# Exploring the Impact of Decision Tree Depth

Alic Szecsei
University of Iowa
alic-szecsei@uiowa.edu

Diego Castaneda
University of Iowa
diego-castaneda@uiowa.edu

Willem DeJong
University of Iowa
willem-dejong@uiowa.edu

## ABSTRACT

Lobortis scelerisque fermentum dui faucibus in ornare quam viverra orci sagittis eu volutpat odio facilisis mauris sit amet massa vitae tortor condimentum lacinia quis vel eros donec ac odio tempor orci dapibus ultrices in iaculis nunc sed augue lacus viverra vitae congue eu consequat ac felis donec et odio pellentesque diam volutpat commodo sed egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium vulputate sapien nec sagittis aliquam malesuada bibendum arcu vitae elementum curabitur vitae nunc sed velit dignissim sodales ut eu sem integer vitae justo eget magna fermentum iaculis eu non diam phasellus vestibulum lorem sed risus ultricies tristique nulla aliquet enim tortor at auctor urna nunc id cursus metus aliquam eleifend mi in nulla posuere sollicitudin aliquam ultrices sagittis orci a scelerisque purus semper eget duis at tellus at urna condimentum mattis pellentesque id nibh tortor id aliquet lectus proin nibh nisl condimentum id venenatis a condimentum vitae sapien pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas sed tempus urna et pharetra pharetra massa massa ultricies mi quis hendrerit dolor magna eget est lorem ipsum dolor sit amet consectetur adipiscing elit pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas integer eget aliquet nibh praesent tristique magna sit amet purus gravida quis blandit turpis cursus in hac habitasse platea dictumst quisque sagittis purus sit amet volutpat consequat mauris nunc congue nisi vitae suscipit tellus mauris a diam maecenas sed enim ut sem viverra aliquet eget sit amet tellus cras.

## CCS CONCEPTS

• **Computing methodologies → Machine learning**; *Supervised learning by classification*; *Classification and regression trees*; Cross-validation.

## KEYWORDS

decision trees, model selection

## 1 BACKGROUND AND MOTIVATION

Machine learning algorithms fall into one of two categories: classification and regression. In regression, input data is consumed and transformed by a function which can produce values along a numeric range; in classification, data is categorized into a finite number of possible values, and machine learning algorithms seek only to label data sets. These classification problems can use both numeric and categorical data attributes. For purely categorical data, decision trees are a way to build machine learning algorithms that are understandable by both humans and machines. Rather than a sort of "black box" in which numbers are passed in and answers are returned, there is a logical tree hierarchy that is familiar to anyone who has played the game Twenty Questions.

Decision trees require both a way to construct them, and a way to use them to categorize data. This categorization is a trivial tree traversal, and so many of the innovations regarding decision trees has to do with their construction. Most decision tree construction algorithms use a two-phase approach: first a *growing* phase, followed by a *pruning* phase. In the growing phase, the decision tree is built out, trying to fit the provided training data as closely as possible. To combat over-fitting, the pruning phase determines which branches of the tree are too "noisy" using $\chi^2$ tests and removes them.

Additionally, decision tree models can be constrained by size to combat overfitting. Russell and Norvig [6] showcase an implementation of restricting a decision tree to be beneath a maximum size by generating the tree in breadth-first fashion, and stopping when the maximum number of nodes has been reached. As stated in Garofalakis, Hyun, Rastogi, and Shim [3], there is no point in creating a branch when it is guaranteed to be pruned later.

The amount of pruning which occurs is heavily dependent on the data set chosen. As such, the exact values which optimize our trained decision trees are relatively unimportant. However, our goal was to examine how impactful depth-based pruning was, and how much it assisted with reducing overfitting.

## 2 METHODS

### 2.1 Dataset Selection

Our first step was selecting datasets to use as the foundation for our programs. We used four UCI data sets [2]:

(1) `MUSHROOMS`, a dataset to classify mushrooms as edible or inedible
(2) `BALANCE`, a dataset to classify whether or not a scale with objects of varying weights and distances was balanced
(3) `CARS`, a dataset of car evaluations
(4) `TICTACTOE`, a dataset of Tic-Tac-Toe board states.

These datasets were selected as they have a variety of both number of attributes and number of examples. `MUSHROOMS` has 8124 examples with 22 attributes, for example, while `BALANCE` only has 625 examples and 5 attributes. This variety ensured that our algorithm testing would not be skewed by certain kinds of datasets.

### 2.2 Decision Tree Generation

The foundation of our decision tree algorithm was the one provided by Russell and Norvig [6] which, in turn, is based on the ID3 algorithm [5].

A two-program approach was used: first, a decision tree generator was developed, called `dtl`. This program would take in a flag to determine which of our data sets was represented by the input. As the ordering of attributes and classes was inconsistent between datasets, this allowed us to significantly simplify the process of importing data.

**Function** DecisionTreeLearning(*examples*, *attributes*, *parent_examples*, *depth*)

```
if examples is empty then
│ return Plurality-Value(parent_examples)
else if depth = 0 then
│ return Plurality-Value(examples)
else if all examples have the same classification c then
│ return a leaf node c
else if attributes is empty then
│ return Plurality-Value(examples)
end
A ← argmax_{a∈attributes} Importance(a, examples);
tree ← a new decision tree with root test A;
foreach value v_k of A do
│ exs ← {e : e ∈ examples and e.A = v_k};
│ subtree ←
│   DecisionTreeLearning(exs, attributes - A, examples, depth - 1);
│
│ add a branch to tree with label (A = v_k) and subtree
│   subtree;
end
return tree;
```

## 2.3 Classification

A second program, `classify`, was used to classify data using a decision tree generated from `dtl`. To communicate a decision tree between these two programs, a JSON representation of the decision tree was used. This data format allowed us to produce human-readable representations of the decision trees, which we could use as a sanity check during development.

## 2.4 External Packages

To assist with the command-line interface, we used the `cmdliner` module. This allowed us to define data types and arguments for input, and then it handled the majority of the command-line parsing. It also automatically generated a `--help` option to inform users of what the different options were. This massively simplified developing optional new features.

Additionally, we used `adtgen` to generate a JSON serializer and deserializer. This was done at the very start of the project, allowing us to work with and visualize decision tree data from the very beginning. This also meant that as our decision tree model was updated and changed, the JSON code was automatically shared and kept up-to-date between `dtl` and `classify`.

## 2.5 $k$-fold Cross-Validation

Finally, we used Python to generate data for $k$-fold cross-validation for both our decision tree program as well as the decision tree algorithm provided by scikit-learn [4]. This provided a simple baseline for our experimentation.

We then ran each of our datasets through the DecisionTreeLearning algorithm with maximum depths ranging from 1 to 10 and performed $k$-fold cross-validation to determine error rates, with

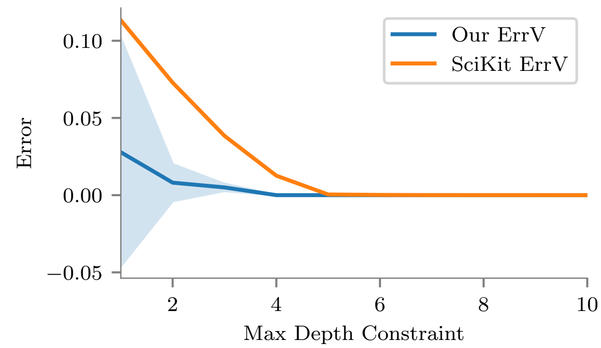


**Figure 1: Error ($\pm 2 \times \sigma$) for our algorithm versus scikit-learn on the `MUSHROOMS` dataset**

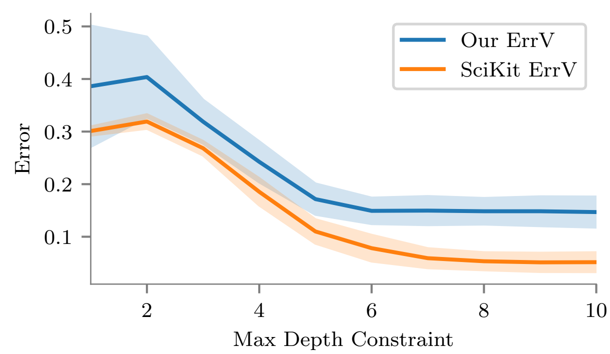


**Figure 2: Error ($\pm 2 \times \sigma$) for our algorithm versus scikit-learn on the `TICTACTOE` dataset**

$k = 4$. As we noticed that subsequent experiments could obtain dramatically varying error rates, we repeated each of these experiments for 100 trials, to determine both average error rates and the variance of these error rates.

# 3 RESULTS

The results of our experimentation varied quite a bit between datasets. For `MUSHROOMS` in particular, our algorithm was able to achieve significantly greater accuracy than scikit-learn for smaller maximum depth constraints (figure 1). However, for other datasets, such as `TICTACTOE`, our algorithm produced consistently more error-prone results at all depth values (figure 2).

The `CARS` dataset proved the most interesting. While our algorithm had worse accuracy than scikit-learn for small depth values, as the allowed depth increased we began to outperform scikit-learn's algorithm (figure 3).

Overall, we found that our algorithm performed much worse on datasets with a large amount of interconnected data, such as Tic-Tac-Toe, where the impact of one attribute was heavily dependent on all of the other attributes. However, for more traditional classification

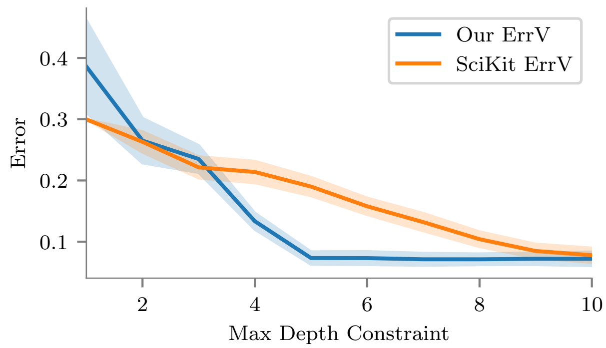


**Figure 3: Error ($\pm 2 \times \sigma$) for our algorithm versus scikit-learn on the `CARS` dataset**

problems, such as evaluating cars or mushrooms, our algorithm was very accurate.

## 4 DISCUSSION

We were surprised at the discrepancies between our algorithm and the one provided by scikit-learn, as we expected both decision tree algorithms to produce similar results. While an effort was made to produce similar constraints on both algorithms (scikit-learn was used with the "entropy" heuristic, as well as the same tree depth constraint), there were a few differences that we were unable to resolve. First, scikit-learn uses "an optimised version of the CART [1] algorithm."

Additionally, scikit-learn cannot be used with categorical data. To get around this restriction, we used one-hot encoding for the scikit-learn algorithm. This has a negative impact on the depth: scikit-learn's decision tree is forced to be a binary tree, and so depth constraints act far more harshly than they do on our algorithm.

## 5 ROLE ASSIGNMENT & CONTRIBUTIONS

### 5.1 Alic

Alic was responsible for implementing the decision tree learning algorithm, the JSON decision tree representation, helped improve the binary executables, and implemented unit testing for several core functions. He implemented the decision tree learning algorithm from Russell and Norvig [6], and added an optional depth constraint. He also was responsible for constructing both the internal decision tree representation and determining the JSON representation for communicating between the `dtl` and `classify` binaries. Additionally, he improved the binary executable command-line experience, enabling order-independent options as well as a `--help` option. Finally, he wrote unit tests for core functions such as `Plurality-Value`, `Entropy`, and `DecisionTreeLearning` itself. He was the project checker.

### 5.2 Willem

Willem worked primarily on decision tree creation, data file read in, and model selection in ocaml. He created a data reader for CSV files, which was used by all of the OCaml binaries (`dtl`, `classifier`, and the OCaml-based model selection program). He helped program the `dtl` executable, which reads in a data file for the training data, then creates a decision tree. He also wrote up the large amount of hard coded information about the different sets of data, including information about the attributes, their possible values, the possible classification, etc. He also created the OCaml version of model selection using cross validation, which takes in various optional and required command line arguments and prints out the selected tree, error rate in the training set, error rate in the validation set, and the chosen max depth. He also elected to be the project recorder.

### 5.3 Diego

## ACKNOWLEDGMENTS

This work was supported in part by the University of Iowa.

## REFERENCES

[1] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees.* Wadsworth.
[2] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml
[3] Minos Garofalakis, Dongjoon Hyun, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Constructing Decision Trees with Constraints. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00).* ACM, New York, NY, USA, 335–339. https://doi.org/10.1145/347090.347163
[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[5] J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (01 Mar 1986), 81–106. https://doi.org/10.1007/BF00116251
[6] Stuart J. Russell and Peter Norvig. 2010. *Artificial intelligence: a modern approach* (3rd ed.). Prentice Hall.

## A SUPPLEMENTAL GRAPHS & DATA

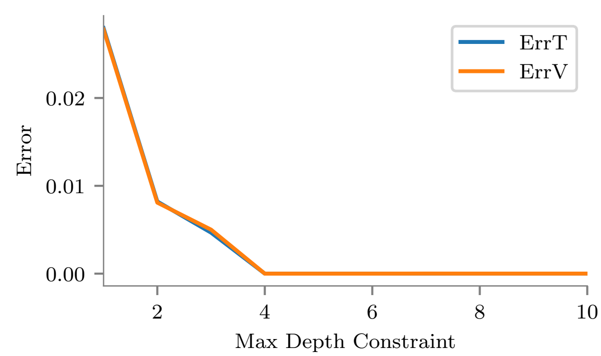


**Figure 4: The error on training and validation data for the `MUSHROOMS` data set**