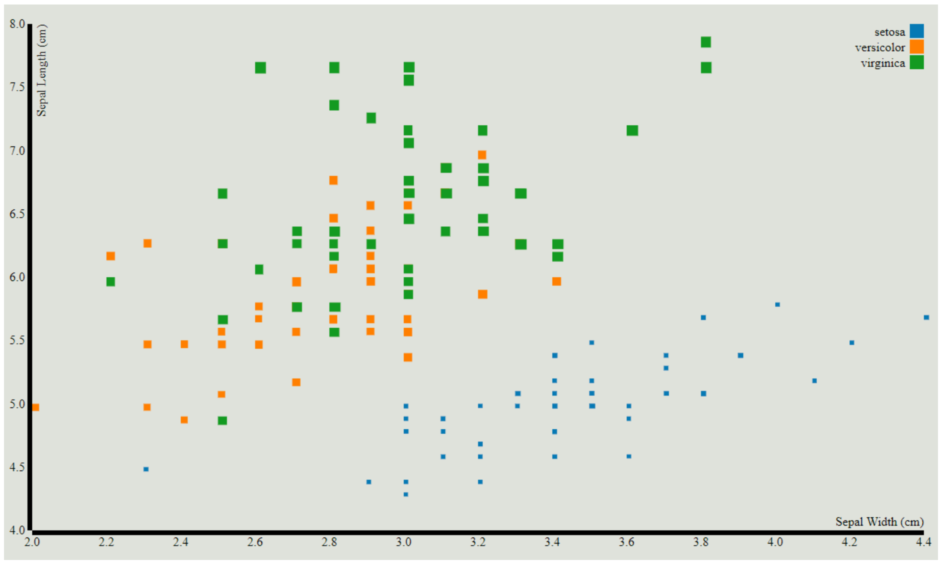
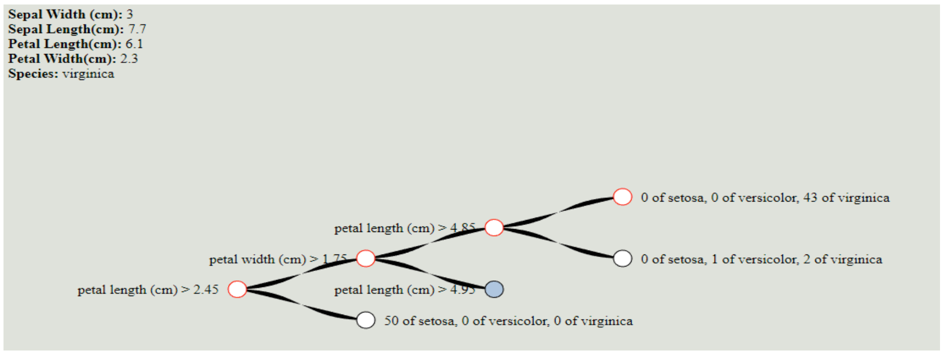


**Figure 8:** Scatterplot of Iris Data in the Iris Visualizer. In addition to the axis showing sepal width and sepal length, petal length and width are represented by the length and width of each point respectively.

The iris visualizer is interactive and dynamic. Users can click on points directly on the scatterplot and the decision tree will regenerate showing the classification path of the clicked point. The members in the scatterplot also react to events such as mouse hovering and clicks. Also, member information is shown to the user directly on the page on the top left of the tree. Users can also select internal nodes to expand their subtrees.



**Sepal Width (cm):** 3
**Sepal Length(cm):** 7.7
**Petal Length(cm):** 6.1
**Petal Width(cm):** 2.3
**Species:** virginica

petal length (cm) > 2.45
petal width (cm) > 1.75
petal length (cm) > 4.85
petal length (cm) > 4.95
0 of setosa, 0 of versicolor, 43 of virginica
0 of setosa, 1 of versicolor, 2 of virginica
50 of setosa, 0 of versicolor, 0 of virginica

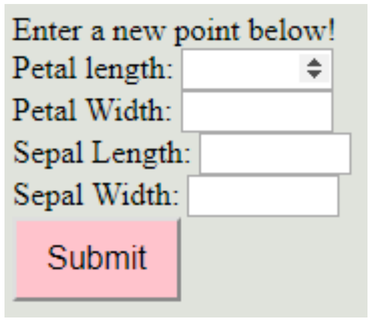
**Figure 9**: Snapshot of Iris Decision Tree Visualizer

Figure 9 showcases a snapchat of the program. To use the program, the user must select a member by either entering in new member or clicking on a test member in the scatterplot portion of the program. Upon selecting a member, the tree will regenerate. Member information will be available to see on the top left, as well as the class/species. If the user has entered in new member information, the class/species field will show a prediction of the new member's class. In figure 9's example, the member selected is of class virginica and was part of the test dataset. A walkthrough of a classification path of the decision tree will be shown below.

1. **Petal Length (cm) > 2.45**
    a. Our member has a petal length value of 6.1, this test returns true
    b. Traverse to the higher branch right of the current node.
2. **Petal Width (cm) > 1.75**
    a. Our member has a petal width value of 2.3, this test returns true
    b. Traverse to the higher branch right of the current node.

3. **Petal Length (cm) > 4.85**
    a. Our member has a petal length value of 6.1, this test returns true
    b. Traverse to the higher branch right of the current node
4. **External Node Reached: 0 of setosa, 0 of versicolor, 43 of virginica**
    a. Class prediction is Virginica since the other two classes have zero as their value
        i. meaning that in the test dataset, in this specific partition, only members with class Virginica were in it, thus the classifier predicts new member is of class virginica.

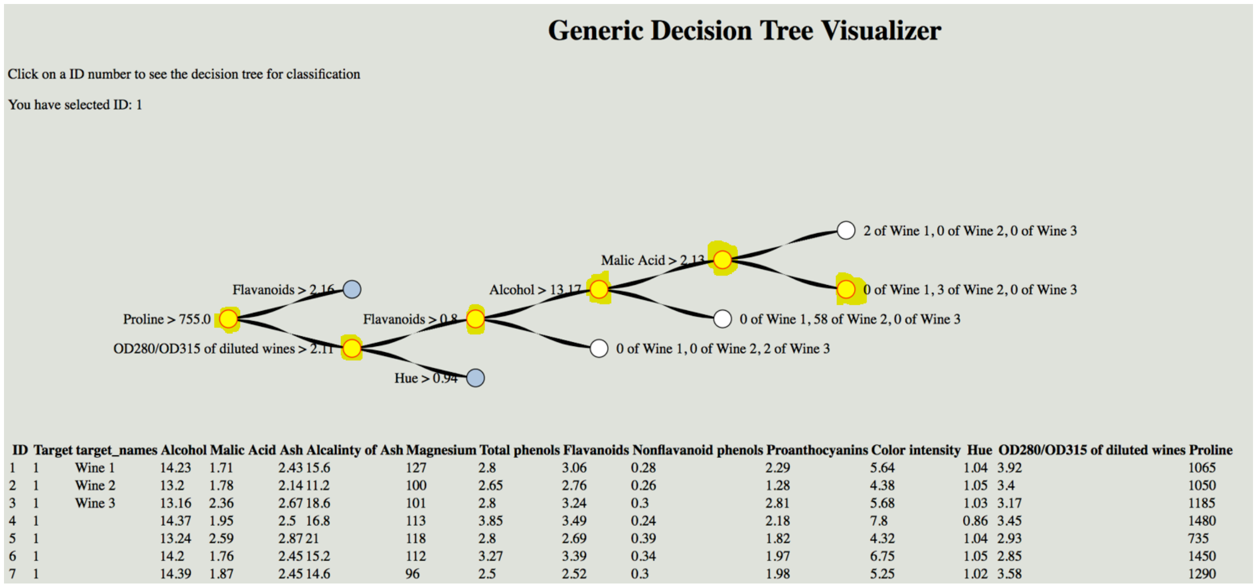
Users can also add in new member information into the program via a small GUI as seen in Figure 10. After the user enters in acceptable data into the fields, they can press the submit button and the decision tree classifier will predict the class of the new member (in this case, the species of the member). It will also update the tree to show the classification path the new member took. The user must enter in member information for **all** features to be predicted, and the **value in each feature must be in between the range** of the minimum and maximum value for that feature. Specifically, the minimum and maximum value for that feature found in the test dataset. This is a limitation on the visualizing tool.



| Feature | Range |
|---|---|
| **Petal Length** | 1.00 ≤ 6.90 |
| **Petal Width** | 0.10 ≤ 2.50 |
| **Sepal length** | 4.30 ≤ 7.90 |
| **Sepal Width** | 2.00 ≤ 4.40 |

**Figure 10:** Graphical User Interface that allows user to enter in new member information and table showing range for the data.

## Generic Dataset Viewer



Generic Decision Tree Visualizer

Click on a ID number to see the decision tree for classification

You have selected ID: 1

Proline > 755.0
Flavanoids > 2.16
OD280/OD315 of diluted wines > 2.11
Flavanoids > 0.8
Hue > 0.94
Alcohol > 13.17
0 of Wine 1, 0 of Wine 2, 2 of Wine 3
Malic Acid > 2.13
0 of Wine 1, 58 of Wine 2, 0 of Wine 3
2 of Wine 1, 0 of Wine 2, 0 of Wine 3
0 of Wine 1, 3 of Wine 2, 0 of Wine 3

| ID | Target | target_names | Alcohol | Malic Acid | Ash | Alcalinty of Ash | Magnesium | Total phenols | Flavanoids | Nonflavanoid phenols | Proanthocyanins | Color intensity | Hue | OD280/OD315 of diluted wines | Proline |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | Wine 1 | 14.23 | 1.71 | 2.43 | 15.6 | 127 | 2.8 | 3.06 | 0.28 | 2.29 | 5.64 | 1.04 | 3.92 | 1065 |
| 2 | 1 | Wine 2 | 13.2 | 1.78 | 2.14 | 11.2 | 100 | 2.65 | 2.76 | 0.26 | 1.28 | 4.38 | 1.05 | 3.4 | 1050 |
| 3 | 1 | Wine 3 | 13.16 | 2.36 | 2.67 | 18.6 | 101 | 2.8 | 3.24 | 0.3 | 2.81 | 5.68 | 1.03 | 3.17 | 1185 |
| 4 | 1 |  | 14.37 | 1.95 | 2.5 | 16.8 | 113 | 3.85 | 3.49 | 0.24 | 2.18 | 7.8 | 0.86 | 3.45 | 1480 |
| 5 | 1 |  | 13.24 | 2.59 | 2.87 | 21 | 118 | 2.8 | 2.69 | 0.39 | 1.82 | 4.32 | 1.04 | 2.93 | 735 |
| 6 | 1 |  | 14.2 | 1.76 | 2.45 | 15.2 | 112 | 3.27 | 3.39 | 0.34 | 1.97 | 6.75 | 1.05 | 2.85 | 1450 |
| 7 | 1 |  | 14.39 | 1.87 | 2.45 | 14.6 | 96 | 2.5 | 2.52 | 0.3 | 1.98 | 5.25 | 1.02 | 3.58 | 1290 |

**Figure 11:** Screenshot of Decision Visualizer on Wine Data set from the UCI Machine Learning Repository. (Lichman, 2013). Classification path is highlighted for member number 1.

The generic dataset viewer allows the user to provide their own test dataset (in the form of a csv file), and then can visualize the decision tree on a webpage. A snapshot of the program can be seen on Figure 11. **Explicit instructions can be found in Appendix B**. But at an elevated level, all a user needs to use this program is to provide a csv file which contains a dataset that has members described by feature-value pairs, and has discrete classes for each member. The user can then supply this dataset to a tree generator program which will create a representation of the tree. Then, the user sees a visualization of the tree on a web browser.

Like the Iris Visualizer, the user can interact with the tree and select rows/members of their dataset and see how the tree classifies them. But instead of clicking members on the scatterplot, the user will have to click the ID number of the member in table form. Also, the user can enter in new data and see how the decision tree classifies that data via the changeTreeObject(item) function in the developer console available in their browser. This function will simply take as a parameter a JavaScript object. This object needs to have all the same keys (the features) for the function to be able to change the tree. For convenience, a template of this object that is editable in the console can be found by just calling changeTree(0). The returned object of this function will be a sample object.

**Note: the underlying tree structure is the same and is stored as a JSON file. The changeTree function simply "traces" (executes the decision tree on member information) and "paints" the underlying tree and reloads to it to the webpage.  The tree is NOT regenerated every time the changeTree() function is called.**

## Summary and Future Work

Decision visualizer exists as a visual representation of the decision tree classifier. It acts as an interface between the user and the information space. Users can interact with the tool and effectively see how predictions are executed in the classifier. As users interact with this software, they will gain an understanding of the computations underlying the algorithm. Through this interaction, they will be able to gain a degree of trust with the results of the classifier and understand to a level, how the classifier predicts new members.

Decision visualizer is an attempt to remove the black box surrounding the many of the easy-to-use machine learning libraries. It does this without overwhelming the user in complexity. Future work could be done to show stages of the algorithm generation and the specific steps of the CART algorithm. Also, the tool could be upgraded to allow the user to use different types of classifiers on the same dataset. This would allow the user to explore cases in which some classifiers are a better fit for certain dataset than others (see figure 3 for other classifier algorithms). This will effectively shift the information space to decision tree classifiers to all classifiers, and will assist users in selecting correct algorithms in future tasks.

To further inspire trust in results, the decision tree classifier could allow users to upload a dataset of new members (of which the classes are known) after it has been trained and have the tool predict the class of them. Then, the tool could return a score of correct predictions. If the tool can show that it can classify new members accurately, users will trust the results. Furthermore, the tool could be tweaked to be more user friendly and accessible such that non-programmers will able to use the tool. For instance, the program could use the design principle of layering to manage the complexity and reinforce the relationships in the information.

Currently, there is many open-source and free software that allow users to visualize machine learning software such as Facets, and MLDemos. This tool can exist as a lightweight introduction to machine learning classifiers alongside these others.

## Appendix – A: Iris Visualizer

To access the iris visualizer, it can be found at this URL:

atherqureshi.github.io/Thesis_help/iris_visual_scatter.html

## Appendix – B: Setting up Generic Visualizer

Python 3.62 must be installed as well as these libraries:

- Numpy
- Pandas
- bs4
- sci-kit learn

Generic Dataset viewer can be found here:

github.com/atherqureshi/atherqureshi.github.io/tree/master/Thesis_help/offline%20thesis%20work

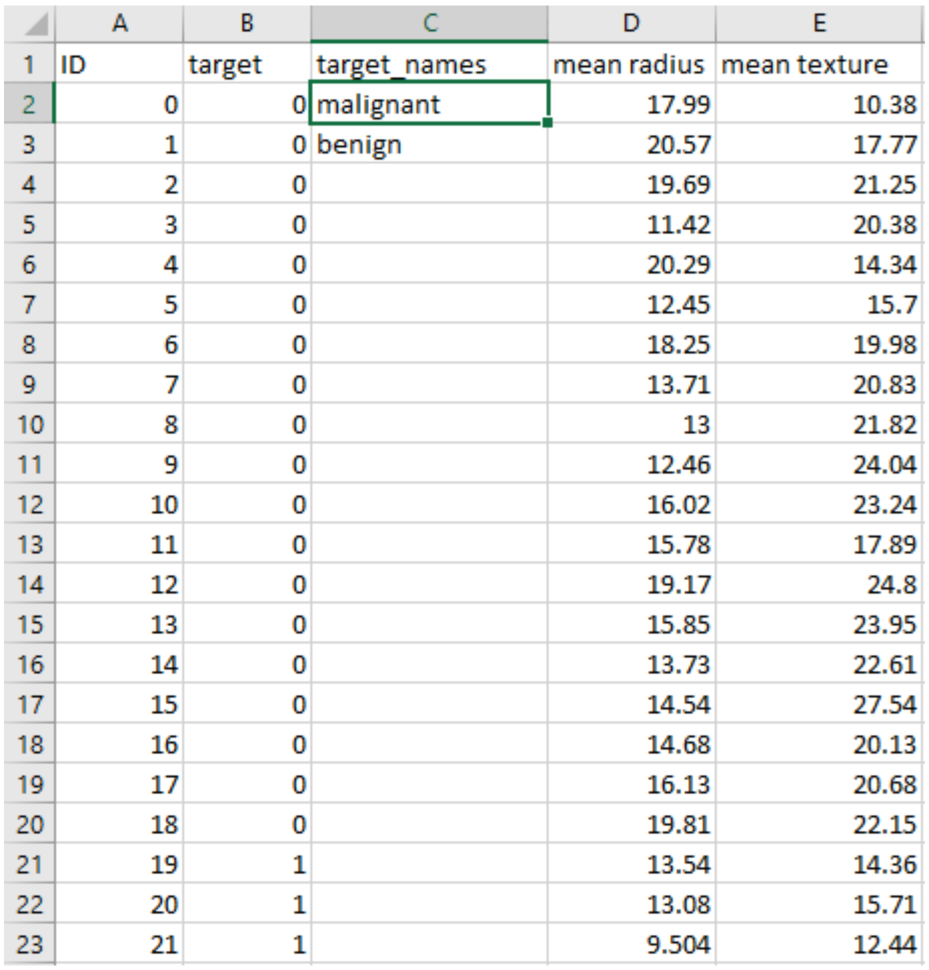
1. Simply download Generator.py, generic_tree.html, and the README.txt to a folder on your Computer.
2. Add in a properly formatted CSV file into the folder (format information can be found under Generic Dataset Viewer in this paper).
3. In a terminal, write via command line in the folder, type "python Generator.py x" where x is your csv file. This should generate two JSON files (tree and data), and a html page called index.
4. In a terminal, write via command line in the folder, type 'python -m SimpleHTTPServer 8000", and leave the terminal open. This will host a local web server in the folder that you can access.
5. Open a browser (Chrome is recommended) and visit the URL "localhost:8000" to see your visualization!

To close the local server, simply close the server you opened in step 4. Then, delete the files generated in Step 3 to have the directory clean for a new visualization.

**Format for CSV File Datasets**

| **Column Index** | **Column Name** | **Expected Data Type** | **Value Domain** | **Description** |
|---|---|---|---|---|
| 1 | ID | Integer | [0…$n_{members}$-1] | Unique identifier for each member |
| 2 | Target | Integer | [0…$n_{class}$-1] | Target class for each member (Duplicates allowed) |
| 3 | Target_names | String | Alphanumerical | Names of all target classes. Only $n_{class}$ different names exist. |
| 4+ | Feature titles | Number | Number | Numerical value for each feature |

**Figure 12:** Structure of CSV file dataset compatible with Decision Visualizer



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | ID | target | target_names | mean radius | mean texture |
| 2 | 0 | 0 | malignant | 17.99 | 10.38 |
| 3 | 1 | 0 | benign | 20.57 | 17.77 |
| 4 | 2 | 0 | | 19.69 | 21.25 |
| 5 | 3 | 0 | | 11.42 | 20.38 |
| 6 | 4 | 0 | | 20.29 | 14.34 |
| 7 | 5 | 0 | | 12.45 | 15.7 |
| 8 | 6 | 0 | | 18.25 | 19.98 |
| 9 | 7 | 0 | | 13.71 | 20.83 |
| 10 | 8 | 0 | | 13 | 21.82 |
| 11 | 9 | 0 | | 12.46 | 24.04 |
| 12 | 10 | 0 | | 16.02 | 23.24 |
| 13 | 11 | 0 | | 15.78 | 17.89 |
| 14 | 12 | 0 | | 19.17 | 24.8 |
| 15 | 13 | 0 | | 15.85 | 23.95 |
| 16 | 14 | 0 | | 13.73 | 22.61 |
| 17 | 15 | 0 | | 14.54 | 27.54 |
| 18 | 16 | 0 | | 14.68 | 20.13 |
| 19 | 17 | 0 | | 16.13 | 20.68 |
| 20 | 18 | 0 | | 19.81 | 22.15 |
| 21 | 19 | 1 | | 13.54 | 14.36 |
| 22 | 20 | 1 | | 13.08 | 15.71 |
| 23 | 21 | 1 | | 9.504 | 12.44 |

**Figure 13:** Example of format required in CSV file shown in Microsoft excel

The user will have to upload their dataset in the form of **comma-separated values.** Figure 12 showcases the format needed for the visualization. Figure 13 is an example csv file that is compatible with decision visualizer. The first row MUST contain the column titles.

The 1st column must contain a sequence of integers from [*0…n-1*] where *n* is the unique identifier for members in the dataset. The 2nd second column must contain target class values for each member. Each integer is considered a different class. In the figure 13 example, the target classes are 0 and 1 for malignant and benign respectively. The 3rd column must contain the names of the target classes starting from 0. The number of target names must equal be the max integer value in target column. Columns 4 and above are for the features of dataset. There is no limit to the number of features of the dataset. All values in the features of the dataset must be numerical. Writing dimensions in the feature column titles are recommended, as the feature column names will be shown in the interactive decision tree.

## References

Chikalov, I., & Lakhmi , J. C. (2011). Average Time Complexity of Decision Trees. Intelligent Systems Reference Library.

Chunmei Liu, D. C. (2013). Applications of Machine Learning in Genomics and Systems Biology. *Computational & Mathematical Methods in Medicine*, 1-1.

Ericcson, K. A., & Hastie, R. (1994). Contemporary Approaches to the Study of Thinking and Problem Solving. In R. Sternberg, *Thinking and Problem Solving* (pp. 37-80). San Diego, CA: Academic Press.

Kubat, M. (2015). An introduction into Machine Learning. In M. Kubat, *An introduction into Machine Learning* (p. 1). Springer.

Lichman, M. (2013). UCI Machine Learning Repository. Irvine, CA, USA: University of California, School of Information and Computer Science.

Mitchell, T. (1997). Decision Tree Learning. In T. Mitchell, *Machine learning.* New York: McGraw-Hill.

Pedregosa, F. (2011). Scikit-learn: Machine Learning in Python. *JMLR 12*, 2825-2830. Retrieved from http://scikit-learn.org.

Sedig, K., & Parsons, P. (2013). Interaction Design for Complex Cognitive Activities with Visual Representations: A Pattern-Based Approach. *Transactions on Human-Computer Interaction*, 84-133.

Wale Akinfaderin. (2016, July 21). *The Mathematics of Machine Learning*. Retrieved from IBM Data Science Experience: http://datascience.ibm.com/blog/the-mathematics-of-machine-learning/