Machine Learning - Homework 5 Report

Lukas Johannes Ruettgers (2023403372)

November 1, 2023

1 Objective

The experiment's objective is firstly to examine and evaluate the influence of hyperparameters on the performance of a Random Forest Classifier (RF) on a multi-feature dataset. Secondly, we compare various techniques to estimate the contribution of the dataset's features to the class separability. The dataset originates from https://www.kaggle.com/c/widsdatathon2020/data and contains medical information of patients at the Intensive Care Unit (ICU) of US-American hospitals. The patients x_i were classified into two groups depending on their survival s_i during their stay at the ICU.

The model is supposed to estimate the probability of survival ($s_i = 1$) or death ($s_i = 0$) respectively for any patient x_i represented by the feature schema of the given dataset.

2 Experiment setup

2.1 Model

While many complex models bear the capacities to learn sophisticated patterns in data, they often tend to overfit on the training dataset. Conversely, random forest classifiers follow the principles of ensemble learning and intend to combine multiple limited decision trees per democratic vote into a strong model. In the optimal case, the individual decision trees should be confined in their complexity so that they need to apply oversimplified decision rules to classify the data. These simple decision rules should be sufficiently diversified across the decision trees and not follow the same inductive biases. In this way, an equally weighted combination of their simplified predictions could rectify their respective fallacies.

The performance of this mechanism heavily relies on these assumptions. To pursue their satisfaction, the decision trees must be artificially restricted in their learning capacity. Owing to the shape and system behind decision trees, there are multiple parameters to limit their power. Before introducing these parameters, let us hence first recall their functioning. Starting at the root, each node classifies incoming instances[1] according to one feature into its child nodes. Finally, the leaf nodes of each tree comprise a group of elements which are considered to be equivalent with regard to the classification. To choose the feature at each node during the training process, the available features are compared using a metric on the class separability of the available training data. Common metrics are the Gini impurity, entropy or simply the logarithmic classification loss when discriminating by the feature. These metrics try to quantize the arbitrariness of this feature with regard to the classification. A feature that is unrelated with the classes would randomly partition the instances across its value domain and effectively contribute nothing to the information

[1] The term "instance" refers to an object of the observed population and should be understood analogously to the term "sample".

of the achieved subclasses with regard to their classification. Such a feature would be said to have a high impurity. The feature which yields the best gain in the used metric is then chosen to further split the group of instances.

However, we can pose more conditions on such a split to effectively take place. Two popular constraints are an upper bound on the tree depth and a lower bound on the number of instances required in a node to be further split. A stronger limitation is to demand a minimum number of instances in each leaf node. Features whose application would result in too small nodes are neglected if this restriction is enabled. These two constraints ensure that the decision tree does not become too dense and its classification not to detailed. To that end, we can also avoid splits that yield only negligible improvement by requiring a minimum gain in impurity for splits to be applied.

So far, these hyperparameters merely impede models to develop too complex predictors that may suffer from overfitting. As stated above, we desire to diversify the predictors of the individual decision trees. However, providing all of them the same training data and the same features to split upon would result in identical classifiers, because the training algorithm is deterministic. This is why we need to introduce randomness to the learning procedure. Firstly, each decision tree does not receive the entire training dataset, but only a bootstrapped[2] subset to force diversification among the models. Secondly, one may also consider only a sampled fraction of available features for splitting a node. This enlarges the diversity of features used along the trees' paths, which is in particular helpful if some features achieve so high scores that they are selected at the top layers of all trees. Lastly, the number of decision trees also improves the statistical properties of the combined prediction at the cost of computational power.

2.2 Preprocessing

The training dataset and the testing dataset comprises $n_{TrSamp} = 5000$ and $n_{TsSamp} = 1097$ samples respectively. Each sample is described by $n_{FT} = 108$ features subject to binary, numeric and integer domains. While the binary values were solely converted into integers, the values of numeric and integer features were all normalized to $[0, 1]$, since their values varied largely in size and units. This was achieved by linearly rescaling each feature value x to $\frac{x-x_{min}}{x_{max}-x_{min}}$, where x_{min} and x_{max} are the maximum and minimum values of the respective feature.

Over and above, we test how the regularization rate affects the final decision hyperplane by comparing the values $C = 1, 0.5, 0.1$.

2.3 Cross Validation

Besides training the model with the entire training dataset, the bias of the models was assessed via 10-fold cross validation (CV). The training data set was split into ten equally sized intervals and the 10-fold CV error was computed as the mean accuracy of the predictions of the models trained on their respective testing interval.

2.4 Implementation details

In this experiment, the Random Forest Classifier class of the `scikit-learn` package is used. Their implementation deviates in small aspects from the original description by Breiman. Instead of collecting the votes of the individual tree to classify the sample based on the majority of votes, this package first averages the class probabilities of each decision tree. Then, it bases the decision of the RF on the mean probability to achieve more granular predictions. This means that each decision tree does not vote for one class, but

[2] Bootstrapping refers to random sampling with replacement.

uses the fraction of class training samples in each leaf node as an estimation of the class probability for a sample that ends up in this leaf node.

2.5 Feature selection

The package also provides a method called *random forest feature importance* to assess the contribution of each feature to the prediction of the model. Specifically, the contribution is computed as the product of two factors. The first factor quantitatively estimates the fraction of instances to whose classification the feature contributes. The second factor assesses how the feature qualitatively improves the classification at each node by considering the loss in impurity. To sum up, at each node where the feature appears, the fraction of training samples that reach this node is multiplied by the impurity decrease that the feature achieves. Of course, both factors solely base on the training data and have no guaranteed relation to the true importance of the feature. Moreover, the impurity decrease favours features whose domains have a high cardinality, since the resulting child nodes are smaller and thus more distinctive than the child nodes a coarser split on average would achieve.

An evaluation algorithm that does not fall short of the second flaw is the *permutation importance* algorithm. This algorithm randomly permutes the values of a given feature and bases the importance of a feature on the performance decrease due to the random permutation.

To address the first shortcoming that importance measures should base solely on the training data performance, one may calculate the feature importance on a held-out validation set.

2.6 Observed variables

Firstly, we want to observe if the restrictions introduced in Section 2.1 effectively impede the decision trees from overfitting the training data. To that end, we choose a model that does not avail to these extensions as a baseline model and then alternate values for

- the maximum tree depth (DEP)
- the minimum impurity decrease for splitting a node (IMPSPLIT)
- the minimum number of samples in a node for splitting (SMPSPLIT)
- the minimum number of samples in each leaf (SMPLF)
- the number of sampled features available for splitting (MAXFT)
- the number of samples bootstrapped for each tree (MAX_SAMPLES).

Specifically, we first run models that only avail to random subsets of training data (MAX_SAMPLES) and features (MAXFT) and try out each combination of the values listed in 1. After that, we will combine IMPSPLIT and SMPSPLIT to restrict the splitting procedure. The influence of a limited tree depth and a minimum amount of samples per leaf are first assessed individually. Finally, the best values for all five hyperparameters are combined into one last model. To compare the performance with the baseline model, we consider the training, validation and testing accuracy, which we define as $\frac{TP+TN}{TP+FP+FN+TN}$.

Furthermore, we apply these measures on both the training and validation datasets to examine both the features that contribute the most to overfitting the training data and the features that aid the most in generalizing to unseen data.

If not stated differently, we use the default values proposed by the package documentation. In particular, our random forests consist out of 100 decision trees.

Hyperparameter Name	Values
DEP	5, 10, 20
IMPSPLIT	0.01, 0.1, 0.2
SMPSPLIT	5, 20, 50
SMPLF	5, 10, 20, 30, 40
MAXFT	$\log_2(n_{FT})$, $\sqrt{n_{FT}}$, $0.3 \cdot n_{FT}$
MAX_SAMPLES	$0.1 \cdot n_{TrSamp}, 0.3 \cdot n_{TrSamp}$.

Figure 1: Experiment values for the hyperparameters, where $n_{TrSamp} = 5000$ and $n_{FT} = 108$.

3 Results

3.1 Baseline model

Without any restrictions, the standard random forest perfectly overfits the training data and achieves an accuracy of 100%. On the other side, the RF baseline model with the gini coefficient classifies only 77.21% of the training dataset correctly and receives a cross validation error[3] of 19.02%. If employed with the entropy coefficient, this accuracy further decreases to 75.66%. Figure 2 contrasts this huge performance gap between the training and the testing data. Since the baseline model with the gini coefficient achieved the best performance, we keep this model as baseline model for the upcoming comparisons.

3.2 Random subsets of features and training samples

The introduction of randomness to the selection of available features for the splitting procedure and the limitation of the bootstrapped samples to $\frac{1}{10}$ or $\frac{3}{10}$ of the training data size does not improve the models performance. Indeed, the training accuracy remains 100% for all possible combinations of hyperparameter values in Figure 1. The best model with MAXFT=$0.3 \cdot n_{FT}$ and MAX_SAMPLES= $0.1 \cdot n_{TrSamp}$ achieves only a slightly better training accuracy of 77.94%. However, for the other values for MAXFT, MAX_SAMPLES= $0.3 \cdot n_{TrSamp}$ seems to yield better results.

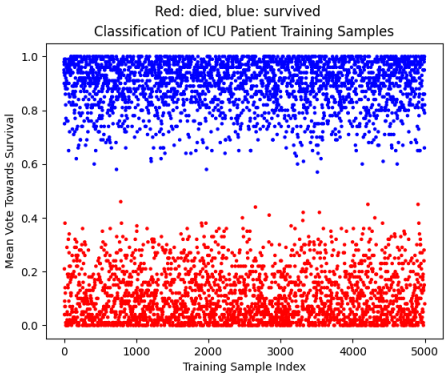
Because of the unconstrained splitting procedure, the model still splits nodes until there is only one sample left in each node. Even though the number of bootstrapped samples is smaller, the trees still become too granular and overfit on their subset. Considering that the random forest comprises 100 trees in total and that the lowest fraction of MAX_SAMPLES was just one tenth of the dataset, for each training sample there are enough trees that have been trained on this sample to perfectly classify it.

Figures have been left out in this subsection for the reason that the performance in training, validation and testing does not differ from the baseline model.

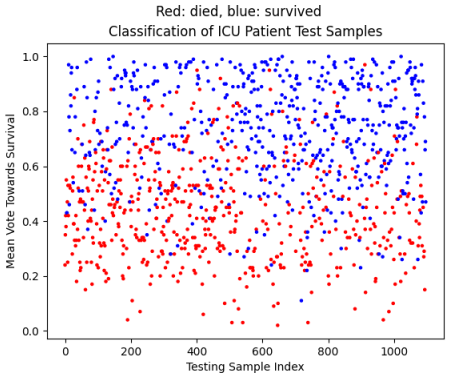
3.3 Restricted splitting

Solely constraining the splitting to a minimum decrease in inpurity or by a minimum number of samples in the node to be split does also not significantly raise the performance above the baseline. Although we observe that raising IMPSPLIT to 0.2 achieves an accuracy of 78.12% on the testing data, the training accuracy of 100% appears even if not less than 50 sampled are allowed in a node to be split. So far, there might be two possible reasons for this. The first reason could be that the overall overfitting is not primarily achieved on the last fine layers of each decision tree, but that the interplay of the rougher upper layers already suffice to perfectly memorize the data. The second reason could be

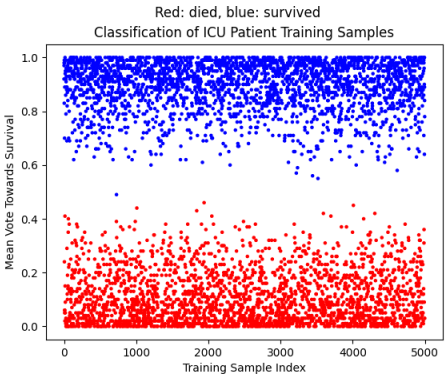
[3]Here, error refers to the opposite of accuracy. Precisely, error = $\frac{FP+FN}{TP+FP+FN+TN}$



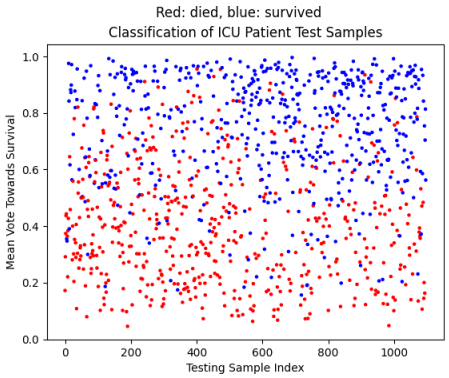
(a) RF with entropy criterion on testing data



(b) RF with entropy criterion on testing data



(c) RF with gini criterion on testing data



(d) RF with gini criterion on testing data

Figure 2: The performance of the baseline model with the entropy and the gini criterion on the training (left) and testing (right) datasets. Red dots indicate patients that died, blue dots indicate patients that survived.

that some nodes use features with a high cardinality to overfit the data. This way, nodes could contain far more samples than SMPSPLIT= 50, but achieve a fine classification by availing to such a feature with high cardinality. The resulting child nodes would all contain quite few samples and leads to detailed decision making which is not generalizable. The results in the next section provide further arguments why the second reason could be true.

3.4 Leaf size restriction

Surprisingly, setting the minimum number of samples per leaf to 20 notably shrinks the training accuracy to 90.92% and raises the test accuracy to 79.03%. The other values close to 20 also achieve a similar improvement. In contrast to setting a lower bound for the inner nodes as a prerequisite to be further split, directing the restriction to the leafs of the trees seems to be particularly effective. Recalling that the training accuracy remained at 100% when increasing SMPSPLIT in the last section, this further supports the second hypothesis in the last subsection. It stated that the cardinality of some feature values used for splitting is so large such that for many nodes whose splitting was not impeded by SMPSPLIT= 50, the splitting was now restricted by SMPLF. This is because even if nodes with many samples they would be further split by such a feature, then the numerous child nodes would contain only few samples respectively. SMPLF therefore effectively contributes to hindering decision trees to overfit the training data on granular decisions based on features with high cardinality.

3.5 Tree depth

A quite similar improvement is observed when limiting the depth of each decision tree. Although the performance varies only slightly across depths $5, 10$ and 20, the model with DEP=5 decreases the training accuracy to 90.56% and achieves a test accuracy of 78.85% which is almost as good as the one for the best SMPSPLIT model.

The observation that the training accuracy does remain around 90% for increasing tree depths directly contradict the first hypothesis in Subsection 3.3 that claimed that that the interplay of the rougher upper layers already suffice to perfectly memorize the data. We therefore discard this hypothesis and conclude that the overfitting must occur on the deeper layers of each decision tree respectively.

3.6 Combining the best values

Because of their significant and equal contribution to reducing overfitting, we choose SMPLF=20 and DEP=5 for our final model. As MAXFT=0.3 dominated the other choices, we will set this value accordingly. Because the results did not clearly indicate a relation between MAX_SAMPLES and IMPSPLIT and an improvement in the model's performance, we will both try their best values and also left them out. Since SMPLF can be regarded as a harder constraint than SMPSPLIT, the latter is completely left out. The combination of limiting the tree depth and the leaf node size does not further decrease the training accuracy, which keeps at 90%. This might indicate that the features individually sufficiently regularize the model. However, the testing accuracy remains at 78.85% and does not improve in comparison to the individual models with DEP and SMPLF. Nonetheless, we observe that setting IMPSPLIT=0.2 indeed improves the model slightly.

3.7 Feature selection

As stated in the documentation, the Random Forest Feature Importance measure strongly overweights the high-cardinality feature "apache_4a_icu_death_prob" and assigns this feature the highest importance of 0.39 for the baseline model. The next feature has only an importance of around 0.02. In the improved model, this importance measure even increases up to 0.56.

This confirms the warning in the documentation about