ML–DS I: Project I Report

Sebastijan Trojer
ML-DS I 2024/25 , FRI, UL
st5804@student.uni-lj.si

I. Introduction

The goal of project 1 was to implement a decision tree and a random forest classifier that support numeric input values and a binary target variable. This report goes over the main focus points of the project.

II. Methodology

A. Quantifying the uncertainty

Properly quantifying the uncertainty of a model is a vital step to ensure appropriate model evaluation. In our case, we used bootstrap sampling to ensure more accurate results. After building the model we acquired the test set predictions, determined whether the prediction for each row was correct or not and applied bootstrap to the error vector. For each model, we report misclassification rates and their standard error (SE) computed as the mean of misclassification rates and SE of the misclassification rates. As seen in Table I, the train error is 0 both for the tree and random forest, which is expected, because we built full trees, therefore making them overfit to the train data. Furthermore, the random forest misclassification rate is lower than the tree, which is expected due to the model being more robust, however both models achieved relatively low SE.

Model	Data split	Misclf. rate [%]	SE [%]
Full tree	train	0.00	0.00
Full tree	test	26.90	6.03
Random forest	train	0.00	0.00
Random forest	test	21.46	5.14

TABLE I
Different model performances on train and test data splits.

Figure 1 shows the misclassification rate versus the number of trees in the forest. Notice that performance does not improve after using more than 150 trees, as indicated by the misclassification rate and SE. This may be due to two factors: (1) our dataset is relatively small, limiting tree diversity, and (2) full trees are used, which might cause overfitting on the training data.

B. Permutation-based variable importance

We also implemented the permutation-based variable importance method, which uses out-of-bag (OOB) samples to measure feature importance. For each tree in the forest, we first record the baseline accuracy using the original OOB

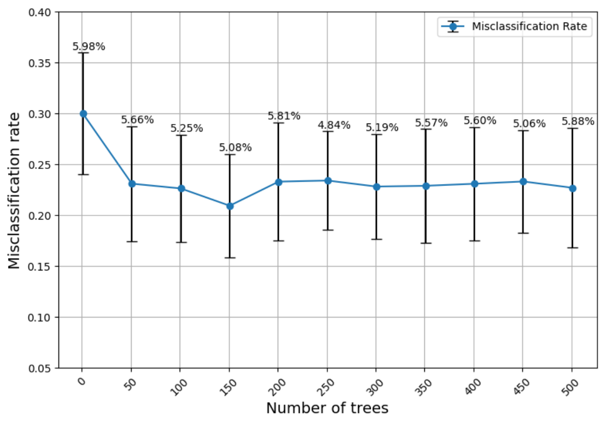

Fig. 1. Full tree random forest classification performance based on the amount of trees. Above each point, the SE is reported.

samples. Then, for each feature, we randomly permute its values and measure the change in accuracy. The drop in accuracy indicates the importance of that feature to the model. Finally, the importance scores are averaged over all trees and shown in Figure 2.

Additionally, we built 100 non-randomized trees, where each split considered all features instead of a subset. This ensures that the best features are selected most often while still allowing variability from bootstrapping. Features used at the root nodes are highlighted in orange. We notice a strong correlation between permutation-based importance scores and the features most frequently selected in these trees.

C. 3-variable permutation-based importance

The single-variable permutation-based importance was upgraded to consider all combinations of 3 variables, allowing us to account for correlation between them. To avoid checking all 10 million combinations per tree, we checked only combinations of the features that were actually used for splits in the tree. We tested both approaches on a random forest with 1000 trees and used the best 3 features to build a new full tree. For the single variable importances, those were features 274, 275 and 276, and for the 3 variable combinations 274, 275 and 361, so they only differed by a single feature.

The tree built on the first combination yielded misclassi-

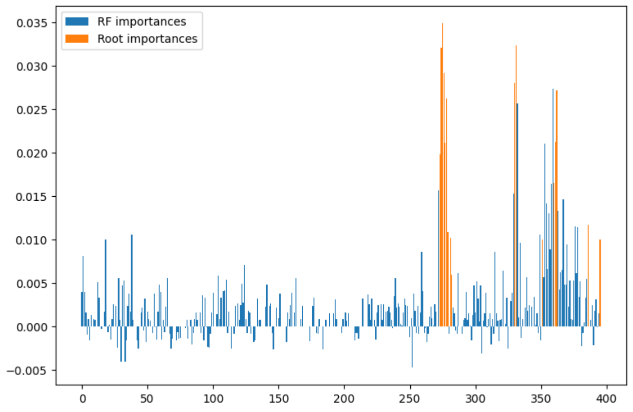

Fig. 2. Permutation-based variable importances and the root features of 100 non-random trees.

fication rate of 42.99% ± 4.51% and the one using the 3-combinations reached a significantly better result of 28.23% ± 4.24%. The performance of the combination is most likely better because some of the variables in the first case were strongly correlated, so they were important on their own but redundant when used together.

D. 3-variable importance on pre-built forest

As a part of the project we were also faced with a problem of acquiring the best combination of 3 features in the case where we cannot use any data to do so (pre-built model). The model being pre-built also means we can't remember any gini impurity reductions on splits, so the most obvious solution is to simply count how many times a feature has been used for splitting and take the combination that has the most occurrences. We can also upgrade this approach by accounting for the depth at which the feature was used for splitting - root splits have more importance than deeper splits, however that adds complexity in how to combine the information.

We implemented both approaches, using normalized feature occurrences and split depths on a random forest with 100 trees, built with the same random seed as the one in subsection II-C. The method based solely on split counts selected features 276, 330, and 379, while the depth-aware method returned 274, 279, and 362. We also tried putting a logarithmic scaling on the depth which returned another set of features. Then we built trees on all 3 sets and compared their performance. The method based only on feature occurrences outperformed the other two, though logarithmic scaling improved the depth-aware approach. Notably, all methods selected better features than the single-variable permutation-based feature selection.

III. Conclusion

In this project, we implemented decision trees and random forests while exploring different methods for feature selection and uncertainty quantification. Our experiments have shown that it is better to rely on combinations of features when analyzing feature importances, as this can mitigate some high correlation related issues.