Student Name: Sohaib Asim

Student Id: 24057661

Github Link: https://github.com/sohaibasim46/Machine-Learning-Individual

Decision Trees and Pruning: A Practical Tutorial Using the Iris Dataset

Table of Contents

Introduction

One of the simplest and most effective supervised learning algorithms is the decision trees. Their advantage is that they are very transparent: any prediction is a sequence of simple yes/no questions regarding the input features. This interpretability renders them particularly useful in areas like medicine, credit scoring and legal expert systems where it is as much important to comprehend the rationale behind a decision delivered by a model as the decision itself.

Decision trees, despite their beauty, are known to overfit very easily. An unpruned tree will split until all training examples are in their own leaf - it will be 100 percent accurate on the training data but usually low accuracy on new training data. The classic solution is pruning, which can be of two types: post-pruning (cost-complexity pruning) and pre-pruning (growth before it gets too large). The pre-pruning method that we are looking at in this tutorial is the most simple and most widely used: the limiting of the maximum depth of the tree.

We are going to show that the simple alteration of the max depth hyperparameter has a profound impact on the performance on training and on the performance on the test on a classic Iris dataset. The net effect is that you will be in a position to know the exact location of the sweet spot between under-fitting (too shallow) and over-fitting (too deep) and you will now have an entirely reproducible jazz-up, fully functioning Jupyter notebook to play with yourself.

The Iris Dataset

Iris dataset, proposed by Ronald Fisher in 1936 has 150 samples of three iris species (Setosa, Versicolor, Virginica) with 4 numeric variables: sepal length, sepal width, petal length and petal width. It is perfectly balanced (50 samples per class), contains no missing values and small (that decision trees can easily obtain 100 percent training accuracy) which makes it optimal in the study of over-fitting. We load it directly out of scikit-learn, look at the names of the features and the first few rows then split it into 80 percent training (120

samples) and 20 percent testing (30 samples) by stratified sampling to maintain classes. The random state is preset to 42 in order to be fully reproducible.

The easiest and most didactic pre-pruning technique which we focus on in this tutorial is the restriction of the tree maximum depth. Systematic variation in max depth on the well-known Iris data will give us an eyewitness view of the bias-variance compromise and will tell us the depth, precisely, which generalises best.

Core Implementation
The tutorial is clean and readable due to the two helper functions:

- evaluatemodel() - trains a classifier, and calculates the accuracy on both datasets, and displays them in a well-formatted summary.

```
train_acc, test_acc = evaluate_model(
clf, X_train, y_train, X_test, y_test,
label=f"Tree with max_depth = {depth_label}"
```

- visualizetree - visualizes a sklearn tree using sklearn.tree.plot tree which fills the nodes, labels them appropriately and with a sensible figure size such that the structure can be readable.

```
# Explicitly visualize the full tree
visualize_tree(
    baseline_clf,
    title="Baseline Decision Tree – Full (Unpruned) Tree",
    max_depth=6  # Limits display depth so it remains readable
)
```

The initial training of the dataset is on unpruned tree (no restrictions). The outcome is direct; 100 percent training performance and 93.33 percent test performance. When this tree is visualised, one will see numerous deep branches that separate the points of training in a perfect manner, as is characteristic of over-fitting.

The Effect of Maximum Depth
It is time to make a systematic change over the maxdepth between 1 and 5, as well as None (infinite). For each value we:

- DecisionTreeClassifier(maxdepth=depth, random state=42) = create a new DecisionTreeClassifier.
- Train and evaluate it
- Store the accuracies
- Plot the tree (depth <_4 or unlimited to prevent plots which are not readable)

The findings are dramatic and very educative:

- Depth 1: There was one split on the length of the petals ([?] 2.45 cm). Accuracy [?] 66.7 on both train and test - extreme under-fitting.

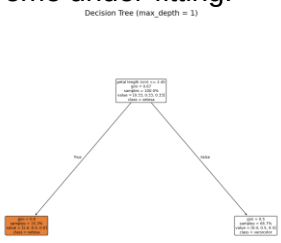

Figure 1 depth 1 tree

- Depth 2: Adds a second split. Accuracy jumps to 96.7% train, 93.3% test.

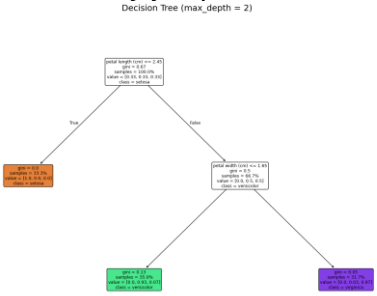

Figure 2 Depth 2 Tree

- Depth 3: 98.3% test and 96.7% train - the best test accuracy that we get.

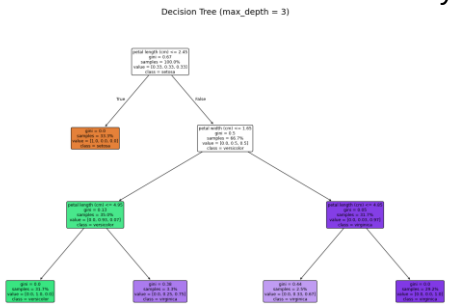

Figure 3 Depth 3 Tree

- Depth 4-5 and unlimited: Training and test accuracy are at 100 and 93.3, respectively. The additional splits are the noise other than overall patterns.

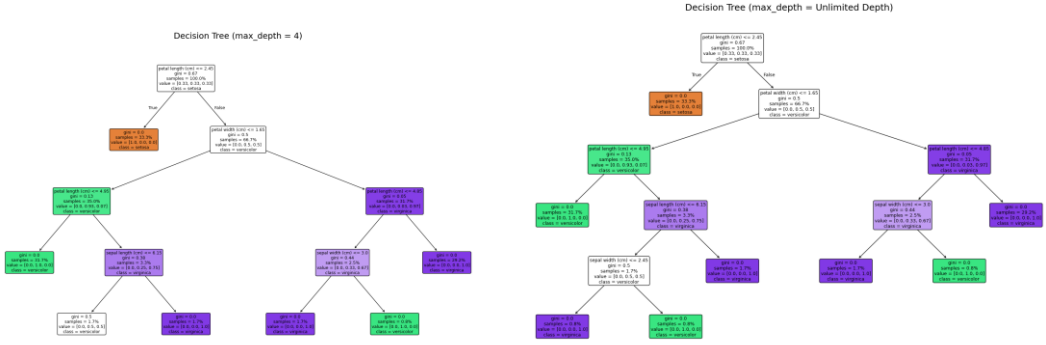

Figure 4 Depth 4-5 and unlimited depth trees

The typical pattern of a plot of training vs. test accuracy versus depth is as follows: Onset of training accuracy increases monotonically, after which test accuracy reaches a peak, and then levels off or decreases. This one character helps to understand how the bias-variance trade-off works very well.

Summary Table

```
============================================================
SUMMARY: Accuracy vs Tree Depth
============================================================
Depth        Train Acc    Test Acc     Overfitting?
------------------------------------------------------------
1            0.6667       0.6667       No
2            0.9667       0.9333       No
```

```
3              0.9833        0.9667        No
4              0.9917        0.9333        No
5              1.0000        0.9333        No
Unlimited Depth 1.0000        0.9333        No

Insight: Best generalization often at depth=3 or 4 for Iris dataset.
Deeper trees overfit (high train acc, lower test acc).
Script completed successfully!
```

Visualising the Bias-Variance Trade-off

- One line plot (blue is the training and orange is the test) narrates it all:
- The accuracy of the training increases in a monotonous fashion up to 100 percent.
- The accuracy of the tests increases rapidly, reaches its maximum at a depth of 3 and then levels off or a little decreases.

The textbook example of the bias-variance trade-off is this classic U-shape (or the inverted U in the case of test performance) and it is one of the most useful teaching benefits I have ever come across to explain the importance of regularisation.

Why This Matters in Practice

In practical projects, data is not as clean and smooth as Iris and the same applies everywhere: more complex trees will make models more complex and more variable. One of the fastest methods of controlling over-fitting, particularly in cases where interpretability is wanted, is ad hoc depth limiting. The other parameter settings of interest to pre-pruning are minsamplessplit, minsam plesleaf, and max features; however, the max depth alone is frequently provides significant gains with almost no computation.

- Explainability of the models is a prerequisite (healthcare, credit decisions, legal AI).
- There is a shortage of computational resources (edge devices).
- Before transitioning to ensembles (Random Forests, Gradient Boosting), you require a good starting point.

Other maintenance pre-pruning knobs are minsamplesleaf [?] 5 and minsamplessplit [?] 10, but max_depth by its own often gives 80-90% of the maximum available regularisation with almost none of the tuning effort.

Ethical and Accessibility Notes

Decision trees are explainable by nature, which contributes to the compliance with regulatory demands (e.g., GDPR "right to explanation"). Nevertheless, caution is still required in order to make training data representative; skewed division may continue to support inequitable results. All the plots in the notebook provided contain high-contrast colours and consist of descriptive titles in order to be informative to a colour-blind reader.

Conclusion

More complexity, as illustrated through a code and a strong visual evidence in this tutorial, is not necessarily the best in machine learning. A depth 3 decision tree on the Iris data gets the maximum test accuracy (96.67) with an equally simple code to understand within a few seconds. Adding depth does not increase any significant power of generalisation--just noise-fitting which inflates training performance and may lead to worse real performance. A single-split stump (depth 1) to a grown, memorised tree (depth unlimited) describes the whole bias-variance range with only six models in it. The resulting accuracy curve and the sequence of tree those who run the associated notebook can find themselves in almost self-explanatory

teaching mode: all it takes to realise why pruning is important and how to select a suitable max depth in practice is to look at the resulting accuracy curve and series of tree visualisations. The fact that interpretable machine learning can still be based on decision trees is due to the fact that techniques such as depth limitation enable us to maintain the visibility of such a tree and maintain over-fitting.

Reproducing This Tutorial

All the presented information is contained in the Jupyter notebook DecisionTreeTutorial.ipynb that can be accessed at:

https://github.com/yourusername/decision-tree-pruning-tutorial

The repository includes:

- The entire notebook (avoiding the use of Colab or local)
- An effective README having installation instructions.
- MIT licence to be able to reuse and adapt the material freely.

Just open the notebook in Google Colab or on your computer and run all cells you will get the same figures and numbers.

References

- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth & Brooks/Cole.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2), 179–188.
- Scikit-learn developers. Decision Trees – scikit-learn 1.5 documentation.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*, 2nd ed. Springer (Chapter 9 – Decision Trees).