Machine Learning for Data Science HW1 Decision Trees

Nikolay Kormushev

Introduction

In this paper, I delve into decision trees and random forests, popular machine learning algorithms for classification and regression. While decision trees offer interpretability and simplicity akin to human decision-making, random forests sacrifice some interpretability for robust performance.

I explore methods to regain interpretability in random forests, such as feature importance. Additionally, I investigate the relationship between feature importance and the frequency of features chosen as tree roots. This study lays the groundwork for understanding feature selection and importance in machine learning models.

Methods

In this section I will describe my implementations of the main methods I used for the decision trees with a focus on the choice of split and calculating feature importance.

Split

The split selection is integral to how a decision tree works. In this part I will look at my implementation.

The approach I went for is an exhaustive search through all potential splitting points. First I choose a feature, then extract the unique values from the corresponding column which I sort for a small optimisation.

Instead of choosing the value of the dataset element as a splitting point I choose the average between it and the next value in the ordered column values. As a result the split is moved in-between the two elements making sure that new values go to the same side of the split as the element they are more similar/closer to be it the original split element or the next one.

To measure the quality of my splits I use the Gini index which measures diversity in a dataset. The lower the weighed sum of the Gini indexes of the resulting datasets the better the split is. The Gini indices are weighed based on the number of elements each dataset contains so if one side of the split has many more elements its Gini index contributes more to the sum making its diversity more important.

I have found an optimal split in the case that Gini index reaches zero and I use that one and terminate the search. If this does not happen I the point with the lowest index is chosen as the split

Importance

Importance is a metric that helps with explainability and understanding of random forests. It shows how sensitive is a feature to being changed with regards to the missclassification error.

To calculate it I first determine the out-of-bag error against our normal unchanged dataset. The out-of-bag error is the mean of misclassificaton rates of each tree run only against its out-of-bag elements i.e. the elements not used to train the tree.

To calculate the importance of each feature I loop through the columns of our dataset and I permute the values of the column which introduces noise to the data. To see how this effects the results I once again compute the out-of-bag error and compare it with the original error by getting the percentage increase using the following formula:

$$\frac{noisyErr - predErr}{predErr + eps} \cdot 100 \tag{1}$$

I add the epsilon as a small value to avoid division by zero which is possible if the model is 100% accurate.

The result of this is a vector of importances of the features.

Testing

I wrote unit tests for each of my methods using the python unittest library. I wanted to test my methods in isolation and to do that I used the Mock object which allowed me to verify function calls were being made with the correct arguments and the correct amount of times.

For integration tests I build either a random forest or a tree. I make a call to its predict method and verify the results. This is done using a simple inputs to ease the validation of the results.

Results

Evaluation

To evaluate my model I used metrics such as execution time, measurement error and standard error. For accurate results I

ran the code 100 times saving the metrics and averaged the results to form my conclusions

The metrics are calculated on my personal laptop using a 12th Gen Intel(R) Core(TM) i5-12500H processor

The metrics were measured using the FTIR spectral data set

Root split vs importance

In Figure 1 is a comparison of the feature importance and the root split frequency i.e. how often a feature was chosen to be a root when making a tree. To test this I created a 100 trees and checked what was split on first.

The figure shows that the two features with the highest importance are the ones most often used as roots of the trees. This shows a correlation between the two metrics.

A possible explanation can be derived by thinking of how decision trees work. Features that are chosen as root splits determine the first decision made to classify a sample. This means if a wrong choice is made during the first decision it is highly unlikely to classify the sample correctly later which leads to high misslcassification rate if the values of that feature are permuted. As a result the feature importance would be high since it measures the increase in wrongly classified samples.

On the other hand if a feature deeper in the tree is changed. The model will misclassify less often because it needs to get to that location in the tree to start making errors and until then the sample might already be classified correctly.

From here a the conclusion can be made that features that are less often chosen as root or even to split would get a lower importance.

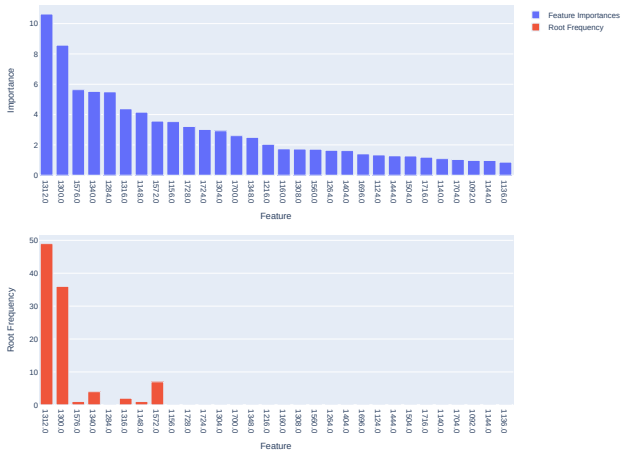

Figure 1. Variable importance vs root frequency

Performance and number of trees

In Figure 2 it can be seen how missclassification rate changes with the number of trees n.

After a certain point the train error stays at zero because the forest overfits on the training set. This does not happen immediately because each tree is built based only on a subset of features.

The test error has big spikes up and down in the beginning. This can be explained by the change of majority votes when adding new random trees. Since at the start there are less trees a new tree has a bigger importance when choosing the modes which leads to big changes in the error. However as the number of trees increases a new tree has a lower impact on the forest shown by the spikes at the end of the graph becoming smaller.

Also after around 60 trees the missclassification rate evens out with small spikes up and down which means that after a certain point adding more trees does not improve accuracy

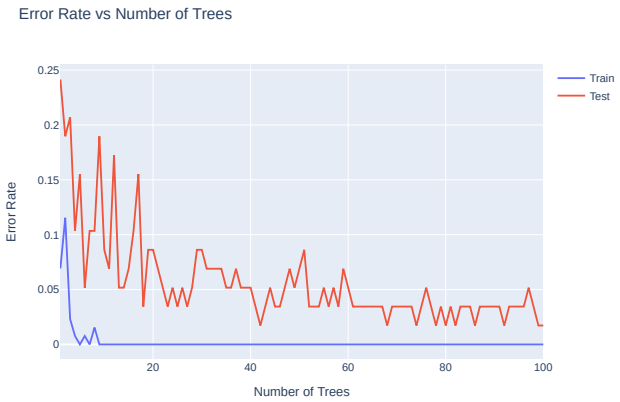

Figure 2. Missclassification rate when increasing tree count

Execution times

In Table 1 build times and missclassification rates of the trees and random forest are displayed. For the importance I have listed the execute time. Time is measured in seconds

Table 1. Measurements table

Name	Missclassification Rate	Execute time
Tree	0.189 ± 0.051	1.755 ± 0.024
Forest	0.028 ± 0.021	12.073 ± 0.614
Importance	NA	6.656 ± 0.296

Conclusion

In conclusion, my exploration of decision trees and random forests has shed light on their strengths and weaknesses. I also discovered that root frequency and feature importance are correlated and that the test error varies a lot for a small number of trees and stays more consistent the bigger the number is. For future improvement I can use parallelization for building the trees in the random forest, choosing the splits and calculating the importance for each feature.