Feature importance measures for random forests: the problem of Mean Decrease Impurity, solutions and alternatives

Gaétan de Castellane

September 12, 2025

Abstract

In the field of tabular machine learning, tree-based models such as random forest and gradient boosted trees are widely popular for their ease of use, excellent predictive capabilities and interpretability. However, this interpretability is often based on the Mean Decrease Impurity (MDI), a measure of feature importance proposed by Breiman in his original paper, that has been proven to give misleading results when one wishes to accurately assess the extent of the relationship between a feature and the target of a machine learning task. This report presents our effort to study the problem of MDI for random forest, the proposed solutions and how they compare to the broader field of feature importance measurement. It also unifies tree feature importance measures and proposes a new method that has additional desirable properties. Finally, it assesses the performance of all presented methods.

1 Problem statement

The importance of interpretability

In machine learning, understanding the decision of a trained model is often critical, especially for so called *black box models* where the learned decision is not understandable at first sight. This is particularly true in fields where mistakes are costly (e.g. patient care) or in fields where machine learning is performed to understand the mechanisms that are studied (e.g. biology research). The need for interpretability (Molnar 2020) in Artificial Intelligence (AI) models is further motivated by the recent European Union AI act[1] that imposes transparency on deployed high-risk AI systems.

Feature importance is one way to increase the interpretability of a model and its resulting predictions. It is generally defined as a value attributed to each feature reflecting how much the output of the model depends on it. Intuitively, we want to assign higher importance to features that have the highest impact on the model's prediction. We give a formal definition of feature importance later, after discussing why its definition may vary depending on the objective of the practitioner.

Tree-based models are interpretable

Tree-based models such as random forest (Breiman 2001) strike a good balance in the trade-off between model interpretability and performance (Molnar 2020). Indeed, a single decision tree is highly interpretable –a prediction is made because the individual is in a specific region of the input space– and ensemble of trees retain some of that interpretability. Mean Decrease Impurity (MDI) is a measure proposed by Breiman (2001), that aggregates across the trees of the forest how much each feature has allowed to separate the input space into homogeneous subspaces (w.r.t. the target value).

MDI can be misleading

However, the MDI suffers from the overfitting and selection bias of decision trees. It leads it to consistently assign non-zero importance to features that are independent of the target. The effect is worse for features takes many unique values, such as continuous features or high-cardinality categorical

[1] https://artificialintelligenceact.eu/high-level-summary

features, as seen in Fig. 2. Consequently, it is impossible to perform feature selection with MDI and it may mislead practitioners into considering a feature as relevant in their problem despite it being, for instance, random noise, or into over-estimating the importance a feature because of its high cardinality.

MDI is widely used in scikit-learn

Scikit-learn (Pedregosa et al. 2011) is an open-source Python library that provides efficient implementations of state-of-the-art machine learning algorithms. During the training of tree-based models, scikit-learn precomputes MDI to measure the importance of features and presents it under the public attribute `feature_importances_`. Many users of the library use this attribute to inspect their tree-based models because it is easy and cheap to access, while unaware of the potentially misleading results. This attribute is widely used in practice making it difficult to deprecate. Therefore, the developers of the library are seeking for a more reliable replacement, that does not suffer from the aforementioned biases and is reasonably cheap to compute.

Context of the internship

This internship was conducted within the open source team of probabl, a company created around scikit-learn, to foster and fund the growth of the project. Under the supervision of Olivier Grisel, Antoine Baker and Guillaume Lemaitre, I conducted the following research work to find a solution to the issue of scikit-learn's tree feature importance. I also built optimized implementations of the solutions proposed in the literature for scikit-learn.

Contents

2 Background and notations

2.1 Setting

In what follows, we consider the standard supervised learning framework by denoting $X = (X_1, ..., X_p) \in \mathcal{X} \subset \mathbb{R}^p$ a multivariate random variable representing the features and $Y \in \mathcal{Y} \subset \mathbb{R}$ the target. We also define $X_{-j} = (X_1, ..., X_{j-1}, X_{j+1}, ..., X_p)$ the set of features deprived of feature j and more generally for S a subset of $\{1, ..., p\}$, we define $X_S = (X_i)_{i \in S}$ and $-S$ the set $\{1, ..., p\}$ deprived of S.

Letting P the joint distribution of X and Y, such that $(X, Y) \sim P$, we consider the dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with n i.i.d. samples from P. We also let $m : \mathcal{X} \to \mathcal{Y}$ such that $m(x) = \mathbb{E}[Y|X = x]$ the Bayes model, i.e. the best approximation of the target knowing the features.

2.2 Feature importance

Generally, the importance of a feature is defined as the amount of predictive power it provides to a model. However, there are two considerations that we need to take into account before measuring feature importance.

Firstly, the dependency of a model on a feature can be measured marginally or conditionally on the other features. In the first case, any feature that brings information is deemed important, regardless of whether this information is already present in an other feature. For example, if we want to predict the weight of a person using their height, an additional binary feature indicating whether the person is above the average height or not is important, despite providing no additional information about the person. This is why, in most cases, we are more interested in conditional feature importance that rewards feature with unique information. In the same example, considering the age of the person may bring new information about their weight, even when knowing their height. Here, we focus on conditional importance as it allows to select feature by removing those with redundant information. All mentions of importance in the future implicitly refer to the conditional version.

Secondly, when measuring feature importance, we must distinguish between explaining the model versus explaining the mechanisms that underlie the observed data. Since in practice we never know the true data generating process, the model is used as a proxy for the relationship between the features and the target, which potentially leads to misinterpretation of the feature importance measure. Indeed, models that perform well but not optimally may use uninformative features in their decision or ignore informative ones, which can give a false interpretation of the link between a target and its features. The issue is worse for models that underperform by either underfitting or overfitting, in which case feature importance should not be used to explain the observed process. It should be kept in mind that all the methods presented here aim to explain the statistical association between the variables used as target and input feature, but actually explain the link between a feature and the output of the model.

Each feature importance method has its own approach to measuring the predictive power contribution of a feature. For instance, MDI measures how much a feature contributed to the reduction in prediction error on the training set through impurity decrease in tree splits. Permutation importance (Breiman 2001) quantifies how much performance is lost when the information in feature X_j is corrupted by randomly permuting its values. Shapley Additive Global importancE (SAGE) (Covert, Lundberg, and Lee 2020) values measure the average reduction in prediction error when feature X_j is added to all possible subsets of other features, providing a game-theoretic allocation of the model's total performance improvement.

For practitioners interested in feature selection, an additional desirable property is that the importance measure should satisfy $FI(j) = 0$ if and only if feature X_j is not strongly relevant for predicting the target. It creates a direct correspondence between zero importance and the absence of predictive value, enabling automatic feature filtering based on importance thresholds. We consider here the following definition of strong relevance, from Kohavi and John (1997):

Definition 2.1. A feature $j \in \{1, ..., p\}$ is strongly relevant if and only if $X_j \not\perp\!\!\!\perp Y | X_{-j}$.

Strong relevance means that the feature cannot be removed without loss of prediction performance. Finding the set of all strongly relevant features is the objective of feature selection.

One differentiates feature importance measures by the theoretical and practical properties they satisfy. Key differentiating properties include: model agnosticism (whether the method can be applied to any trained model), additive decomposition of performance improvement (whether individual feature contributions sum to the total model performance), satisfaction of the Shapley axioms from game theory, computational cost, robustness to feature correlations, reliance on held-out data not used for training, and bias and variance induced by the finiteness of the data sample used to estimate it. The choice of feature importance method is inherently subjective, and each practitioner should select the approach that best aligns with their specific requirements regarding these properties and the desired interpretation of feature contributions.

Remark. Here we study **global** importance, that explains the impact of features on the whole model. **Local** importance, on the other hand, focuses on explaining the impact of features on a single prediction, and is not studied here.

2.3 Random forest

In this section, we briefly define what random forest models are and how they are trained. Let us first define decision trees, impurity functions, and bagging.

Definition 2.2 (Decision Tree)**.** In machine learning, a decision tree t is a non-parametric model that is composed of a set of nodes m, each associated with a value v_m and a hyper-rectangle $R_m \subset \mathcal{X}$ that is a sub-rectangle of the space defined by its ancestor node. The decision function f_t of tree t is be expressed as

$$f_t(x) = \sum_{m \in \text{leaves}(t)} v_m \mathbb{1}_{x \in R_m}, \tag{1}$$

where the leaves of t are the nodes in t that have no children.

To explain how decision trees are trained, let us first introduce the notion of impurity.

Definition 2.3 (Impurity function)**.** Given the loss function l that has been chosen to train a tree t, the impurity of node m is is the average loss between the target value of the samples that go through node m (i.e. that are in R_m) and the value of the node. This is be formally defined as

$$H(m) = \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} l(y_i, v_m) \tag{2}$$

where we let $n_m = \sum_{i=1}^{n} \mathbb{1}_{x_i \in R_m}$ the number of points that go through node m. We refer to H as an impurity function.

The CART method, proposed by Breiman et al. (1984), builds binary decision trees that recursively partition the input space $\mathcal{X}$ by splitting the subspace R_m of node m into two subspaces $R_{l_m} = \{x \in R_m : x_{j_m} \leq s_m\}$ and $R_{r_m} = \{x \in R_m : x_{j_m} \geq s_m\}$. The feature j_m and threshold s_m used for splitting m are chosen so that the resulting partition minimizes $\frac{n_{l_m}}{n} H(l_m) + \frac{n_{r_m}}{n} H(r_m)$, i.e. the weighted impurities of the resulting nodes l_m and r_m. The split feature j_m is chosen among all or a subset of the features $\{1, ..., p\}$.

This recursive partitioning is continued until a stopping criterion is met for each node. Some example of stopping criteria are: the node is pure –$H(m) = 0$, the node is at a maximum depth or there is less than the maximum amount of samples in the node. The stopping criterion is usually chosen by the user as a hyper-parameter and allows to modulate between bias and variance: deep trees overfit more than shallow trees that might underfit.

Once a parent node is split, the value of a resulting child node m is defined as the average response of the individuals in m. In regression, it is

$$v_m = \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_i. \tag{3}$$

In K-class classification, it is the vector

$$v_m = [p_{m,k}]_{k=1}^K = \left[\frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_{i,k} \right]_{k=1}^K \tag{4}$$

as we consider that the target $y_i = [y_{i,k}]_{k=1}^K \in \{0,1\}^K$ is a one-hot encoder of the classes.

We illustrate the concepts we just introduced in Fig. 1. In this figure, we trained a decision tree of depth 2 on the `titanic` (Harrell and Cason 2017) dataset, where the task is to predict the survival of the passengers of the titanic, based on their age, sex, passenger class, etc. We see that `sex` is the first features that is used to split, with the level 0.5. Since we used an ordinal encoder on the feature, this separates samples with `sex="female"` in the left child node and the samples with `sex="male"` in the right child node. The root node has a Gini impurity value of 0.472 while the resulting children have an impurity of 0.397 and 0.309 respectively. These children node are further split on the feature `pclass` and `age`. Out of the four leaves, the two purest are the left and right most leaves, with gini impurities of 0.127 and 0.282, respectively. They correspond to female passengers boarded on passenger class 1 where 233 out of 250 ($= 0.932$) survived and to male passengers with age greater than 9 where 664 out of 800 ($= 0.83$) did not survive.

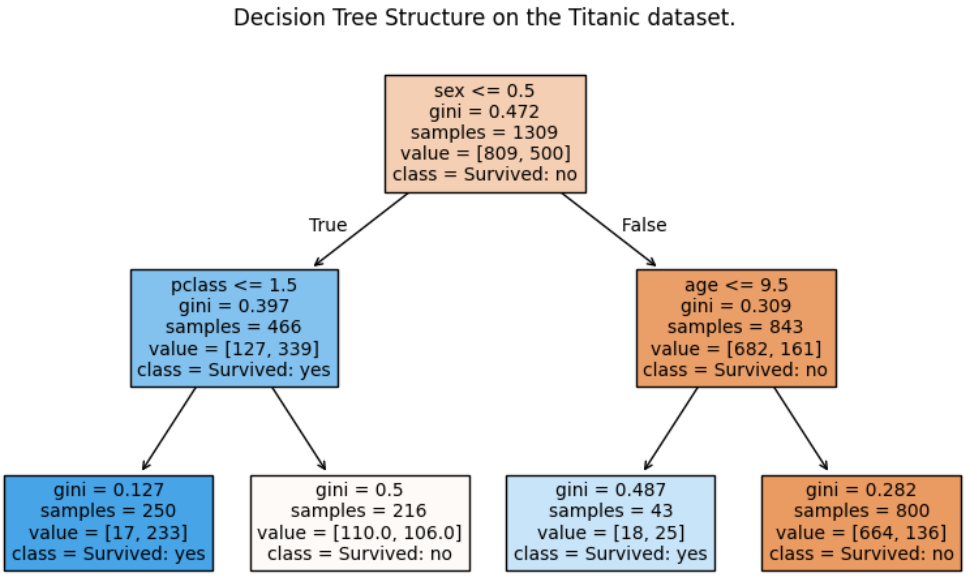

Figure 1: Decision tree of depth 2 to predict the survival of passengers of the titanic (Harrell and Cason 2017) based on passenger information.

Let us now define what *bagging* means, as introduced by Breiman (1996):

Definition 2.4 (Bagging)**.** Bootstrap aggregating, also referred to as bagging, is an ensemble learning method where T training sets are constructed by sub-sampling with replacement $n_{bootstrap}$ data points from a full training set $\mathcal{D}$ (bootstrap). Then, a separate machine learning model is trained on each sampled dataset $\mathcal{D}_b$, $b \in \{1, ..., T\}$. Their decision functions are then aggregated to produce a final decision function (aggregation).

Remark.

- Usually we take $n_{bootstrap}$ equal to n, the number of points in the full dataset $\mathcal{D}$.
- The out-of-bag samples of a given tree are likely to serve as in-bag samples for other trees in the forest.

- In-bag samples most often contain repeated observations because bootstrapping samples with replacement.

For each one of the trained models, this procedure leaves aside samples that are not be seen by the model. We refer to these samples as the *out-of-bag* samples, which is be defined as follows:

Definition 2.5 (Out-of-bag samples)**.** The out-of-bag (oob) samples of dataset $\mathcal{D}_b$ are the samples of the full dataset $\mathcal{D}$ that have not been used to train the b-th model. The set of oob samples $\mathcal{D}'_b$ is defined as:

$$\mathcal{D}'_b = \{(x_i, y_i) \in \mathcal{D} : (x_i, y_i) \notin \mathcal{D}_b\} \tag{5}$$

Remark.

- The number of oob samples is not constant across the training sets since we are sampling with replacement. When $n_{bootstrap} = n$, there is an average of $n' = \frac{1}{e} \approx 0.368$ of the samples that are left out as oob samples.
- These oob samples can serve as testing samples for individual trees to estimate generalization error.

We now define a random forest, as originally proposed by Breiman (2001).

Definition 2.6 (random forest)**.** A random forest F is an ensemble of T bagged decision trees, where each split is chosen only on a random subset of features of size $p' \leq p$. Its decision function is the average decision of the trees:

$$f(x) = \frac{1}{T} \sum_{t \in F} f_t(x). \tag{6}$$

2.4 MDI

Breiman et al. (1984) also proposed to measure the importance of features with the Mean Decrease in Impurity (MDI). For a feature $j \in \{1, ..., p\}$, it is be defined as follows:

Definition 2.7 (Mean Decrease in Impurity)**.** Let H the impurity function used during training, inter(t) the internal (i.e. non leaf) nodes of tree t, l_m and r_m the left and right children of node m and $\omega_m = \frac{n_m}{n}$ the proportion of training samples that go through node m. ω_{l_m} and ω_{r_m} are defined in the same manner. Then the MDI of feature $j \in \{1, ..., p\}$ is defined as

$$\text{MDI}(j) = \frac{1}{T} \sum_{t \in F} \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \left[\omega_m H(m) - \omega_{l_m} H(l_m) - \omega_{r_m} H(r_m) \right]. \tag{7}$$

This function measures how much each feature was instrumental in creating homogeneous subspaces of the feature space $\mathcal{X}$, by averaging the impurity decrease of nodes that were created using them.

3 Related work

In this section, we present the work related to feature importance measurement for tree-based models through the MDI. We also introduce other popular feature importance measures that do not depend on the choice of model.

3.1 Identifying the bias of MDI

There is a large amount of literature describing the issues with MDI for feature importance. We have highlighted three such issues that we present here.

Firstly, the MDI exhibits a systematic positive bias, assigning non-zero importance even to irrelevant features. While a rigorous proof that $\mathbb{E}[\text{MDI}_j] > 0$ for all features is difficult due to pathological cases where features can be perfectly orthogonal to the target, the general phenomenon is well-established both empirically and theoretically. The literature recognizes that MDI can be decomposed into a heterogeneity reduction component (representing true importance) and a positive bias component (Sandri and Zuccolotto 2008). This positive bias typically arises from finite sample effects: even irrelevant features exhibit spurious correlations with the target in any finite dataset, and the random forest's design, through bootstrap sampling and random feature subsampling, ensures these spurious correlations are systematically exploited across trees. This systematic bias makes MDI unsuitable for feature selection, as it can mislead practitioners into considering irrelevant features as relevant.

A second issue is the one reported by Strobl et al. (2008) on the preference for high cardinality features. Indeed, they show that the MDI consistently favors features that are either continuous or categorical with a high number of categories. They argue that this issue stems from the selection bias of trees built with the CART method: features with many unique values offer more threshold values to select from, which allows more potential impurity improvement than lower cardinality features. This effect can compound with the previously mentioned one making the MDI even more misleading for high cardinality noise features.

Thirdly, the MDI suffers from amplification bias due to overfitting in individual decision trees. Li et al. (2019) demonstrate in their Theorem 1 that in finite sample regimes, which characterize most practical applications, the sum of MDI values for irrelevant features is lower bounded by a value inversely proportional to the maximum leaf size. Since smaller maximum leaf sizes correspond to deeper trees, this bound increases with tree depth, meaning deeper trees systematically assign more importance to irrelevant features. This overfitting bias compounds the positive bias discussed above: while finite sample effects create spurious correlations that give irrelevant features some importance, deeper trees amplify these spurious signals by fitting more closely to the training data noise. Since random forest eliminates overfitting issues through aggregation, trees are typically grown very deep by default, which exacerbates the assignment of substantial importance to irrelevant features in most practical implementations.

These three issues –systematic positive attribution, cardinality preference, and overfitting amplification, interact to make MDI particularly unreliable for distinguishing relevant from irrelevant features in finite sample regime. It should be noted that these issues vanish in the large samples limit: the probability of selecting an irrelevant feature to split, because of spurious correlations, goes to zero as the training set increases in size.

Figure 2 shows these three effects interacting together: all features receive importance, even the random features `random_num` and `random_cat`. On the plot on the right, where we limited the depth of the trees with `min_samples_leaf=20`, the impact of overfitting is lessened but there is still the positivity bias. The second point is illustrated by the fact that in both cases, the feature `random_num`, which has a unique value for each point, receives higher importance than the feature `random_cat` which has three unique values.

3.2 Debiasing the MDI

We now cover the solutions that have been proposed in literature to fix the aforementioned problems.

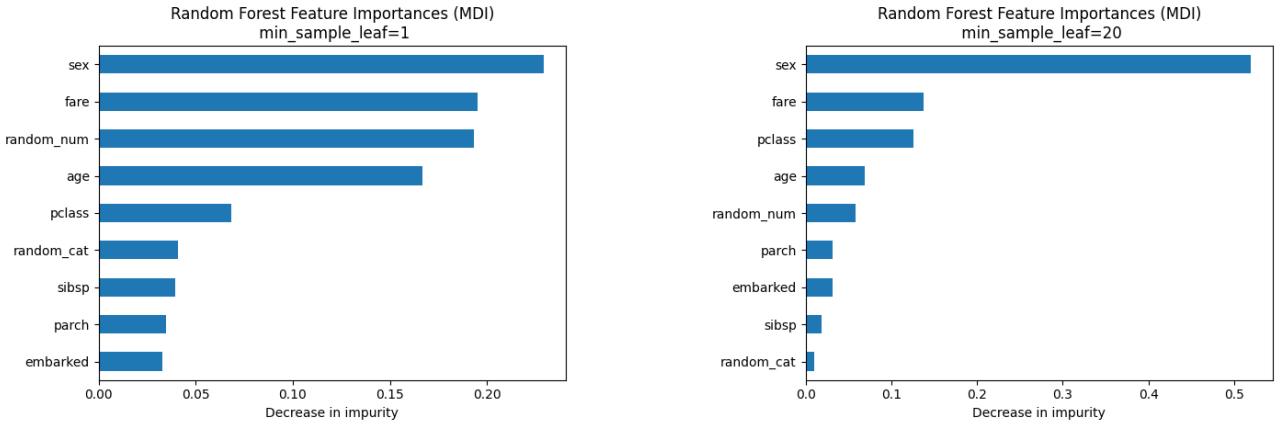

Figure 2: The bias of MDI on the titanic (Harrell and Cason 2017) dataset with two random feature of high (`random_num`) and low (`random_cat`) cardinality for forests of fully developed (left) and depth-limited (right) trees. For deep trees, which are the default in `scikit-learn`, `random_num` appears more important than actual predictive features such as `age` and `pclass`.

Stepping away from CART

Strobl et al. (2008) argue in favor of changing the tree building process to avoid the selection bias of CART-built trees and therefore eliminate the cardinality bias of MDI. Their proposed alternative is conditional inference trees (Hothorn, Hornik, and Zeileis 2006), which use a statistical test of independence between the features and the target as a stopping criterion. It also solves the overfitting bias by limiting the depth of the trees without the need to fix an hyper-parameter or to prune the trained trees.

This procedure, however, is much more computationally intensive compared to CART built trees. In the simulations of Strobl et al. (2008), a `cforest` (random forest with conditional inference trees, available in the R package `partykit` (Hothorn and Zeileis 2015) took between 25 and 35 times longer to train than a classical random forest. This high cost increase, coupled with the need to redesign the tree building process, made us consider other alternatives to fix the issue of MDI in `scikit-learn`.

Correcting the MDI with oob samples

An other solution, proposed by Loecher (2022), Z. Zhou and Hooker (2021) and Li et al. (2019), is to modify the formula for MDI by using the oob samples that are available for each tree of the forest. While the formulas of these three papers have some practical differences, the common idea is the following: change the impurity function so that it incorporates information on the oob samples and average the decrease in impurity over the nodes that split on the feature, like the MDI. The hope is that by leveraging information on the oob samples, which are essentially validation samples, the MDI generalizes outside the training set, and negates the impact of overfitting and selection bias.

To illustrate this, we look at the *UFI* (Unbiased Feature Importance) method proposed by Z. Zhou and Hooker (2021) where they suggest the following modified impurity function H_{UFI} in the case of binary classification with the Gini split criterion:

$$H_{UFI}(m) = 1 - p_{m,1}p'_{m,1} - p_{m,2}p'_{m,2} \tag{8}$$

where $p_{m,1}$ is the proportion of in-bag samples of class 1 that go through node m, while $p'_{m,1}$ is the same quantity for oob samples. $p_{m,2}$ and $p'_{m,2}$ are the same quantities for the samples that belong to class 2.

This expression stems from a modification of the Gini impurity criterion, which is the usual impurity function used to build classification trees:

$$H(m) = 1 - p^2_{m,1} - p^2_{m,2} \tag{9}$$

Remark. This formulation may seem far from the formalism we proposed in Eq. (2) for impurity functions but we show in Sect. 4 that it is the impurity function obtained when we use the Brier loss.

Using this modified impurity function allowed Z. Zhou and Hooker (2021) to consistently find the only informative low-cardinality feature in a context of high cardinality noise features. In this simulated example, their performance match or improve those of the MDI of `cforest`, in terms of average rank of the informative feature, and largely beat the regular MDI that struggles in this setting.

While their notations and formulations are very different, we show in Sect. 4 that the formulas proposed by Li et al. (2019) in their *MDI-oob* method are essentially the same as those of Z. Zhou and Hooker (2021) in *UFI*, up to a different weighting of the impurities in the decrease. We highlight that their measures converge to the same value (the MDI) in infinite samples regime which explains why Li et al. (2019) achieve performance on par with UFI when detecting noisy features in a controlled setting.

Finally, Loecher (2022) presents the idea of correcting the impurity with oob samples in a more general framework with no clear conclusion on what combination of in-bag and out-of-bag information is most desirable.

3.3 Permutation importance

The method to evaluate the importance of features that is most widely used is arguably permutation importance. Since the objective of feature importance is to quantify the impact of each feature on the output, a straightforward idea is to measure the degradation in prediction performance when we perturb the features. When the perturbation is a random permutation of the values in one column of the dataset, we call the procedure permutation importance. The idea was originally proposed by Breiman (2001) and the corresponding algorithm is be outlined as follows:

Algorithm 1 Permutation Importance

Require: Fitted predictive model f, validation dataset $\mathcal{D}$, number of repetitions K, scoring function Score

Ensure: Permutation importance scores $PI(j)$ for all features j

Compute reference score $s_0 \leftarrow \text{Score}(f, \mathcal{D})$ ▷ e.g., Brier score or R^2
for each feature j in $\mathcal{D}$ **do**
  for $k = 1$ to K **do**
    $\tilde{\mathcal{D}}_k^{(j)} \leftarrow \text{RandomlyShuffle}(\mathcal{D}, \text{column } j)$ ▷ Corrupt feature j
    $s_{k,j} \leftarrow \text{Score}(f, \tilde{\mathcal{D}}_k^{(j)})$ ▷ Score on corrupted data
  end for
  $PI(j) \leftarrow s_0 - \frac{1}{K} \sum_{k=1}^{K} s_{k,j}$ ▷ Permutation importance
end for
  return $\{PI(j) : j \in \text{features}\}$

Remark. Permutation importance values are be estimated either on the training set or an held out test set. The latter is recommended to assess the ability of the model to use the features to make predictions that generalize well to previously unseen cases.

This method attributes importance to features whose absence degrade performance. It is intuitively simpler to just evaluate the performance after removing feature j entirely (procedure known as *Leave-One-Covariate-Out*, LOCO) but that involves retraining the model many times, which comes with high computational costs and makes the procedure dependent on the randomness of the training procedure.

While permutation importance has been extensively used to measure the impact of features in a trained predictive model, the method was also criticized for cases where it is be misleading.

Hooker, Mentch, and S. Zhou (2021) explain that permutation importance gives misleading results when highly correlated features are present in the dataset. Indeed, when two features have a strong correlation, permuting one of the two features breaks the dependence and thus the original correlation. This leads to making predictions on a low density or empty region of the feature space, like on a two

meter tall person that weights forty kilos. The prediction then depends on the extrapolation behavior of the model rather than the link between the feature and the target.

Correlated features are also assigned a lower importance than they should if they are are informative. Indeed, when permuting one of two correlated and informative features, the model should not loose too much predictive power as it can use the other feature as a surrogate for the permuted one.

To address this issue, Hooker, Mentch, and S. Zhou (2021) and Chamma, Engemann, and Thirion (2023) proposed to permute the feature X_j conditionally on the other feature X_{-j}. It avoids breaking the inter-dependencies between features. According to the authors, this conditional permutation importance leads to statistically valid feature importance but it adds an additional computational burden to the already heavy permutation importance algorithm. In the "$n \ll p$" case where there are many more features than samples, which is when we are interested in feature importance and selection –e.g. when studying gene expression, permutation importance becomes very costly. Even more so for models that take long to predict such as deep neural networks or large forests of deep trees.

3.4 Shapley value importance

An other framework that has seen a lot of usage to measure the importance of features is additive explanations based on Shapley values from game theory, introduced by Shapley et al. (1953). Covert, Lundberg, and Lee (2020) propose the Shapley Additive Global ImportancE (SAGE) framework which allows to additively assign importance to features without making assumptions on the underlying model.

While we do not go into too much details about the theory of Shapley values here, we define SAGE values and the four axioms they satisfy. For a subset $S \subset \{1, ..., p\}$ of features, let us define $V(S) = \mathbb{E}[l(y, \mathbb{E}[f(X)])] - \mathbb{E}[l(y, \mathbb{E}[f(X)|X_S])]$ the value function of a learned model of decision function f. It represents the loss improvement, with respect to the constant model, of the model restricted to the features in $S \subset \{1, ..., p\}$. Then, SAGE values are defined as follows:

Definition 3.1 (SAGE values)**.** For a trained model of value function V, the SAGE value of feature X_j is defined as:

$$\text{SAGE}(j) = \frac{1}{p} \sum_{\substack{S \subseteq \{1,...,p\} \\ j \notin S}} \binom{p-1}{|S|}^{-1} (V(S \cup \{j\}) - V(S)). \tag{10}$$

The above expression uniquely satisfies the following four axioms:

- **Efficiency**: The total importance is fully distributed among all features: $\sum_j \text{SAGE}(j) = V(\{1, ..., p\})$.
- **Symmetry**: If two features j and k contribute equally to every subset, i.e., $V(S \cup \{j\}) = V(S \cup \{k\})$ for all S, then $\text{SAGE}(j) = \text{SAGE}(k)$.
- **Dummy**: If a feature j does not contribute to any subset, i.e., $V(S \cup \{j\}) = V(S)$ for all S, then $\text{SAGE}(j) = 0$.
- **Linearity**: If two value functions V and V' yield values $\text{SAGE}(j)$ and $\text{SAGE}'(j)$ respectively, then the value of $V + V'$ is $SAGE(j) + SAGE'(j)$.

The *Efficiency* axiom is particularly desirable for interpretability purposes, as it ensures that the sum of all feature importance scores exactly equals the total model performance improvement over the baseline. This provides a clear accounting of how the model's predictive power is distributed across features, making SAGE values easier to interpret and compare across different models or datasets.

The *Dummy* axiom, however, imposes very constraining requirements for feature selection applications. A feature receives zero importance under SAGE only if it is independent of the target variable **and** independent of all other features simultaneously. It means that even irrelevant features can receive non-zero SAGE values if they are correlated with relevant features, making SAGE unsuitable for straightforward feature selection tasks.

The main practical limitation of SAGE values is their computational complexity, which scales exponentially with the number of features. Computing SAGE exactly requires evaluating 2^p different feature subsets, making it prohibitively expensive for high-dimensional problems. Additionally, each evaluation requires conditional sampling to estimate $\mathbb{E}[f(X)|X_S]$, which introduces additional computational burden and potential approximation errors, especially when features are highly correlated.

Several recent works have proposed modifications to address SAGE's limitations. Janzing, Minorics, and Blöbaum (2020) argue that marginal SAGE values, where the sampling of dropped features is marginal rather than conditional, may be more appropriate for many applications. Similarly, Reyero-Lobo, Neuvial, and Thirion (2025) propose a minimal axiom that is satisfied by marginal SAGE but not its classical conditional version.

An important theoretical connection exists between SAGE and MDI in specific settings. Sutera et al. (2021) demonstrate that in *categorical setting* under asymptotic conditions, MDI and SAGE values converge to the same quantities. It provides theoretical justification for MDI's widespread use, though the assumptions required are quite restrictive in practice. Indeed, the *categorical setting* requires all features and the target to be categorical, and it requires the tree building process to create as many children nodes as there are categories for the feature that was used to split. It limits us to binary features as the majority of the decision tree implementations produce binary trees. We illustrate these theoretical connections and their practical limitations in our experimental Sect. 5.

4 Unifying tree feature importance measures

In this section, we unify the *UFI* (Z. Zhou and Hooker 2021) and *MDI-oob* (Li et al. 2019) methods under common notations and show that they are very closely related. We also suggest a new method that enjoys the additive property of SAGE values while converging to the other two in infinite sample regime.

We start by rewriting the MDI of a tree as an additive decomposition of the loss on the training sample. It allows us to propose a new method that extends the MDI to the oob samples which we call the *oob-score*. Finally we compare it to UFI and MDI-oob.

4.1 Rewriting MDI

Consider a learned decision tree t, of decision function f_t, trained on a dataset $\mathcal{D}$ of size n, with n_{oob} samples in the oob dataset $\mathcal{D}'$. For a node m, let v_m its value and let j_m denote the feature that was used to split the associated region R_m in two children nodes l_m and r_m. Following Saabas (2017), for any $x \in \mathcal{X}$, we decompose the prediction $f_t(x)$ in a sum of "contributions" for each feature:

$$f_t(x) = v_0 + \sum_{j=1}^{p} f_{t,j}(x) \tag{11}$$

where v_0 is the value of the root node and the contribution $f_j(x)$ of feature X_j to the prediction $f(x)$ is

$$f_{t,j}(x) = \sum_{\substack{m \in \text{inter(t)} \\ j_m = j}} \mathbb{1}_{R_{l_m}}(x) v_{l_m} + \mathbb{1}_{R_{r_m}}(x) v_{r_m} - \mathbb{1}_{R_m}(x) v_m. \tag{12}$$

The proof is pretty straightforward:

$$\begin{aligned}
v_0 + \sum_{j=1}^{p} f_{t,j}(x) &= v_0 + \sum_{j=1}^{p} \sum_{\substack{m \in \text{inter(t)} \\ j_m = j}} \mathbb{1}_{R_{l_m}}(x) v_{l_m} + \mathbb{1}_{R_{r_m}}(x) v_{r_m} - \mathbb{1}_{R_m}(x) v_m && (13) \\
&= v_0 + \sum_{m \in \text{inter(t)}} \mathbb{1}_{R_{l_m}}(x) v_{l_m} + \mathbb{1}_{R_{r_m}}(x) v_{r_m} - \mathbb{1}_{R_m}(x) v_m && (14) \\
&= \sum_{m \in \text{leaves}(t)} v_m \mathbb{1}_{x \in R_m} && (15) \\
&= f_t(x) && (16)
\end{aligned}$$

At line (14), all the terms cancel each other out except for the leaf nodes that have no children.

This concept can be extended to decompose the error of a prediction. Consider a couple $(x, y) \sim P$, then we write

$$l(y, f_t(x)) = l(y, v_0) + \sum_{j=1}^{p} l_j(y, x), \tag{17}$$

with the contribution of feature j to the error of the prediction $f_t(x)$ defined as:

$$l_j(y, x) = \sum_{\substack{m \in \text{inter(t)} \\ j_m = j}} \mathbb{1}_{R_{l_m}}(x) l(y, v_{l_m}) + \mathbb{1}_{R_{r_m}}(x) l(y, v_{r_m}) - \mathbb{1}_{R_m}(x) l(y, v_m). \tag{18}$$

If we apply the previous decomposition to all the elements of the training set, we compute a train score S_{train} for tree t, that is its improvement in train loss compared to the constant model. Therefore we let

$$S_{\text{train}} := \frac{1}{n} \sum_{i=1}^{n} [l(y_i, v_0) - l(y_i, f_t(x_i))] = \sum_{j=1}^{p} \tilde{S}_{\text{train},j}, \tag{19}$$

where the contribution $S_{train,j}$ of each feature X_j to S_{train} is

$$S_{\text{train},j} = \frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \mathbb{1}_{R_m}(x_i)l(y_i,v_m) - \mathbb{1}_{R_{l_m}}(x_i)l(y_i,v_{l_m}) - \mathbb{1}_{R_{r_m}}(x_i)l(y_i,v_{r_m}) \tag{20}$$

$$= \sum_{\substack{m\in\text{inter(t)}\\ j_m=j}} \left[\frac{1}{n}\sum_{\substack{i\in\{1,\dots,n\}\\ x_i\in R_m}} l(y_i,v_m) - \frac{1}{n}\sum_{\substack{i\in\{1,\dots,n\}\\ x_i\in R_{l_m}}} l(y_i,v_{l_m}) - \frac{1}{n}\sum_{\substack{i\in\{1,\dots,n\}\\ x_i\in R_{r_m}}} l(y_i,v_{r_m})\right] \tag{21}$$

$$= \sum_{\substack{m\in\text{inter(t)}\\ j_m=j}} \frac{n_m}{n}H(m) - \frac{n_{l_m}}{n}H(l_m) - \frac{n_{r_m}}{n}H(r_m) \tag{22}$$

$$= \text{MDI}(j). \tag{23}$$

We just showed that the MDI of feature X_j is its contribution to the average improvement in training loss with respect to the constant model. This quantity could be estimated on the oob samples instead which we explore in the next section where we introduce the oob-score.

Remark. We use the term *constant model* to designate v_0 which is the best prediction we can make when knowing nothing about the features i.e. the observed average of the target. It is the prediction that is made at the root node of the tree, before any split of the input space $\mathcal{X}$.

4.2 Oob-score

If we instead compute the aforementioned train score S_{train} on the *oob dataset* $\mathcal{D}' = (x'_i, y'_i)_{i=1}^{n'}$, we obtain

$$S_{\text{oob}} := \frac{1}{n'}\sum_{i=1}^{n'}\left[l(y'_i,v_0) - l(y'_i,f_t(x'_i))\right] = \sum_{j=1}^{p} S_{\text{oob},j} \tag{24}$$

with

$$S_{\text{oob},j} := \frac{1}{n'}\sum_{i=1}^{n'}\sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \mathbb{1}_{R_m}(x'_i)l(y'_i,v_m) - \mathbb{1}_{R_{l_m}}(x'_i)l(y'_i,v_{l_m}) - \mathbb{1}_{R_{r_m}}(x'_i)l(y'_i,v_{r_m}). \tag{25}$$

This leads to the following modified version of the MDI that we call the oob-score:

$$\text{oob-score}(j) := S_{\text{oob},j} = \sum_{\substack{m\in\text{inter(t)}\\ j_m=j}} \omega'_m H'(m) - \omega'_{l_m}H'(l_m) - \omega'_{r_m}H'(r_m), \tag{26}$$

where we let $\omega'_m = \frac{n'_m}{n'}$ the proportion of all oob samples that go through node m, and likewise for l_m and r_m the children of m. We also use a *cross impurity* function H' (using the target y'_i from the oob samples but the predicted value v_m from the train) for a node m defined as

$$H'(m) = \frac{1}{n'_m}\sum_{\substack{i\in\{1,\dots,n'\}\\ x'_i\in R_m}} l(y'_i,v_m). \tag{27}$$

If we introduce the reduction in risk with respect to the constant model

$$S := \mathbb{E}_{x,y\sim P}[l(y,v_0) - l(y,f_t(x))], \tag{28}$$

we see that S_{oob} is an estimation of S on the oob dataset.

This measure allows for an additive decomposition of the reduction in risk of a tree which is not the case with the methods we present in the next section. As an additional bonus, this measure is flexible in the choice of loss: we never specified which loss to use in the impurity functions H and H'. While it is most common to use the Mean Squared Error (MSE) loss in regression and the Brier

loss (which leads to the Gini impurity) in classification, there are other losses that may be used. For instance, it is also common to use the Shannon entropy as a measure of classification error or the Mean Absolute Error (MAE) and Poisson deviance for regression error.

However, while the additive decomposition works for a singular tree, it sadly does not hold for a random forest. Indeed, if we define the oob-score of a random forest as the average oob-score of its trees, we do not recover an additive decomposition of the reduction in risk of the forest. It is because the reduction in risk of a random forest is not the average reduction in risk of its trees. It is actually why we build random forests in the first place: an ensemble of trees is more expressive than an average tree.

4.3 Naive oob

While the oob-score is based on the so called *cross-impurity* between the values of the target on the oob points and the in-bag value of the node, one might consider using an *oob impurity* H'' computed only on oob data: instead of taking the node value that is the average value of the **in-bag samples** that go through node m, we could consider using the average value of the **oob samples** that go through node m. We therefore introduce *naive oob*, named that way by Li et al. (2019), the equivalent of MDI that uses only oob information. We thus define the oob impurity H'' as

$$H''(m) = \frac{1}{n'_m} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} l(y'_i, v'_m), \tag{29}$$

with v'_m the value of node m based on oob samples, defined in regression as

$$v'_m = \frac{1}{n'_m} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} y'_i \tag{30}$$

and in K-class classification as

$$v'_m = [p'_{m,k}]_{k=1}^K = \left[\frac{1}{n'_m} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} y'_{i,k} \right]_{k=1}^K . \tag{31}$$

Then, naive-oob writes as

$$\text{naive-oob}(j) = \sum_{\substack{m \in \text{inter(t)} \\ j_m = j}} \omega'_m H''(m) - \omega'_{l_m} H''(l_m) - \omega'_{r_m} H''(r_m). \tag{32}$$

We show in Sect. 5 that this method suffers from some of the issues of MDI that we described earlier such as the positivity bias.

4.4 UFI & MDI-oob

We now rewrite the formulas of UFI (Z. Zhou and Hooker 2021) and MDI-oob (Li et al. 2019) under our notations to see how they relate. We first look at specifically the regression case with the MSE impurity and the classification case with the Gini impurity and then show how to generalize the ideas of UFI and MDI-oob to other impurities such as the Shannon entropy.

Regression with MSE

With the MSE loss, the impurity function becomes the variance split criterion

$$H(m) = \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} (y_i - v_m)^2 \tag{33}$$

$$= \tau_m - \bar{y}_m^2, \tag{34}$$

where we let $\bar{y}_m$ and τ_m be the first two moments of node m on the train set

$$\bar{y}_m = \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_i \quad \text{and} \quad \tau_m = \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_i^2. \tag{35}$$

If we let $\bar{y}'_m$ and τ'_m be the first two moments of node m on the oob set, then the cross impurity of node m writes explicitly as

$$H'(m) = \tau'_m - 2\bar{y}'_m \bar{y}_m + \bar{y}_m^2. \tag{36}$$

And the oob-score of feature X_j writes explicitly as

$$\text{oob-score}(j) = \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m(\tau'_m - 2\bar{y}'_m \bar{y}_m + \bar{y}_m^2) - \omega'_{l_m}(\tau'_{l_m} - 2\bar{y}'_{l_m} \bar{y}_{l_m} + \bar{y}_{l_m}^2) - \omega'_{r_m}(\tau'_{r_m} - 2\bar{y}'_{r_m} \bar{y}_{r_m} + \bar{y}_{r_m}^2) \tag{37}$$

$$= \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m(-2\bar{y}'_m \bar{y}_m + \bar{y}_m^2) - \omega'_{l_m}(-2\bar{y}'_{l_m} \bar{y}_{l_m} + \bar{y}_{l_m}^2) - \omega'_{r_m}(-2\bar{y}'_{r_m} \bar{y}_{r_m} + \bar{y}_{r_m}^2), \tag{38}$$

because we have

$$\omega'_m \tau'_m = \frac{1}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} y_i'^2 = \frac{1}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_{l_m}}} y_i'^2 + \frac{1}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_{r_m}}} y_i'^2 = \omega'_{l_m} \tau'_{l_m} + \omega'_{r_m} \tau'_{r_m}$$

as R_{l_m} and R_{r_m} partition R_m. The same simplification on in-bag samples leads to $\omega_m \tau_m = \omega_{l_m} \tau_{l_m} + \omega_{r_m} \tau_{r_m}$.

The authors of UFI suggest summing both the train and cross impurity while using the in-bag weights ω_m, ω_{l_m} and ω_{r_m}. Therefore the UFI for feature X_j writes as

$$\text{UFI}(j) = \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega_m(H(m) + H'(m)) - \omega_{l_m}(H(l_m) + H'(l_m)) - \omega_{r_m}(H(r_m) + H'(r_m)) \tag{39}$$

$$= \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega_m(\tau_m - 2\bar{y}_m \bar{y}'_m + \tau'_m) - \omega_{l_m}(\tau_{l_m} - 2\bar{y}_{l_m} \bar{y}'_{l_m} + \tau'_{l_m}) - \omega_{r_m}(\tau_{r_m} - 2\bar{y}_{r_m} \bar{y}'_{r_m} + \tau'_{r_m}) \tag{40}$$

$$= 2 \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega_m(-\bar{y}_m \bar{y}'_m + \tau'_m) - \omega_{l_m}(-\bar{y}_{l_m} \bar{y}'_{l_m} + \tau'_{l_m}) - \omega_{r_m}(-\bar{y}_{r_m} \bar{y}'_{r_m} + \tau'_{r_m}). \tag{41}$$

On the other hand, the authors of MDI-oob suggest

$$\text{MDI-oob}(j) = \frac{1}{n'} \sum_{i \in \{1,\dots,n'\}} \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \left[\bar{y}_{l_m} \mathbb{1}_{x'_i \in R_{l_m}} + \bar{y}_{l_m} \mathbb{1}_{x'_i \in R_{r_m}} - \bar{y}_m \mathbb{1}_{x'_i \in R_m} \right] y'_i \tag{42}$$

$$= \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \left[\frac{\bar{y}_{l_m}}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_{l_m}}} y'_i + \frac{\bar{y}_{r_m}}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_{r_m}}} y'_i - \frac{\bar{y}_m}{n'} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} y'_i \right] \tag{43}$$

$$= \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m(-\bar{y}_m \bar{y}'_m) - \omega'_{l_m}(-\bar{y}_{l_m} \bar{y}'_{l_m}) - \omega'_{r_m}(-\bar{y}_{r_m} \bar{y}'_{r_m}) \tag{44}$$

$$= \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m(-\bar{y}_m \bar{y}'_m + \tau'_m) - \omega'_{l_m}(-\bar{y}_{l_m} \bar{y}'_{l_m} + \tau'_{l_m}) - \omega'_{r_m}(-\bar{y}_{r_m} \bar{y}'_{r_m} + \tau'_{r_m}) \tag{45}$$

when adapted with our notations.

Under these common notations, we see that UFI and MDI-oob are extremely related: UFI uses the in-bag node weights ω_m, ω_{l_m} and ω_{r_m} while MDI-oob uses the out-of-bag weights and UFI has a factor of 2 in front. Additionally, we have that as the number of samples grows, oob-score, MDI-oob and $\frac{1}{2}$UFI converge to the same value.

Indeed, in infinite sample regime, the inbag and oob weights are equal as the samples are drawn from the same distribution and so MDI-oob and $\frac{1}{2}$UFI converge to the same value. Additionally $\bar{y}'_m$ and $\bar{y}_m$ converge to $\mathbb{E}[Y_m] = \mathbb{E}[Y \mathbb{1}_{X \in R_m}]$ so $(-\bar{y}_m \bar{y}'_m)$ and $(-2\bar{y}'_m \bar{y}_m + \bar{y}^2_m)$ converge both to $-\mathbb{E}[Y_m]^2$, which implies the convergence of the oob-score and MDI-oob to the same value.

Classification with the gini criterion

In K-class classification, we let the target $y = [y_k]_{k=1}^K \in \{0,1\}^K$ be a K dimensional vector that one-hot encodes the classes. In this case, the decision function $f_t = [p_k]_{k=1}^K$ of a tree is also a K dimensional vector that predicts the probability of each class, and the value of each node is

$$v_m = [p_{m,k}]_{k=1}^K = \left[\frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_{i,k} \right]_{k=1}^K . \tag{46}$$

The Brier loss is then defined as $l(y, f_t(x)) = \sum_{k=1}^K (y_k - p_k)^2$. When using it we recover the Gini impurity criterion:

$$\begin{aligned}
H(m) &= \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} \sum_{k=1}^K (y_{i,k} - p_{m,k})^2 && (47)\\
&= \sum_{k=1}^K \frac{1}{n_m} \left[\sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_{i,k} - 2p_{m,k} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_{i,k} + n_m p^2_{m,k} \right] && (48)\\
&= \sum_{k=1}^K p_{m,k} - p^2_{m,k} && (49)\\
&= \sum_{k=1}^K p_{m,k}(1 - p_{m,k}) && (50)\\
&= 1 - \sum_{k=1}^K p^2_{m,k}. && (51)
\end{aligned}$$

Then the associated cross impurity function is:

$$\begin{aligned}
H'(m) &= \frac{1}{n'_m} \sum_{\substack{i \in \{1,\dots,n'\} \\ x'_i \in R_m}} \sum_{k=1}^K (y'_{i,k} - p_{m,k})^2 && (52)\\
&= \sum_{k=1}^K p'_{m,k} - 2p'_{m,k} p_{m,k} + p^2_{m,k} && (53)\\
&= 1 - \sum_{k=1}^K 2p'_{m,k} p_{m,k} - p^2_{m,k}. && (54)
\end{aligned}$$

UFI suggest another cross impurity function defined in binary classification as $H_{\text{UFI}}(m) = 1 - p_{m,1}p'_{m,1} - p_{m,2}p'_{m,2}$. We generalize it to K-class classification with

$$H_{\text{UFI}}(m) = 1 - \sum_{k=1}^{K} p_{m,k}p'_{m,k}. \tag{55}$$

Interestingly, we have that

$$H'(m) + H(m) = 2 - \sum_{k=1}^{K} 2p'_{m,k}p_{m,k} = 2H_{\text{UFI}}(m), \tag{56}$$

which is consistent with the regression case, despite the authors presenting the two methods in a separate way.

Now we explicitly write the oob-score and UFI of feature X_j in the following manner:

$$\text{oob-score}(j) = \sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \omega'_m \left[1 - \sum_{k=1}^{K} 2p'_{m,k}p_{m,k} - p^2_{m,k}\right] - \omega'_{l_m}\left[1 - \sum_{k=1}^{K} 2p'_{l_m,k}p_{l_m,k} - p^2_{l_m,k}\right] \tag{57}$$

$$- \omega'_{r_m}\left[1 - \sum_{k=1}^{K} 2p'_{r_m,k}p_{r_m,k} - p^2_{r_m,k}\right] \tag{58}$$

$$= \sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \omega'_m \sum_{k=1}^{K} -(2p'_{m,k}p_{m,k} - p^2_{m,k}) - \omega'_{l_m}\sum_{k=1}^{K} -(2p'_{l_m,k}p_{l_m,k} - p^2_{l_m,k}) \tag{59}$$

$$- \omega'_{r_m}\sum_{k=1}^{K} -(2p'_{r_m,k}p_{r_m,k} - p^2_{r_m,k}) \tag{60}$$

$$\text{UFI}(j) = \sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \omega_m \sum_{k=1}^{K}(-p'_{m,k}p_{m,k}) - \omega_{l_m}\sum_{k=1}^{K}(-p'_{l_m,k}p_{l_m,k}) - \omega_{r_m}\sum_{k=1}^{K}(-p'_{r_m,k}p_{r_m,k}). \tag{61}$$

We see again that UFI and oob-score converge to the same value as the weight values converge and the in and out of bag proportion too.

While the authors of MDI-oob do not explicitly give a formula in the case of classification, we extend their regression formula to fit the K-class classification with a one-hot encoded target context by changing $(-\bar{y}_m\bar{y}'_m)$ to $\sum_{k=1}^{K} -p_{m,k}p'_{m,k}$ which gives the expression

$$\text{MDI-oob}(j) = \sum_{\substack{m\in\text{inter}(t)\\ j_m=j}} \omega'_m \sum_{k=1}^{K}(-p'_{m,k}p_{m,k}) - \omega'_{l_m}\sum_{k=1}^{K}(-p'_{l_m,k}p_{l_m,k}) - \omega'_{r_m}\sum_{k=1}^{K}(-p'_{r_m,k}p_{r_m,k}). \tag{62}$$

Once again, we recover UFI with oob weights instead of in-bag ones.

Extending to other losses

While the UFI and MDI-oob methods were originally proposed only for the MSE and Gini split criteria in explicit formulas, we have seen that they respect the same idea across the two criteria, which can be extended to other losses. Indeed, following what we described above, we write a general version of UFI and MDI-oob in the following manner:

$$\text{UFI}(j) = \sum_{\substack{m\in\text{inter(t)}\\ j_m=j}} \omega_m \frac{H(m) + H'(m)}{2} - \omega_{l_m}\frac{H(l_m) + H'(l_m)}{2} - \omega_{r_m}\frac{H(r_m) + H'(r_m)}{2} \tag{63}$$

$$\text{MDI-oob}(j) = \sum_{\substack{m \in \text{inter(t)} \\ j_m = j}} \omega'_m \frac{H(m) + H'(m)}{2} - \omega'_{l_m} \frac{H(l_m) + H'(l_m)}{2} - \omega'_{r_m} \frac{H(r_m) + H'(r_m)}{2}. \tag{64}$$

With this generalization, we can take any other loss function and compute UFI and MDI-oob. For instance, if we consider the entropy loss $l(y, f_t(x)) = -\sum_{k=1}^{K} y_k \log(p_k)$ then we recover the Shannon entropy impurity

$$H(m) = \sum_{k=1}^{K} \frac{1}{n_m} \sum_{\substack{i \in \{1,\dots,n\} \\ x_i \in R_m}} y_{i,k} \log \frac{1}{p_{m,k}} \tag{65}$$

$$= \sum_{k=1}^{K} p_{m,k} \log \frac{1}{p_{m,k}}, \tag{66}$$

with the associated cross impurity

$$H'(m) = \sum_{k=1}^{K} p'_{m,k} \log \frac{1}{p_{m,k}}. \tag{67}$$

This gives the following close forms for oob-score, UFI and MDI-oob:

$$\text{oob-score}(j) = \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m \sum_{k=1}^{K} p'_{m,k} \log \frac{1}{p_{m,k}} - \omega'_{l_m} \sum_{k=1}^{K} p'_{l_m,k} \log \frac{1}{p_{l_m,k}} - \omega'_{r_m} \sum_{k=1}^{K} p'_{r_m,k} \log \frac{1}{p_{r_m,k}} \tag{68}$$

$$\text{UFI}(j) = \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega_m \sum_{k=1}^{K} \frac{p'_{m,k} + p_{m,k}}{2} \log \frac{1}{p_{m,k}} - \omega_{l_m} \sum_{k=1}^{K} \frac{p'_{l_m,k} + p_{l_m,k}}{2} \log \frac{1}{p_{l_m,k}} \tag{69}$$

$$- \omega_{r_m} \sum_{k=1}^{K} \frac{p'_{r_m,k} + p_{r_m,k}}{2} \log \frac{1}{p_{r_m,k}} \tag{70}$$

$$\text{MDI-oob}(j) = \sum_{\substack{m \in \text{inter}(t) \\ j_m = j}} \omega'_m \sum_{k=1}^{K} \frac{p'_{m,k} + p_{m,k}}{2} \log \frac{1}{p_{m,k}} - \omega'_{l_m} \sum_{k=1}^{K} \frac{p'_{l_m,k} + p_{l_m,k}}{2} \log \frac{1}{p_{l_m,k}} \tag{71}$$

$$- \omega'_{r_m} \sum_{k=1}^{K} \frac{p'_{r_m,k} + p_{r_m,k}}{2} \log \frac{1}{p_{r_m,k}} \tag{72}$$

UFI has an additional theoretical guarantee

Z. Zhou and Hooker (2021) prove in their Theorems 2 and 4 that UFI enjoys the following theoretical guarantee in classification and regression: If feature X_j is independent of Y in every hyper-rectangle subset of the feature space, then we have $\mathbb{E}[\text{UFI}(j)] = 0$.

While this result is not as strong as the minimal axiom of Reyero-Lobo, Neuvial, and Thirion (2025) that states: $\text{FI}(j) = 0 \iff X_j \perp\!\!\!\perp Y | X_{-j}$ (which is a minimal condition to perform feature selection), it is still a good theoretical guarantee that makes us favor UFI over MDI-oob. Indeed, the proofs breaks for both regression and classification when using oob weights instead of in-bag weights. The proofs require the following identity to hold: $\omega_m \bar{y}_m = \omega_{l_m} \bar{y}_m + \omega_{r_m} \bar{y}_{r_m}$ which is not satisfied when we replace ω_m, ω_{l_m} and ω_{r_m} with ω'_m, ω'_{l_m} and ω'_{r_m} respectively.

Remark. The condition "feature X_j is independent of Y in every hyper-rectangle subset of the feature space" is implied by $X_j \perp\!\!\!\perp Y | X_{-j}$.

4.5 Summary of the impurity methods

Let us shortly summarize the differences between the five impurity based methods presented above. We introduced H', a modified impurity function that takes into account information on the out-of-bag samples that go through the nodes. We suggest using this cross impurity function directly in our oob-score, to enjoy the property of additively decomposing the risk improvement of single decision trees. On the other hand, the authors of UFI (Z. Zhou and Hooker 2021) and MDI-oob (Li et al. 2019) use $\frac{H+H'}{2}$ as a modified impurity, with a disagreement on the usage of weights computed on either in-bag or out of bag samples. It is also possible to use only information on the out-of-bag samples with naive-oob, but we argue against it as it suffers from the same issues as MDI. We summarize these choices of impurity functions and weights in Tab. 1.

Method	Impurity function	Weights
MDI	H	in-bag
oob-score	H'	out-of-bag
naive-oob	H''	out-of-bag
UFI	$\frac{H+H'}{2}$	in-bag
MDI	$\frac{H+H'}{2}$	out-of-bag

Table 1: Summary of the choice of impurity functions and weights for the 5 impurity-based feature importance methods.

While these differences play a role in finite sample regime, all five methods converge in asymptotic regime, which we illustrate in the next section. In finite sample regime, we advocate for UFI, as it enjoys an additional theoretical consistency guarantee.

5 Experiments

In this section we describe an experimental setup that allows to control the relationships between the features and the target. It is a Gaussian linear model, based on a Bayesian probabilistic graphical model in which we can derive the Markov blanket of the target and thus the optimal feature selection. The setup is used the assess the performance of each method presented above for feature selection and noisy feature identification as well as computation time. We also introduce the `noised_led` dataset, a classification task where asymptotic regime is achievable in a reasonable amount of samples to illustrate the asymptotic properties of the methods. All the code and results are available in this GitHub repository[2].

5.1 Experimental setups

5.1.1 Gaussian Linear Model for regression

Since we never know the true importance of features in the real world, we construct a synthetic dataset where we model the interactions between the features and the target on a causal graph with Gaussian variables interacting linearly. It allows us to consider the Markov blanket of the target and thus see which method can be used to perform feature selection.

We construct our Gaussian linear model as proposed by Bishop and Nasrabadi (2006) in Chapt. 8.1.4: for any Directed Acyclic Graph (DAG) over $p+1$ variables, each node j represents a single continuous random variable X_j having Gaussian distribution. We let the distribution of node j conditionally on its parents pa_j be

$$p(X_j|\mathrm{pa}_j) = \mathcal{N}\left(\sum_{k\in\mathrm{pa}_j} w_{j,k}x_k + b_j, v_j\right), \tag{73}$$

[2]https://github.com/GaetandeCast/Internship/tree/main/report

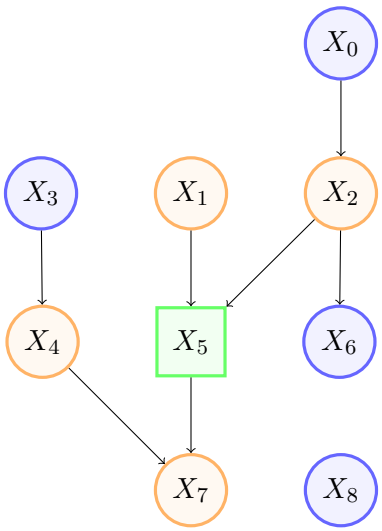

Figure 3: DAG representing the relationships between features and target in a Gaussian linear model. The green square node X_5 is the target, the orange circle nodes are the features in the Markov blanket of the target and the blue circle nodes are the features outside the Markov blanket of the target.

where $w_{j,k}$ models the impact of a parent node k on node j, b_j is a bias term and v_i is the variance of the conditional distribution of X_j. We then compute the marginal expectation of each node recursively with

$$\mathbb{E}[X_j] = \sum_{k \in \text{pa}_j} w_{j,k} \mathbb{E}[X_k] + b_i. \tag{74}$$

And the individual elements of the variance covariance matrix of $X = \{X_1, ..., X_{j+1}\}$ with

$$\text{cov}(X_j, X_k) = \sum_{l \in \text{pa}_k} w_{k,l} \text{cov}(X_j, X_l) + v_j \mathbb{1}_{j=k}. \tag{75}$$

The relationships between the features and the target that we use are described in Fig. 3. The target is X_5 represented by a green square, the features in the Markov blanket are X_1, X_2, X_4 and X_7, represented by a orange circle and the features outside the blanket are X_0, X_3, X_6 and X_8, represented by a blue circle. X_8 is disconnected from the rest of the graph, meaning it is independent of everyone. The Markov blanket B_5 of X_5 verifies :

$$X_5 | X_{B_5} \perp\!\!\!\perp X_{-B_5} \tag{76}$$

Therefore any feature outside the Markov blanket B_5 of X_5 should not be assigned any importance by a measure that can perform feature selection.

5.1.2 Noised LED dataset for classification

To show the asymptotic properties of the methods, we use a different classification dataset, that we call `noised_led`. It is based on the `led` classification problem introduced by Breiman et al. (1984): considering a seven-segment display showing the ten digits, the task is to predict the value of the digit Y when observing the seven binary features $X = \{X_1, ..., X_7\}$ that indicate whether each segment is on or off. All the possible combinations of (X, Y) are in the table of Fig. 5 and the problem is illustrated in Fig. 4 (both figures are from Louppe et al. (2013)).

We modify this problem by noising the observation of the features: each segment has a probability (fixed a 0.2 in the following) to be corrupted and have its value flipped, independently of the other features. The target is unaffected by the potential switch. This models a defective display where some segments may be turned on or off randomly. This setup allows us to show asymptotic properties as it is harder to model than the traditional version while staying simple enough that we can reach asymptotic regime in a reasonable amount of samples.

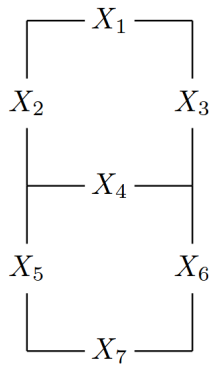

Figure 4: 7-segment display

y	x_1	x_2	x_3	x_4	x_5	x_6	x_7
0	1	1	1	0	1	1	1
1	0	0	1	0	0	1	0
2	1	0	1	1	1	0	1
3	1	0	1	1	0	1	1
4	0	1	1	1	0	1	0
5	1	1	0	1	0	1	1
6	1	1	0	1	1	1	1
7	1	0	1	0	0	1	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Figure 5: Possible values of $(X_1, ..., X_7, Y)$

This problem also satisfies the *categorical setting* of Sutera et al. (2021): the features and the target are categorical and the features are binary like the decision trees of `scikit-learn`. This allows us to show the convergence of the MDI of totally randomized trees (random forest without bootstrapping and with $p' = 1$ the number of features to choose from at each split, (Geurts, Ernst, and Wehenkel 2006)) to SAGE values proved in Sutera et al. (2021). We also use this dataset to show the convergence of the oob-score, MDI, UFI, MDI-oob and naive-oob, as it is easy to reach asymptotic regime.

5.2 Results

5.2.1 Noisy feature detection

We start by analyzing which method can statistically assign zero importance to the irrelevant feature X_8. To do so, we sample 50 datasets of 500 data points with the process described in Sect. 5.1.1 and train a random forest of 50 trees on each dataset. We then compute the feature importance of X_8 with all the methods presented earlier, and test the following hypothesis:

$$\mathcal{H}_0 : X_8 \text{ has zero importance}$$

vs.

$$\mathcal{H}_1 : X_8 \text{ has non-zero importance.}$$

We perform two tests on the population mean of the importance of feature X_8, at level $\alpha = 0.05$: a one sample Student's t-test and a one sample Wilcoxon signed-rank test, to test in both a parametric and non-parametric way, in case the normality assumption of the t-test is not validated. The results of the tests are found in Tab. 2 and 3.

Method	pop mean	std	t stat	t-test p val	t-test rejects $\mathcal{H}_0$ at $\alpha = 0.05$
MDI	0.0480	0.0068	50.084	0.0000	YES
naive-oob	0.0435	0.0056	54.984	0.0000	YES
UFI	0.0007	0.0030	1.604	0.1151	NO
MDI-OOB	-0.0048	0.0025	-13.517	0.0000	YES
oob-score	-0.0475	0.0074	-45.525	0.0000	YES
Permutation	0.0003	0.0059	0.367	0.7149	NO
SAGE	-0.0018	0.0039	-3.263	0.0020	YES

Table 2: Results of student's t-test for testing $\mathcal{H}_0$ vs. $\mathcal{H}_1$

We see that for both tests, the null hypothesis $\mathcal{H}_0$ is rejected for all methods except UFI and permutation importance. It indicates that in this situation, only these two methods were able to consistently assign zero importance to the noise feature X_8. This confirms the result of Z. Zhou and Hooker (2021): UFI assigns zero importance to features that are independent of the target in every

Method	Wilcoxon stat	Wilcoxon p val	Wilcoxon rejects $\mathcal{H}_0$ at $\alpha = 0.05$
MDI	0.000	0.0000	YES
naive-oob	0.000	0.0000	YES
UFI	484.000	0.1407	NO
MDI-OOB	0.000	0.0000	YES
oob-score	0.000	0.0000	YES
Permutation	591.000	0.6597	NO
SAGE	338.000	0.0033	YES

Table 3: Results of a Wilcoxon signed-rank test for testing $\mathcal{H}_0$ vs. $\mathcal{H}_1$

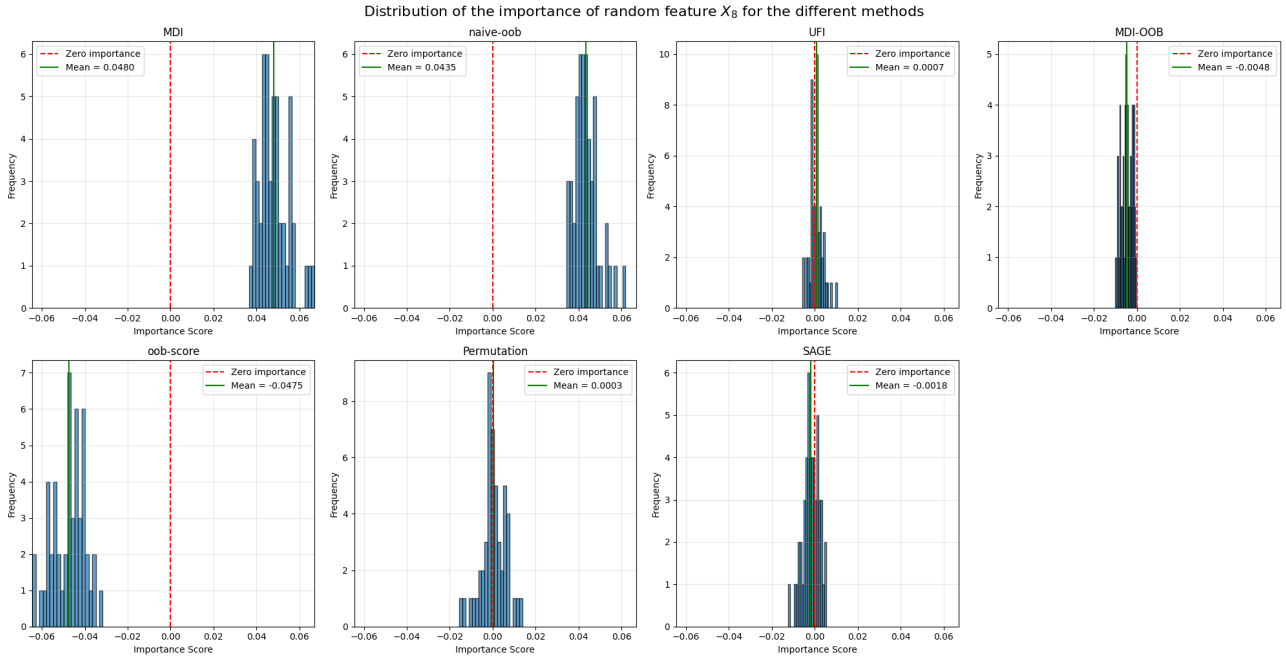

Figure 6: Distribution of the feature importance score of the irrelevant feature X_8 for all methods.

hyper-rectangle subset of the feature space $\mathcal{X}$. This also further confirms permutation importance's ability to remove irrelevant features.

The distribution of the importance score given by each method across all 50 repetitions is visualized in Fig. 6.

5.2.2 Feature selection

Using the setup from before, we also evaluate the methods in how well they perform feature selection, by looking at the top 4 ranked features and see when they match the Markov blanket of Y. The results are presented in Tab. 4.

We observe that the best performing method is permutation importance while the worse performing method is SAGE. This coincides with the analysis of Reyero-Lobo, Neuvial, and Thirion (2025): the index approximated by permutation importance satisfies the feature selection axiom, contrary to that of SAGE. For the impurity methods, we see that UFI and MDI-oob improve on MDI slightly, while oob-score and naive-oob perform way worse.

The results are visualized in Fig. 7. We see that MDI and naive-oob consistently assign positive importance to irrelevant features X_0, X_3, X_6 and X_8 while oob-score consistently assigns them a negative importance. We also see that SAGE does not detect the hard to find relevance of X_4 while permutation importance, UFI and MDI-oob do.

Method	Number of time MB was found
MDI	30/50 times (60%)
naive-oob	14/50 times (28%)
oob-score	15/50 times (30%)
UFI	31/50 times (62%)
MDI-OOB	33/50 times (60%)
Permutation	46/50 times (92%)
SAGE	13/50 times (26%)

Table 4: Amount of time each method ranked the features of the Markov blanket of the target in the top 4.

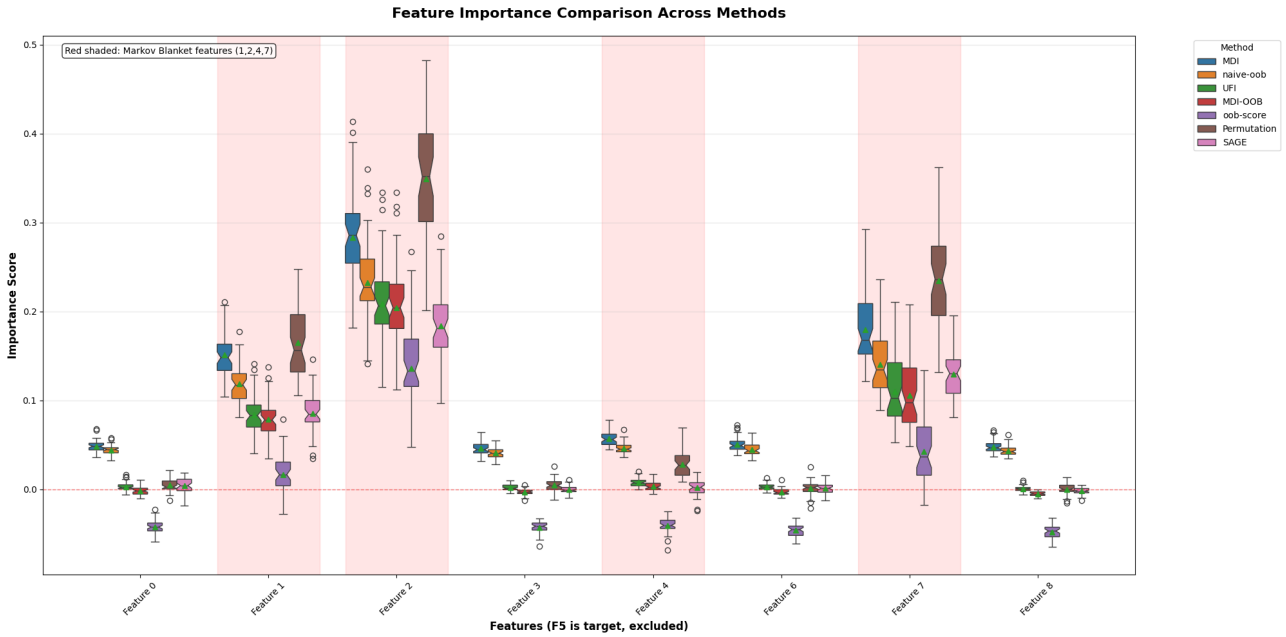

Figure 7: Box plot of feature importance measures. Feature importance was computed 50 times on random forests of 50 trees trained on 500 samples with 500 validation samples of a Gaussian linear model. The features in a red shaded area are in the Markov blanket.

5.2.3 Computation time

On the same dataset, we noted the average time it took to compute feature importance over 50 runs for each method, on the same random forest of 50 trees. We repeated the experiment for 500 train and test points and then for 1000 train and test points. The results are displayed in table 5.

In `scikit-learn`, the MDI is computed during the training of the `RandomForestClassifier` which means that the time reported here is the time it took to retrieve the value stored in the attribute `feature_importances`. The other impurity methods: UFI, MDI-oob, oob-score and naive oob all take the same amount of time, since they are computed in a very similar way, so we only included UFI. For the purpose of the report we used a high level implementation of UFI which is very inefficient, but we also developped a Cython (Behnel et al. 2010) optimized low-level implementation which is available in this[3] fork of `scikit-learn`.

We see that permutation importance is 4 times more expensive than an optimized UFI, while SAGE costs between 14 to 20 times more than UFI.

5.2.4 Asymptotic behavior

Using the `noised_led` dataset presented earlier, we now illustrate the asymptotic properties of the different methods.

[3] https://github.com/GaetandeCast/scikit-learn/tree/unbiased-feature-importance

Method	Avg compute time in ms (500 points)	Avg compute time in ms (1000 points)
MDI	16.8 (±1.3)	17.4 (±1.4)
UFI high level	5506.7 (±329.9)	13606.6 (±1149.4)
UFI low level	192.6 (±26.5)	355.1 (±62.7)
Permutation	872.8 (±121.4)	1294.8 (±170.2)
SAGE	2835.7 (±966.2)	7028.1 (±1657.0)

Table 5: Average computation time (± standard deviation) of each feature importance method for a random forest trained on 500 and 1000 points with as many test points.

First, we display in Fig. 8 the evolution of the different impurity based feature importance methods we presented: the MDI, UFI, MDI-oob, naive-oob and oob-score. We clearly see that as we increase sample size, all five methods converge to the same values, which confirms what we discussed earlier. It should be noted that asymptotic regime is really hard to reach in practice as real world datasets are finite and target-feature relationships are not as simple as the one of this toy example. This means that the choice of method is important in practice, even if results converge in asymptotic regime.

For this experiment, we used a `RandomForestClassifier` with default parameters and added a noisy feature that is sampled from a Bernoulli distribution with parameter $p = 0.5$.

Second, we display in Fig. 9 the convergence of the feature importance given by the MDI and SAGE values. To fit the theoretical framework imposed by theorem 1 of Sutera et al. (2021), we considered an `ExtraTreesClassifier` model with the parameter `max_features=1` since this produces *Totally randomized trees* (Geurts, Ernst, and Wehenkel 2006). The SAGE values are approximated with the Python package `sage`, developed by the authors of the original paper (Covert, Lundberg, and Lee 2020).

This model and dataset combination validates all the hypothesis of Sutera et al. (2021) so we observe the convergence they describe. Once again, while this result guarantees the MDI to have the desirable properties of SAGE in asymptotic regime, the assumptions are very restrictive so it is hard to generalize. Namely, the *categorical setting* imposes that the features are all categorical which is a rare occurrence in machine learning.

We also plot the evolution of the sum of the importance of the features for these two methods as well as the loss improvement of the model with respect to the constant model. We see that the approximation made by the `sage` package respects the additive decomposition of the loss improvement.

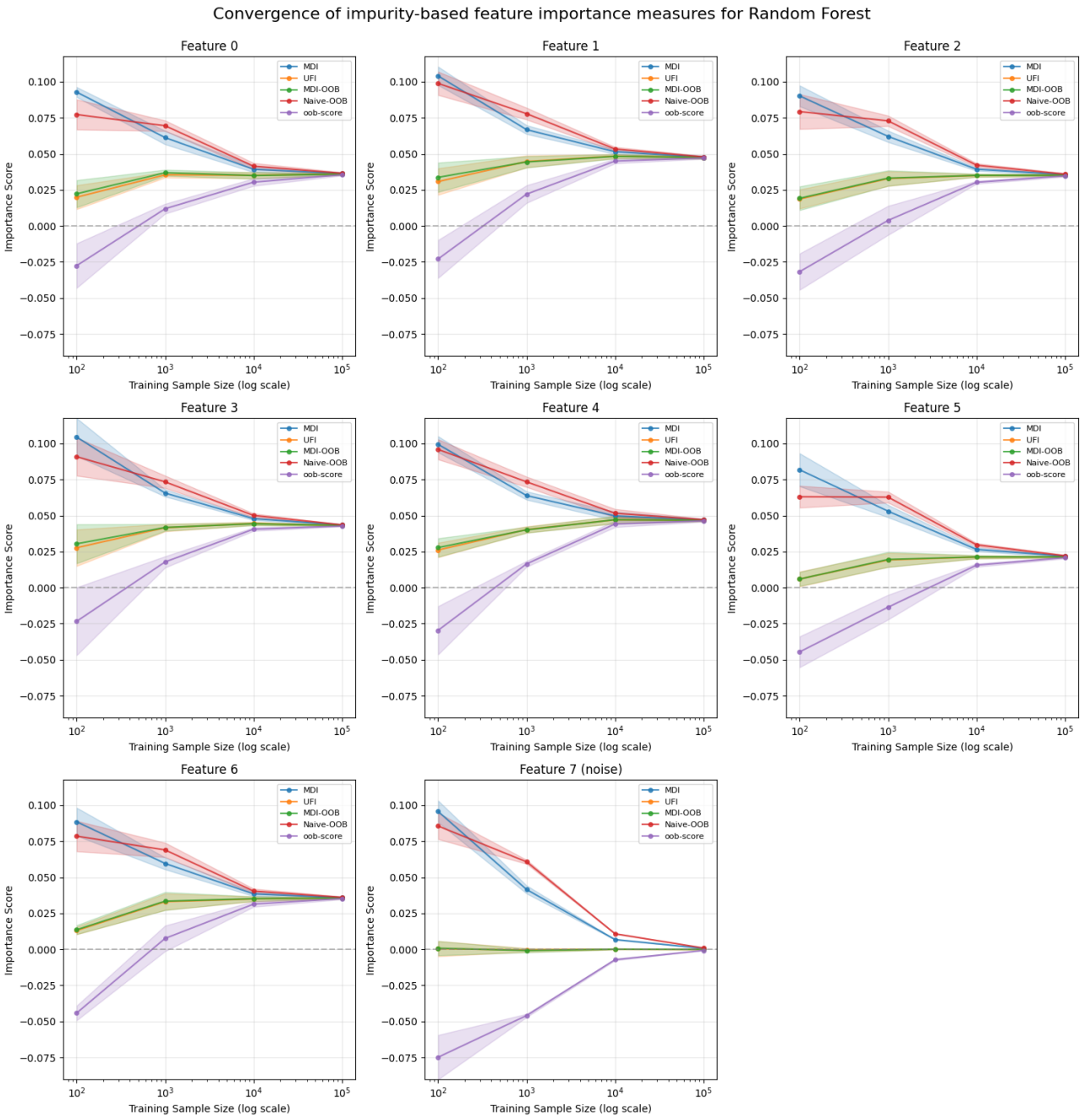

Figure 8: Evolution of the impurity based feature importance measures on the `noised_led` dataset as sample size increases.

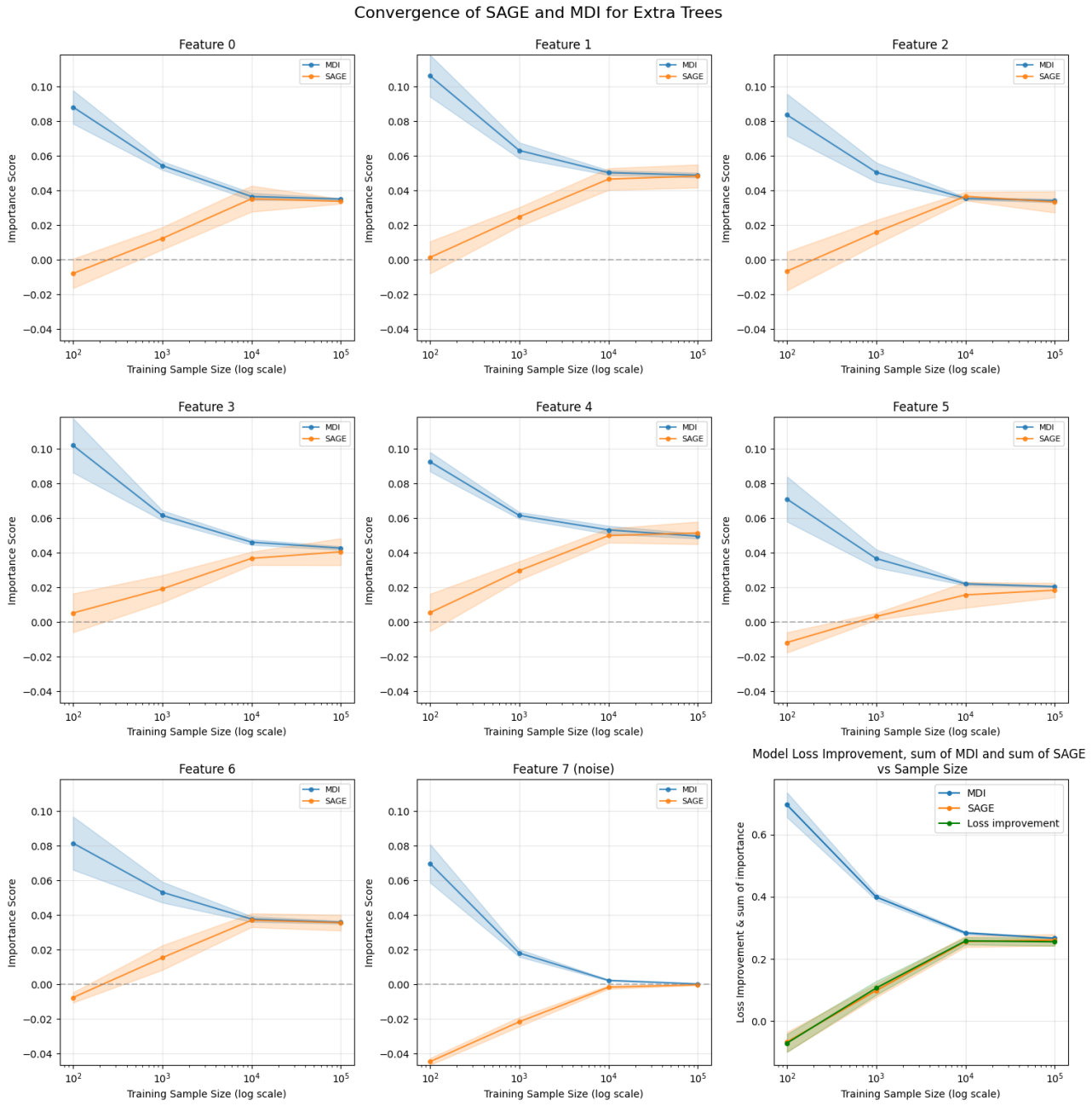

Figure 9: Convergence of the feature importance of SAGE and MDI in the categorical setting for Totally randomized trees, on the `noised_led` dataset.

6 Conclusion

Here are the original contribution that were made in this report and during the internship:

- Unify all existing impurity-based feature importance methods in a common framework to compare them.
- Propose a new impurity-based feature importance measure that additively decomposes the reduction in risk of single trees.
- Provide a way to extend UFI and MDI-oob to other impurity measures.
- Assess the ability of all presented methods to perform feature selection and identify noise features.
- Illustrate the asymptotic properties of the impurity-based methods.
- Propose a Cython optimized version of UFI for `scikit-learn`.

From the analysis conducted here, we conclude that UFI is the best alternative to the MDI for the specific constraints of `scikit-learn`. Since it can be quickly computed during fit with minimal added cost, it is a good replacement the fast but unreliable MDI. It enjoys a theoretical guarantee for noisy features that makes it preferable to MDI-oob. While this guarantee is weaker than that of permutation importance, that allows it to perform feature selection, it is a much cheaper method to compute. Practitioners that want a method of feature importance that is able to perform feature selection have to pay the cost of permutation importance, a method also implemented in `scikit-learn`. For their high computational cost and inability to perform feature selection, we discourage the use of SAGE values, despite their added benefit of decomposing model loss improvement in an additive way.

In the future, we would like to explore the possibility of proving or disproving $\text{UFI}(j) = 0 \implies X_j \perp\!\!\!\perp Y | X_{-j}$ and giving a formal proof that the MDI is strictly positive in finite samples regime. We would also like to consider the case of gradient boosted trees (Friedman 2001), and see if the methods studied here apply in this case.

References

Behnel, Stefan et al. (2010). "Cython: The best of both worlds". In: *Computing in Science & Engineering* 13.2, pp. 31–39.

Bishop, Christopher M and Nasser M Nasrabadi (2006). *Pattern recognition and machine learning*. Vol. 4. 4. Springer.

Breiman, Leo (1996). "Bagging predictors". In: *Machine learning* 24.2, pp. 123–140.

— (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Breiman, Leo et al. (1984). *Classification and regression trees*. Wadsworth.

Chamma, Ahmad, Denis A Engemann, and Bertrand Thirion (2023). "Statistically valid variable importance assessment through conditional permutations". In: *Advances in Neural Information Processing Systems* 36, pp. 67662–67685.

Covert, Ian, Scott M Lundberg, and Su-In Lee (2020). "Understanding global feature contributions with additive importance measures". In: *Advances in neural information processing systems* 33, pp. 17212–17223.

Friedman, Jerome H (2001). "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics*, pp. 1189–1232.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel (2006). "Extremely randomized trees". In: *Machine learning* 63.1, pp. 3–42.

Harrell, Frank E. Jr. and Thomas Cason (2017). *Titanic Dataset*. OpenML dataset ID 40945. Source: Vanderbilt Biostatistics. URL: https://www.openml.org/d/40945.

Hooker, Giles, Lucas Mentch, and Siyu Zhou (2021). "Unrestricted permutation forces extrapolation: variable importance requires at least one more model, or there is no free variable importance". In: *Statistics and Computing* 31.6, p. 82.

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis (2006). "Unbiased recursive partitioning: A conditional inference framework". In: *Journal of Computational and Graphical statistics* 15.3, pp. 651–674.

Hothorn, Torsten and Achim Zeileis (2015). "partykit: A modular toolkit for recursive partytioning in R". In: *The Journal of Machine Learning Research* 16.1, pp. 3905–3909.

Janzing, Dominik, Lenon Minorics, and Patrick Blöbaum (2020). "Feature relevance quantification in explainable AI: A causal problem". In: *International Conference on artificial intelligence and statistics*. PMLR, pp. 2907–2916.

Kohavi, Ron and George H John (1997). "Wrappers for feature subset selection". In: *Artificial intelligence* 97.1-2, pp. 273–324.

Li, Xiao et al. (2019). "A debiased MDI feature importance measure for random forests". In: *Advances in Neural Information Processing Systems* 32.

Loecher, Markus (2022). "Unbiased variable importance for random forests". In: *Communications in Statistics-Theory and Methods* 51.5, pp. 1413–1425.

Louppe, Gilles et al. (2013). "Understanding variable importances in forests of randomized trees". In: *Advances in neural information processing systems* 26.

Molnar, Christoph (2020). *Interpretable machine learning*.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Reyero-Lobo, Angel, Pierre Neuvial, and Bertrand Thirion (2025). "A principled approach for comparing Variable Importance". In: *arXiv preprint arXiv:2507.17306*.

Saabas, Ando (2017). "Interpreting random forests. 2014". In: *URL: https://blog.datadive.net/interpreting-random-forests/*.

Sandri, Marco and Paola Zuccolotto (2008). "A bias correction algorithm for the Gini variable importance measure in classification trees". In: *Journal of Computational and Graphical Statistics* 17.3, pp. 611–628.

Shapley, Lloyd et al. (1953). "A value for n-person games". In: *Contributions to the Theory of Games II*, pp. 307–317.

Strobl, Carolin et al. (2008). "Conditional variable importance for random forests". In: *BMC bioinformatics* 9.1, p. 307.

Sutera, Antonio et al. (2021). "From global to local MDI variable importances for random forests and when they are Shapley values". In: *Advances in Neural Information Processing Systems* 34, pp. 3533–3543.

Zhou, Zhengze and Giles Hooker (2021). "Unbiased measurement of feature importance in tree-based methods". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 15.2, pp. 1–21.