Building a Visualization to Educate Graduate Students on Decision Trees

Vincent Chiu

Visual Analytics

Final Project Report

Table of Contents

Overview

I created a visualization to enable graduate students to learn about and gain a deeper understanding of decision trees. A decision tree is a model that can be used for classification. It can classify a given row of input data based on whether or not the attributes (columns) of that row evaluate to true or false for some decision (rule). In order to make predictions, a sample row fed to a decision tree traverses the branches of the tree based on various true and false rules. When the sample reaches a leaf node, it is classified as the majority class of that leaf node.

A good decision tree is one that has a low classification error rate. In order to have a low classification error rate, it is preferable to have decision nodes that are as "pure" (homogenous) as possible meaning that the node mainly contains samples of the majority class and as few samples of other classes as possible. In order to train a good decision tree, one can employ a greedy algorithm that chooses attribute splits based on picking the attribute with the lowest Gini Split Score. Picking to split the rows based on the lowest Gini Split Score means that the children nodes of the split will be as "pure" as possible. My visualization aims to teach students about decision trees, especially the training (construction) process.

Video:

Video on this visualization can be found at: https://youtu.be/iDN832_sIE0

Introduction

Scope: This project is designed to help graduate students in computing sciences understand decision trees. The user is assumed to have some rudimentary understanding of decision trees already from reading a textbook or article on the topic. This visualization will solidify their understanding of the construction of a decision tree.

Users/Audience: The target audience is master's or PhD students in computing science. Other people may also find this visualization helpful, but they are not the main audience.

In Enrico Bertini and Denis Lalanne's paper, *"Surveying the complementary role of automatic data analysis and visualization in knowledge discovery"*, they mention the idea of Black-Box Integration with feedback Loop. This is a technique in which the data mining or machine learning algorithm is presented as a black box, but the user is able to adjust the hyper-parameters while the model performance and hyper-parameters are being visualized.

The human can formulate a strategy for optimizing hyperparameters through the feedback provided by the visualizations (Chiu).

This is a powerful method for helping students learn. The ultimate goal of project would be to create a tool that allows students to use a popular and widely available machine learning tool such as scikit-learn to generate decision trees and use the trained model as output to generate pedagogical visualizations.

Sample Problem/Domain Questions

The audience may ask questions such as:

- How does a greedy algorithm like Decision Trees choose the best attribute split?
- How do training examples "flow" through the nodes?

Data

The data is from https://www.kaggle.com/uciml/zoo-animal-classification and consists of two comma separated files.

- The first file, zoo.csv consists of 101 animals from a zoo where each row represents one kind of animal. The class types are Mammal, Bird, Reptile, Fish, Amphibian, Bug and Invertebrate. There are 16 attributes that can be used to describe and separate the animals.
- The second file, class.csv, shows the actual word description of the class as well as the string representation of the class type.

Dimensions:

Zoo.csv: 101*18
Class.csv: 7*4

Processing:

I put 67 out of 101 animals into a training dataset. I then used this training dataset to train a machine learning model via scikit-learn and provide the model output in DOT format. The remaining animals were reserved as my test dataset. I also created my own functions that calculated the Gini Split Score for each attribute candidate at every node. A considerable amount of effort was also put into converting the DOT format that scikit-learn outputs into JSON that would be compatible with d3.js or treant.js.

The preprocessing code can be found at:

https://github.com/chiu/decision-tree-visualizer/blob/master/preprocessing/create_tree/decision_tree_preprocessing-Copy12.py

Visualization Design

V1.0

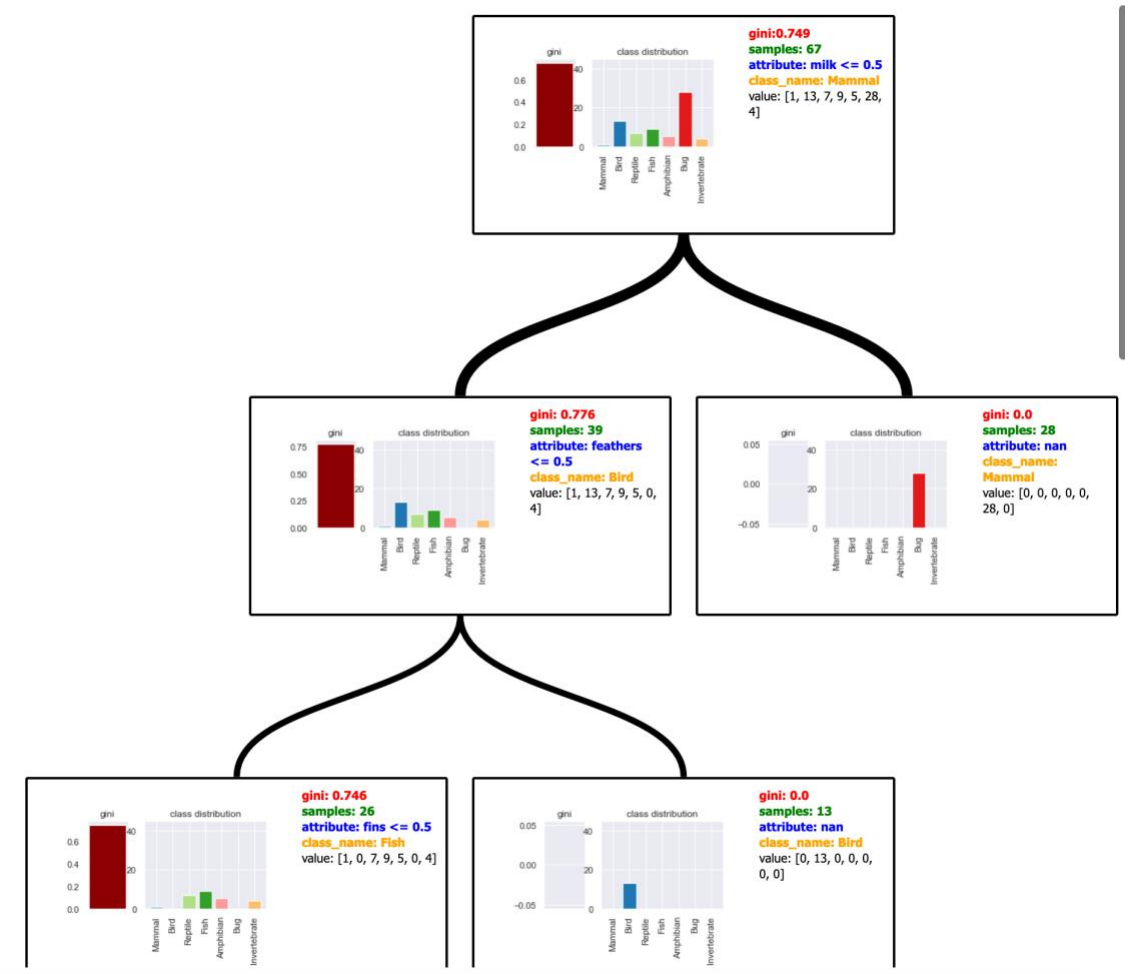

Figure V1: The first version of my visualization

Based on feedback I got from Dr. Bartram, Mr. Mazraeh, classmates, and friends, having the class distribution plots inside each node took up too much space and made it difficult to see the overall structure of the tree. The class distribution plots were also too small which made them hard to read. The first visualization was built with the Treant.js framework.

Furthermore, due to constraints of the Treant.js framework, both links from a given parent had to be the same width. The proper proportional widths of each link could not be accurately represented.
Users liked the general idea behind the visualization, and the use of a node-link diagram. They also liked that the widths of the links were related to the number of samples. They also found the use of vertical bar charts for representing the class distribution to be very intuitive.

V2.0

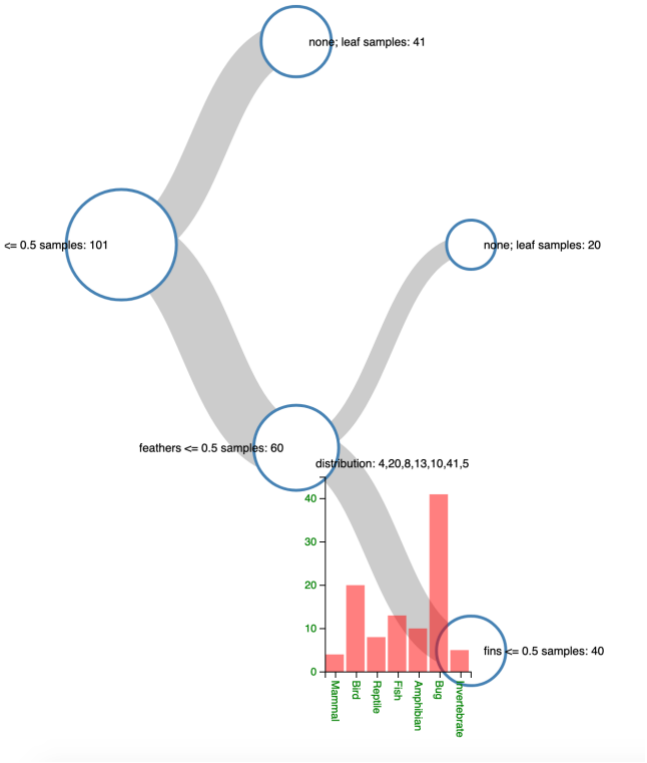

Figure V2a: Collapsible tree on mouseover.

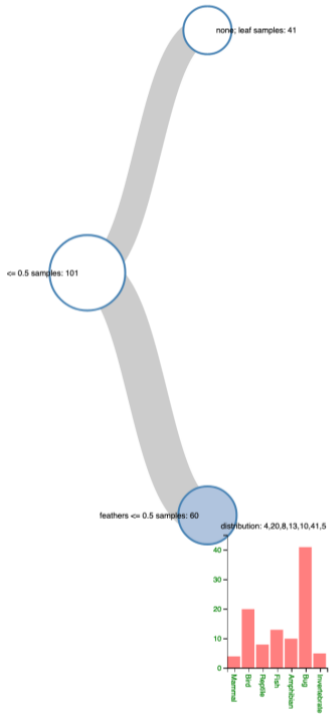

Figure V2b: Collapsible tree on click.

Based on feedback from v1.0, I rewrote my entire visualization. The newer version had circles representing each node. The nodes were also collapsible. If a parent node was clicked, all of its children would disappear.

Participants who tried this version of the visualization noted that the more compact nodes made it easier to see and understand the overall structure of the tree compared to the previous version. However, the class distribution of nodes could no longer be seen without hovering over a node.

Hovering over a node showed a popup with the class distribution. The area of each node was proportional to the number of samples in that node. The links connecting nodes were also proportional to the number of samples flowing between them. I also chose a horizontal

view, for space efficiency. As most computer screens nowadays are widescreen, having a horizontal layout should allow more layers of the decision tree to be displayed simultaneously.

However, according to some study participants, this was confusing because decision trees are traditionally represented vertically in textbooks and in the literature. Other feedback stated that the transparent background of the mouseover plot made it hard to see when it overlapped with the nodes below. Another user suggested that the mouseover functionality was not useful in aiding understanding and that the clicking action could be better reserved for more interesting functionality.

V4.0

Based on all the previous feedback, I decided to create a new visualization almost entirely from scratch using d3.js (Bostock). I started off with a simple template by Sven H. The original template has a simple tree layout with mouseover tooltips.

This version of the visualization has a top-down layout, which is consistent with textbooks and hence easier to understand.

To help students understand the greedy algorithm of choosing the best split attribute, I created an attribute view on the right that appears when a given node is clicked. This view shows potential class distributions of the children of the clicked node, if various split attributes are chosen. The potential class distributions are ordered from the lowest Gini split score (best) to the highest Gini split score (worst). The attribute split with the lowest Gini split score is the one that is chosen in the actual decision tree. The left bar chart represents the left attribute value and the right bar chart represents the right attribute value.

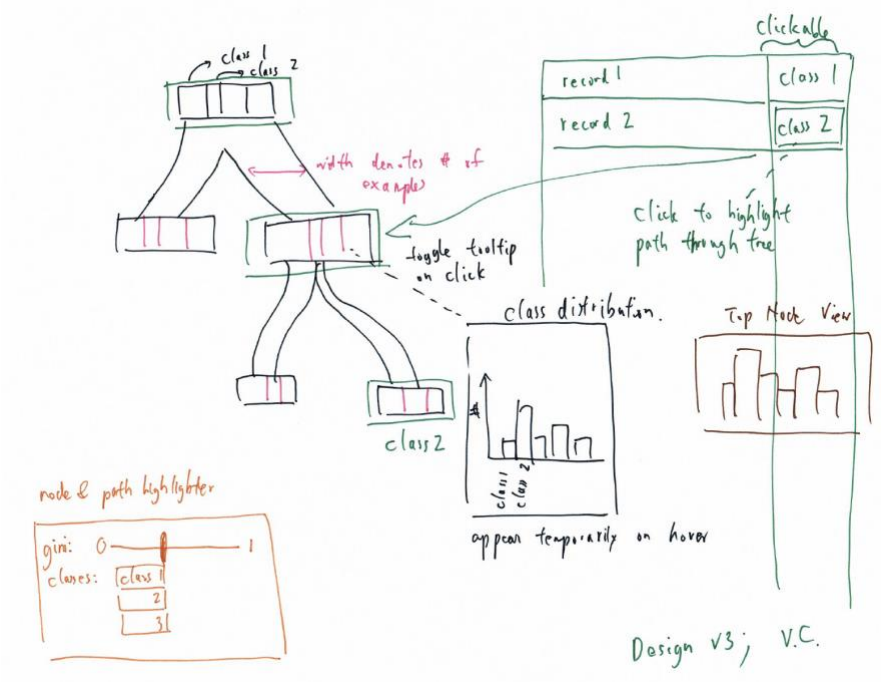

Figure V4a: The mockup for version 4 of my design (using d3).

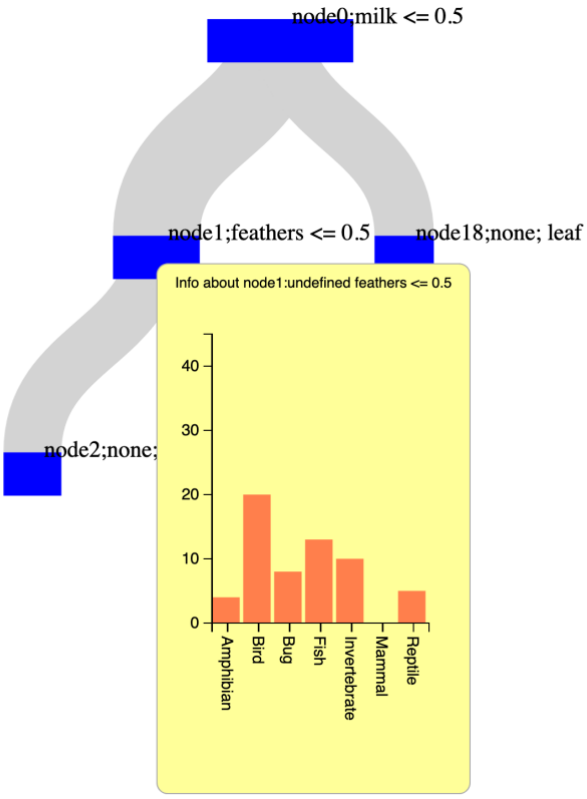

Figure V4b: The refined version, on mouseover.

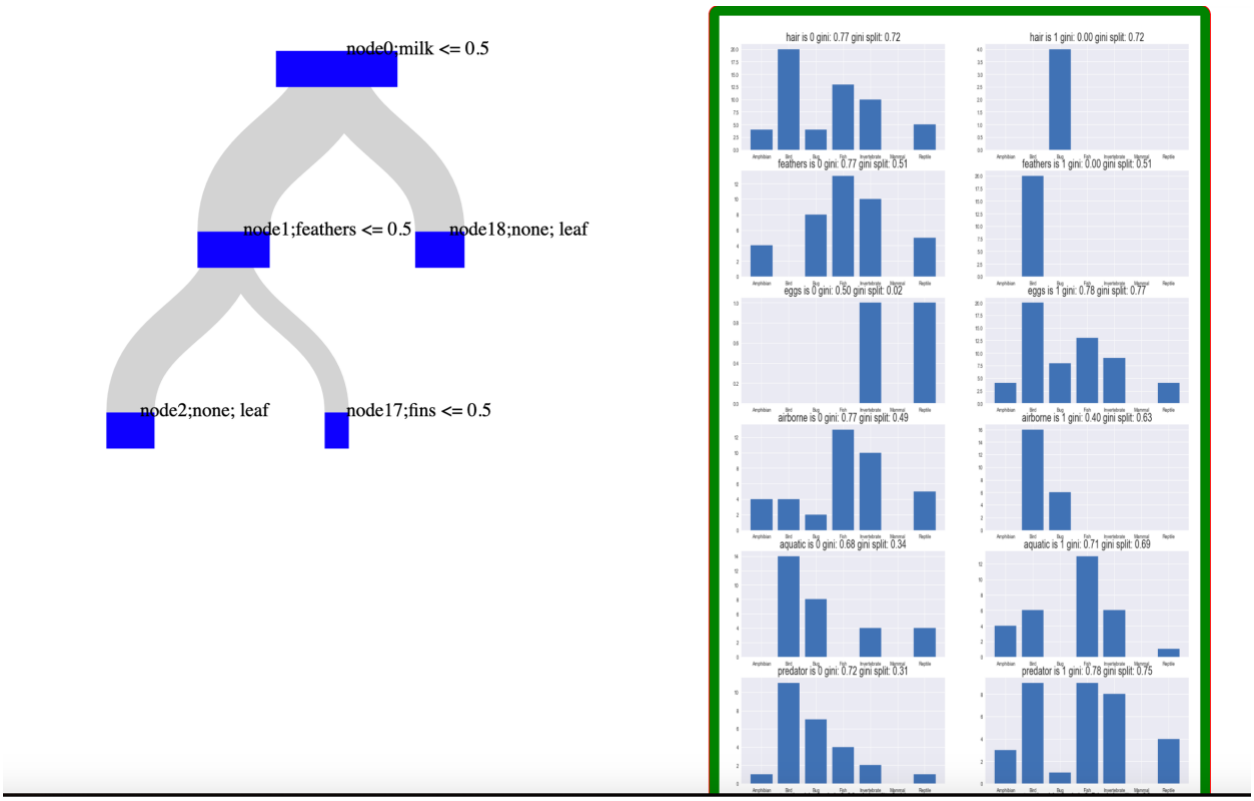

Figure V4c: The refined version, on click.

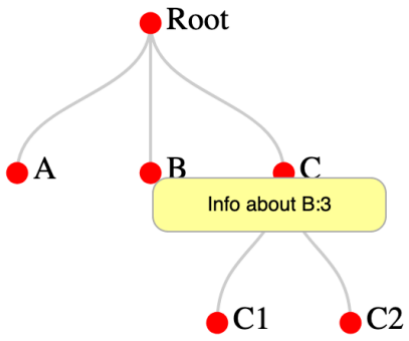

Figure V4d: The code template I began with.

I received the following feedback on this version:

1. On the right view in the Gini Split view, the user should only see the best Gini score, the worst, and 2 in between. This Gini Split Attribute View should also be collapsible. There were too many Gini Split plots which made it overwhelming for new users. Having a representative sample would aid in understanding.
2. Nodes should not be bright red as that is color that is best reserved for highlighting.

3. Having one tornado chart instead of 2 vertical bar charts per attribute would make it easier to compare the number of animals for a given class.

V5.0 (Newest Design)

Deployed Version

The project can be found in the attached zip, please decompress the zipped folder and click on index.html to open the visualization.

A live version is available through the following link:
https://codepen.io/vchiu/project/full/DObPvr

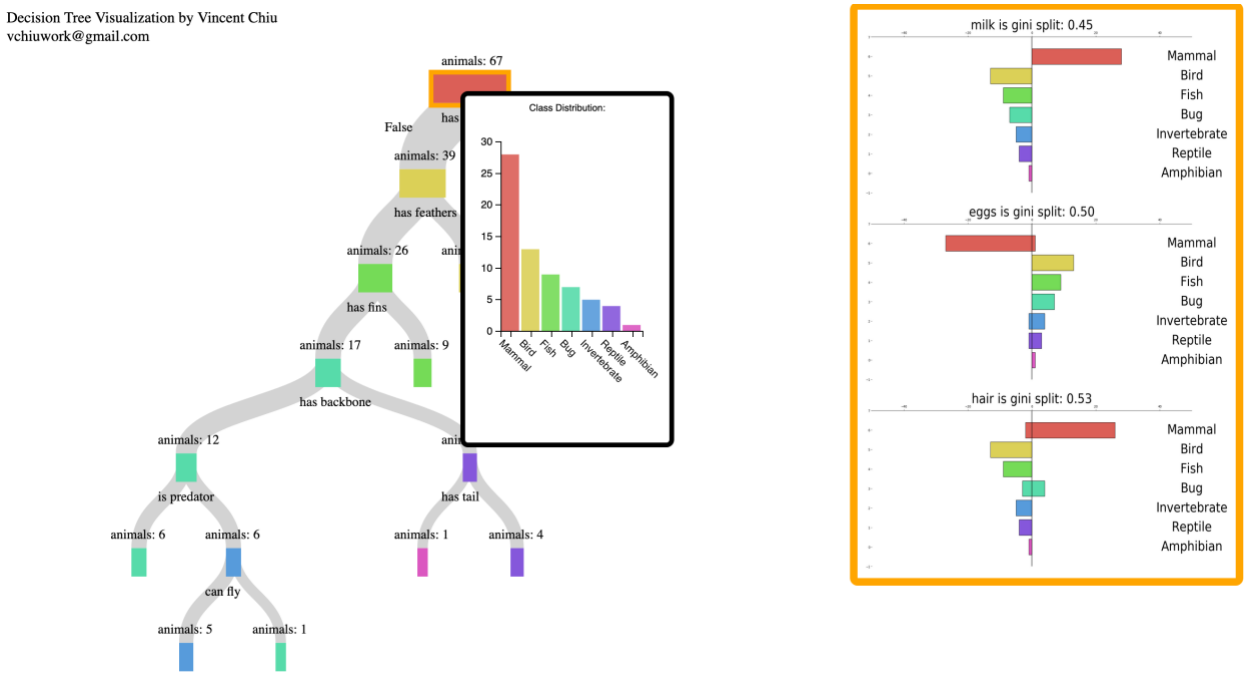

Figure V5a: The Gini Split scores are shown on click.
In this case, the root node was clicked. You can see that milk is the best attribute split. The charts are ordered from the lowest Gini Split score (best) to the highest Gini Split score (worst).

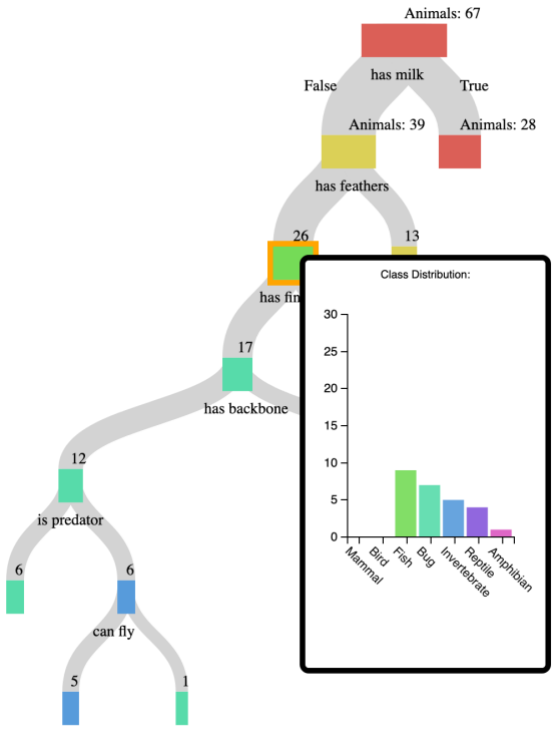

Figure V5b: Mouseover on any node to see class distribution for that node.

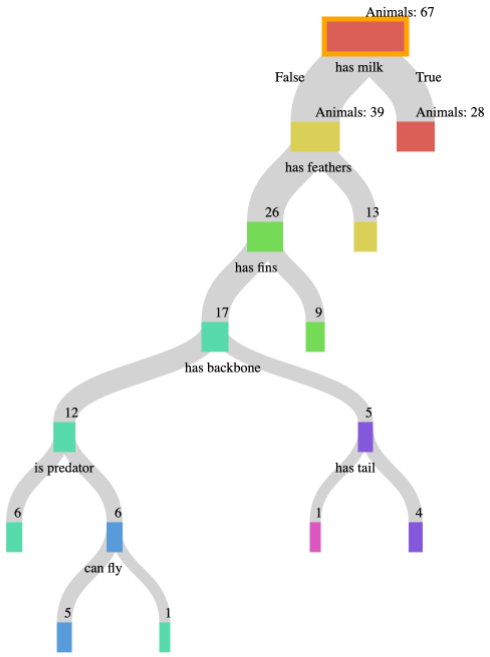

Figure V5c: Entire view of tree without mouseover.

User Manual

1. Hover on the rectangle representing each decision node to see the class distribution for that node.
2. Click on the rectangle representing each decision node to see the candidate attribute splits on the right. The best attribute split, the worst and one in between are shown. The rectangle that was most recently clicked has an orange border.
3. A tornado chart illustrates the intuition behind the calculation of the Gini Split score. A tornado chart is a horizontal bar chart where the left side represents the class distribution for the left attribute value and the right side represents the class distribution for the right attribute value. The Gini Split Score is a weighted average of the left Gini score and the right Gini score.
4. I used a node link representation for my decision tree. The width of the links represents the number of row or samples going into a particular node. The width of the node represents the number of samples in that node.

Visual Mapping

The width of each node corresponds to the number of training samples going into that node. The width of the links represents the number of training examples flowing through it between nodes. The color of a node corresponds to the majority class in that node.

The design of this visualization was partially inspired by the paper "Baobabview: Interactive construction and analysis of decision trees" (Elzen et al.) I decided to use a node-link diagram to represent a decision tree because in that paper, the authors reference Barlow and Neville who have done studies which conclude that node-link diagrams are the best visualization for conveying tree topology based on user preference (Elzen et al.)

In this new design, tornado charts replaced vertical bar charts. The new design allows users to easily compare the number of animals for each type. This aids in their understanding of Gini Split Score. They can see that attributes that more cleanly separates classes correspond to a lower and better Gini Split Score.

Width of nodes was used to represent the number of samples as opposed to area because humans are better at determining relative sizes of widths compared to area.

The colors representing each animal type are chosen such that all colors are of similar intensity and no color stands out too much compared to others. This is because each bar represents a qualitative category and no category should stand out too much compared to others.

Feedback and Future Plans

I received positive feedback saying that my visualization paints a compelling and intuitive story about how decision trees are constructed. Users also liked the use of tornado charts and noted that it was a considerable improvement compared to vertical bar charts. Users also found the use of width as a representation of the number of samples to be intuitive.

Based on feedback, I am considering making additional visualizations with different datasets so that the audience will have more visualizations to help them understand decision trees. I have also received feedback that I could add information about the remaining available attributes for each split.

Conclusion

I created a visualization that is very effective at helping graduate students understand decision trees, according to user feedback. I used various visualization science principles to inform my design. I made multiple improvements to my implementation through an iterative and agile design process, with multiple feedback cycles. My visualization gives users the power to open a black box model such as decision trees and look inside. My design is effective and accessible because I designed it with visualization principles in mind. I have helped prove that visualizations have great power in helping humans understand machine learning models.

References

1) University of California Irvine, Zoo Animal Classification. (2019, April 15). Retrieved from https://www.kaggle.com/uciml/zoo-animal-classification

2) Vincent Chiu, Using Visualizations to Facilitate Human-Machine Collaboration in Data Mining. (2019, April 16).

3) Bertini, E., & Lalanne, D. (2009). Surveying the complementary role of automatic data analysis and visualization in knowledge discovery. ACM.

4) van den Elzen, S., & van Wijk, J. J. (2011). BaobabView: Interactive construction and analysis of decision trees. 2011 IEEE Conference on Visual Analytics Science and Technology (VAST), 151–160. doi: 10.1109/VAST.2011.6102453

5) Bostock, M. (2019, March 22). D3.js - Data-Driven Documents. Retrieved from https://d3js.org

6) Treant.js - javascript library for drawing tree diagrams. (2017, August 07). Retrieved from https://fperucic.github.io/treant-js

7) Sven. H, Simple static tree with tooltips. (2015, October 06). Retrieved from http://bl.ocks.org/anotherjavadude/2952964