https://lib.uliege.be https://matheo.uliege.be

Master thesis : Local permutation importances for random forests

Auteur : Weyders, Pierre-François
Promoteur(s) : Geurts, Pierre
Faculté : Faculté des Sciences appliquées
Diplôme : Master : ingénieur civil en science des données, à finalité spécialisée
Année académique : 2020-2021
URI/URL : http://hdl.handle.net/2268.2/13302

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.

University of Liège
Faculty of Applied Sciences

Department of Electrical Engineering and Computer Sciences

Local permutation importances for random forests

Master Thesis - Graduation Studies conducted for obtaining the Master's degree in Computer Sciences and Engineering

by

Pierre-François Weyders (S160729)

Supervisors :

Prof. Pierre Geurts
Phd. Antonio Sutera

Academic Year 2020-2021

Abstract

The best performing machine learning models to date are by essence difficult to interpret like neural network or consist in ensemble of models like random forests.

These black-box models don't benefit from any direct tool to interpret their predictions unlike simpler models like linear regression that embodied in their definition an explanation of their predictions.

Although black-box models perform well, their deployment to real life application might be undermined by their lack of understanding.

Interpreting the predictions of a model is a key component to the study of each model. This thesis focus on understanding each prediction of an ensemble of tree. The methods that are studied and compared provide an importance to each feature that summarizes the dependence of the model on the feature to correctly predict the outcome.

To the end of understanding the individual predictions of a model, the local methods defined by Saabas and SHAP are compared with two newly introduced methods based on the mean decrease in impurity and mean decrease in accuracy global methods defined by Breiman 2001 [3].

In the second part of the thesis, the methods are used to infer local regulatory networks and are shown to outperform previous state of the art methods.

Acknowledgements

First and foremost I would like to express my special thanks of gratitude to my supervisors Pierre Geurts and Antonio Sutera who have always been present and who have introduced me to the topics of local feature rankings and gene regulatory networks. I am grateful for the regular follow-up of my work they have ensured and for giving their constructive feedback and valuable advice.

I also particularly thanks my mother and sister for their continuous support throughout my studies and for all their care and love.

Table of Contents

1 Introduction **4**

2 Background **5**

2.1 Models . . . 5

2.1.1 Decision Tree algorithm . . . 5

Classification tree: Impurity Reduction . . . 6

Regression: Variance Reduction . . . 6

2.1.2 Random Forest - ExtraTrees . . . 6

2.2 Metrics . . . 7

2.2.1 Normalization . . . 7

l1 - norm . . . 7

l2 - norm . . . 7

2.2.2 Spearman Rank Correlation . . . 8

2.2.3 Jaccard Index . . . 8

2.2.4 Method Expressivity . . . 9

2.2.5 AUROC and AUPR . . . 9

2.3 Related Work: Global Methods . . . 10

2.3.1 Global MDI . . . 11

2.3.2 Global MDA . . . 11

Sensitivity to correlation: . . . 12

Influence of Random split selection parameter: . . . 13

2.4 Related Work: Local Feature Ranking . . . 13

2.4.1 Local-MDI . . . 13

2.4.2 Shapley Values . . . 14

2.4.3 SHAP . . . 15

2.4.4 SAABAS . . . 16

2.5 Datasets . . . 18

2.5.1 Friedman1 . . . 18

2.5.2 Friedman2 . . . 18

2.5.3 Boston . . . 18

2.5.4 Iris . . . 18

2.5.5 Dataset for the linear regression problem . . . 19

3 Local Mean Decrease of Accuracy **20**
3.1 Method . . . 20
3.1.1 Model-Agnostic Method . . . 20
3.1.2 Tree-specific local MDA . . . 21
3.2 Implementation . . . 21
3.3 MDA - Regression Task . . . 22
3.3.1 Error Measurement . . . 22
3.3.2 Variations . . . 23
Graph propagation weights - Uniform parameter . . . 23
Willingness to propagate . . . 24
Weighting on the Tree Ensemble . . . 24
3.4 MDA - Classification Task . . . 24
3.4.1 Error Measurement . . . 25
3.4.2 SHAP and SAABAS rankings . . . 25

4 Experiments and Results **26**
4.1 Results and Experiments - Regression task . . . 26
4.1.1 Spearman Rank Correlation . . . 26
4.1.2 Results on dataset with unused variables . . . 27
Impact of N_T . . . 27
4.1.3 Study of the impact of K on rankings . . . 29
Parameters . . . 29
Spearman Correlation and Jaccard Index . . . 29
How much did rankings changed? . . . 31
Comparison with Forest Global MDI . . . 32
4.1.4 Comparative example with Linear Regression . . . 33
Linear Regression parameters . . . 33
Parameters of the random forest . . . 33
Example1 - Global ranking similar to Ground Truth . . . 33
Example2 - Global Ranking different than Ground Truth 36
Conclusion . . . 37
4.1.5 Ranking Expressivity . . . 38
4.1.6 Performances . . . 38
4.2 Classification Task . . . 39
4.2.1 Study of the Spearman correlation coeficient on the Iris dataset . . . 39
Ranking Expressivity . . . 40

5 Application to Gene Network Inference **43**
5.1 Gene Network Inference Problem . . . 43
5.1.1 Gene Regulatory Network . . . 43
5.1.2 Deriving gene regulation through genes' expression . . . 44
5.2 Background - Dyngen . . . 44
5.2.1 Generation of Genes' expressions . . . 45
5.2.2 Cell-specific Gene Regulatory Network . . . 46
5.2.3 Methods . . . 46

LIONESS + Pearson 46
SSN 47
pySCENIC 48
5.3 Experiments 48
5.3.1 Methodology 48
5.3.2 Scoring Method 49
5.3.3 Methods Comparison 50
Tree parameters 50
Remark on AUPR & AUROC: 50
Results 1 - Graph visualisation 50
Results 2 - In depth tabular visualisation 52
meanAUROC: 52
meanAUPR: 52
5.3.4 Gene Interactions Example on cell id 56
5.4 Conclusion 59

6 Conclusion **60**

A Expressivity plots - Regression task **61**

Chapter 1

Introduction

The topic of this thesis is to study and compare local ranking methods applied to ensembles of trees with the aim to interpret each prediction of a model. Understanding a model is a key part of its deployment to critical application in the fields of justice, medicine and strategic decisions.

The performance of ensembles of trees is largely studied in the literature as well as methods that extract a global ranking of the features such as the global MDI method defined by Breiman[3]. This work contributes to the understanding of the ensembles of trees through the study of methods that locally identify the variables that are important for a prediction.

The second contribution of the thesis is the further application of local ranking methods to the Gene Regulatory Network inference problem. The results of the local methods are compared with ground-truth data inferred from a simulator and their performances are compared to the state of the art method of that field.

Chapter 2

Background

The background section describes the decision tree algorithm that is the base model used throughout the work. Important metrics such as the Jaccard Index and the Spearman correlation are defined in this section, as well as an overview of the the main global and local feature ranking methods used in the literature.

2.1 Models

The topic of the master thesis being the application of feature ranking methods to ensemble of trees in order to interpret their predictions, this section describes the decision tree algorithm and its further generalisation to an ensemble of trees.

2.1.1 Decision Tree algorithm

Decision trees are parts of the supervised learning framework which means that the model optimizes a loss function defined over a learning set (**LS**). The loss function aims at selecting the best internal model's parameters by optimizing them over **LS** so that the predictions of the model are close to the real Y values of **LS**. Decision tree can handle both classification and regression problems and can be used on categorical or qualitative features.

The (binary) decision tree successively partitions the **LS** in two sub-**LS** according to the evaluation of a particular variable. Both the variable and its evaluation value are found to be optimal at each step and are the parameters learned. The decision tree was first described in 1984 by Breiman[2]. It consists in a tree where interior nodes test a value and the branches correspond to the evaluation result and the leaves are labelled with a class or value.

The goal of the supervised model is to learn differentiating the samples of the **LS** based on their value and output Y. The decision tree algorithm decomposes the differentiation problem by choosing at each node the splitting criterion that distinguishes the best their output values. The differentiation is referred to as

a reduction in impurity (classification problem) or as the reduction in output variance (regression problem).

Classification tree: Impurity Reduction

The splitting criterion is a 2-tuple of a variable and its value that reduces the most the impurity in the child nodes. The measure of impurity typically is the entropy or the Gini criterion. The result of a split is the creation of two sub-trees (sub-spaces) and the decrease in impurity is weighted by the number of samples that reaches either of the sub-trees from the parent node. The samples further propagate in these child nodes where other splits may occur.

With regard to the entropy as a measure of uncertainty and information, the best split is the one that removes the most the uncertainty that is left in the two sub-paces after the split occurs. The bigger the decrease in impurity, the better a model distinguishes a class from the others.

Using the formalism of L.Wehenkel [24], for a feature A with $A(\mathbf{LS})$ its set of different values, $\mathbf{LS}_a$ the subset of samples o from $\mathbf{LS}$ such that $A(o) = a$ and I a measure of impurity:

$$\Delta I(LS, A) = I(LS) - \sum_{a \in A(LS)} \frac{|LS_a|}{|LS|} I(LS_a) \tag{2.1}$$

Regression: Variance Reduction

In the regression problem, the biggest reduction of variance is sought and the splitting criterion that leads to the biggest decrease in variance is the best split:

$$\Delta I(LS, A) = var_{y|LS}\{y\} - \sum_{a \in A(LS)} \frac{|LS_a|}{|LS|} var_{y|LS_a}\{y\} \tag{2.2}$$

2.1.2 Random Forest - ExtraTrees

The lack of performance of trees has been shown to moslty come from their high variance (Geurts,2002[9]) that comes from the strong sensitivity of a tree to the variability of the learning set. Hastie[17] proposes to combine several models to reach better performances than single models. Several trees compose an ensemble and the prediction is either averaged over all trees (regression task) or corresponds to the class of the majority votes (classification task).

Random forests are ensembles of trees that contain some degrees of freedom at critical parts of the decision tree algorithm. Trees individually loose a bit of their predictive accuracy when randomization is added. However, averaged over the forest, results have shown that the ensembles of trees have a lower variance.

ExtraTreeRegressor and *ExtraTreeClassificator*, implemented in the *scikit-learn* library, are the two models used in this report. They allow to specify randomization factors such as the maximum number of features to be considered at each split. This parameter, K, will be extensively considered in the latter sections. If K is set to 1, the forest is perfectly randomized as no feature selection occurs. When K is smaller than the number of features, K features are randomly sampled from the feature space X. In Extra Trees, Geurts2006[8], random splits are sampled for the randomly selected features and the best split is chosen among them.

2.2 Metrics

This section of the report describes the metrics that are used to interpret and compare results.

2.2.1 Normalization

Normalization on ranking values is applied when the rankings are compared with regards to the importance being associated to each feature instead of its position in the ranking. The normalization projects each value v in the range $|v| = [0, 1]$.

It is also necessary to consider the absolute values of rankings in order to compare methods that output signed values to methods that outputs unsigned importance values. As this will be the case latter, all rankings are normalized and features importance are kept non-negatives.

l1 - norm

The $l1$-norm is known to give less influence on large values and tends to produce more 0's. $l1$-normalisation consists in dividing each element of a vector by the sum of the absolute values of its elements. $\forall x_i \in X$, the normalized vector N is composed of:

$$n_i = \frac{x_i}{\sum_i |x_i|} \tag{2.3}$$

The $l1$-normalization has the advantage that the absolute values of the normalized data sum to 1.

l2 - norm

The square norm puts more emphasis on large values and reduces the influence of smaller ones. The $l2$-norm is the square root of the sum of the squared values. $\forall x_i \in X$, the normalized vector N is composed of:

$$n_i = \frac{x_i}{\sum_i x_i^2} \tag{2.4}$$

Normalization is not a metric itself but it enables the use of some of the follwing metrics that compare rankings:

2.2.2 Spearman Rank Correlation

The Spearman rank correlation coefficient is a non-parametric measure of the *monotonicity of the relationship between two datasets (rankings)*[1]. The coefficient takes value between $[-1, 1]$, where a value of 1 means that ranking are identical and -1 that rankings are in reverse order. 0 implies no correlation between the rankings.

Let u_i and v_i be the rank of the ith variable of two samples, the Spearman correlation coefficient r_S corresponds to:

$$\begin{aligned} r_S &= \frac{n\sum_{i=1}^{n} u_i v_i - (\sum_{i=1}^{n} u_i)(\sum_{i=1}^{n} v_i)}{\sqrt{[n\sum_{i=1}^{n} u_i^2 - (\sum_{i=1}^{n} u_i)^2][n\sum_{i=1}^{n} v_i^2 - (\sum_{i=1}^{n} v_i)^2]}} \\ &= 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)} \quad \text{where} \quad d_i = u_i - v_i \end{aligned} \tag{2.5}$$

The second equality being a shortcut approximation formula.

The Spearman coefficient has been chosen over other correlation measures such as Pearson as it assumes no linear dependency between variables and focuses instead on the monotonic relationship between two sets. Both methods would fail at detecting correlation if the dependency between variables is more complex. Correlation will be extensively used to compare the feature rankings of several methods and no linear assumption is made between ranking's feature's values.

2.2.3 Jaccard Index

The Jaccard Index is a similarity coefficient measured between the elements of two sets. It informally consists in the ratio between the number of common elements and the number of different elements. Mathematically, it is the size of the intersection divided by the size of the union. The Jaccard index results in a proportion and $J \in [0, 1]$.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.6}$$

The index will be used to compare subsets of rankings e.g. study the similarity of the p most important features of two rankings of N features ($p < N$). The Jaccard Index is computed on vectors whose data is the id of features. If $N = 6$, computing the index on the $p = 3$ most important features of two rankings indicates to what extent the two methods are close enough to detect

[1] Description from the documenation of scipy.stats.spearmanr - August 2021

the same important variables. Averaging the Jaccard Index over all samples (all local rankings) shows the propensity of the two methods to detect the same variables in the top3 of the most important features.

It is interesting to observe how different can be the results of methods that share the same aim (to provide a ranking of features) and that are based on the same data (a random forest).

2.2.4 Method Expressivity

The feature rankings provided by different methods are usually compared using the Spearman correlation coefficient that uses the relative position of each feature among the two rankings to compute a similarity measure. The expressivity of a method consists in measuring the gap between the biggest and least importance values among all rankings of a method, all gaps are then reported on the same graph for different methods. The graph enables a visual analysis of the importance values assigned by each method. If the method finds that all features are *equally important to explain the prediction*, the gap will be small while if some features are found to be very important *to explain the prediction*, the gap will be higher.

Observing the plot of all gaps makes possible to quickly compare methods and see which one identifies or not some features to be more important than the others. In order to compare methods with importances in the same range of values, the ranking are normalized with the *l1*-norm that makes values sum to 1.

2.2.5 AUROC and AUPR

The area under the Receiver Operating Characteristic curve and the area under the Precision/Recall curve are two measures used for judging the quality of a classifier. A definition of the two measures exists for multi-class classification problems. In this report, the AUROC and AUPR metrics will be used in the context of a binary classifier in the Gene Network Inference section.

The metrics are used to compare the ground truth data to a model's prediction i.e. a vector of probability whose magnitudes are linked with the certainty of predicting a specific class. The main advantage of the metrics is to provide a measure of a classifier's quality without inferring a fixed threshold on probabilities' values to separate predictions between the two classes.

The ROC curve consists in a plot representing the true positive rate (TPR or *Recall*) as a function of the false positive rate (FPR). Each dot of the curve corresponds to the evaluation of the model for a specific threshold value. Each different value of the prediction vector is typically associated to a threshold. The metric therefore summarizes a the quality of a model by considering all possible probability thresholds and builds a confusion matrix for each threshold.
The area under the ROC curve represents the degree of separability between

the predictions and thus, how much a model is capable of distinguishing the two classes. If a model has no predictive power and cannot distinguish the classes, then its AUROC value will be close to 0.5. The more the area under the curve gets closer to 1 or 0, the more a model is able to distinguish the classes. An AUROC close to 0 still requires to switch the labels of the predicted classes to be a performing classifier.

1. $TPR(k) = recall(k) = \frac{TP(k)}{TP(k)+FN(k)} = \frac{TP}{P}$
2. $FPR(k) = \frac{FP(k)}{FP(k)+TN(k)} = \frac{FP}{N}$
3. FP : false positive (negatives classified as positive)
4. TP : true positive (positive predicted as positive)
5. TN : true negative (negative well classified as negative)
6. FN : false negative (positive classified as negative)

The *precision* - number of true positive among all predicted positive - is compared to the *recall* - proportion of well predicted positive - in the PR curve. This metric is often used to quantify a classifier's quality in a strong class-imbalance situation. By definition, the metric shows how the proportion of positives detected (TPR) relates to the proportion of true positive detected among all positive predictions. Indeed, a model can reach a high TPR and therefore detect most of the positive elements at the cost of a large number of false positive predictions, which is captured in the *precision* indicator. An AUPR close to 1 is synonym of quality and a random classifier would have an AUPR close to $\frac{P}{P+N}$ which corresponds to the average precision of a random classifier [19].

$$Precision = \frac{TP(k)}{TP(k)+FP(k)}$$

AUPR and AUROC are the main metrics one could find in the literature to assess models' quality in the gene inference problem [5].

2.3 Related Work: Global Methods

This section presents the two main global feature importance methods that were originally described by Breiman[3] in 2001. The first method is the Mean Decrease in Impurity that evaluates the importance of a feature with regard to its contribution in impurity reduction. The second method, Mean Decrease of Accuracy uses the predictive accuracy of a feature to derive its importance.

2.3.1 Global MDI

The MDI method is based on the splitting criteria that is used to expand the trees which usually consists in the Entropy, the Gini criterion of a variance reduction measure. At each split, the decrease in impurity is recorded for each variable x_j of the feature space X that is used to build the splitting condition. The weighted-average of all decreases in impurity associated to x_j among all trees yields the feature importance of the feature x_j. The weight associated to each decrease in impurity corresponds to the size of the node[22] i.e. the ratio of the number of samples passing through the node (N_t for $node_t$) and the total number of samples in the learning set (N).

Using the formalism of A. Sutera [22] and Louppe et al.[13], the mean decrease of impurity importance of feature $x_j \in X$ is:

$$VIMP_{mdi}(x_j) = \frac{1}{N_T} \sum_{T} \sum_{t \in T : v(s_t) = x_j} p(t) \Delta i(s_t, t) \tag{2.7}$$

Where $p(t)$ is the ratio N_t/N at node t and $v(s_t)$ is the feature of split s_t.

MDI provides a non-null importance value to all features involved in the forest. The method extracts the features that discriminate most the output i.e. that contribute the most to the reduction of impurity. Relevant features are useful to discriminate the output, these features will be used in splits within the forest at some point and lead to a decrease of impurity. If on the contrary, less-relevant features are assumed to have a lower discriminating power, they would only be seldomly considered in the forest and their likelihood to be selected as the splitting variable at a node close to the root is low with regard to relevant features.

As mentioned in Breiman,2001 [3], an advantage of the MDI method is its computational simplicity as the decrease in impurity is already computed in each node of the tree. One only needs to navigate once in each tree of forest to derive the MDI importance of all variables.

2.3.2 Global MDA

The second method calculates the importance of a feature as the mean decrease of accuracy(Breiman [3]) implied by adding noise to features' values. The noise usually consists in a permutation of the values or in adding a Gaussian noise to a variable. The MDI method requires the definition of an impurity metric and MDA requires to define an error function to measure the decrease in accuracy between the predictions of the sample and its noised form.

The importance associated to each variable x_j of the feature space X is the average over the forest of the mean decrease in accuracy measured between each sample s_i and s_i^j, where s_i^j correspond to s_i with noise on the jth feature.

In its original definition, the MDA measure is applied to ensemble of trees using bootstrapping which consists in training each tree of the ensemble on a

different subset of the sample set. The importance of a feature x_j is measured by its mean decrease of accuracy based on the out-of-bag (OOB) error when this feature is removed. The removal of a feature is simulated by permuting its value[22] and its impact is measured on all OOB samples. Global-MDA is also referred as the permutation importance: it is indeed the decrease in a model score when a single feature value is removed. The OOB context, in which f is a predictor (tree), allows to define $VIMP_{MDA}$ as follow: given a set of samples D of input/output pairs (x,y), a loss function L and a shuffled version of D according to feature x_j, D_j, the MDA importance of feature x_j for D corresponds to[22]:

$$VIMP_{G-MDA,f}(x_j, f, D, D_j) = \frac{1}{|D|}(\sum_{(x,y)\in D_j} L(f(x), y) - \sum_{(x,y)\in D} L(f(x), y)) \tag{2.8}$$

The dependence on the particular permutation is removed by averaging the importance over several permutation schemes.
Previous definition naturally scales to random forests as the averaged feature importance measured over all trees. With T being the forest and **LS** the learning sample[22]:

$$VIMP_{G-MDA,T}(x_j, T, \mathbf{LS}) = \frac{1}{N_T}\sum_{i=1}^{N_T} VIMP_{G-MDA,f}(x_j, T_i, \mathbf{LS}_i^{oob}, \mathbf{LS}_{i,j}^{oob}) \tag{2.9}$$

The main assumption of MDA is that features that are important to predict Y are important contributors to the model capacity to predict Y. Permuting the values of a feature breaks the statistical link inferred by the tree between the feature and Y[1] and this permutation is said to mimic the removal of the feature[22]. A feature with a high feature importance is found to be very important for the model as its removal leads to large prediction errors. Less relevant variables are instead assumed to be of lesser importance for the model and therefore should lead to smaller predictive errors.

The loss functions considered in practice always provide non-negative importances and one usually use the Mean Square Error or the Mean Absolute Error. A importance close to zero means the features is useless to predict a sample. The method is model agnostic and the following discussion concerns its application on trees and forests.

Sensitivity to correlation: Auren et al.[1] mention the sensitivity of the permutation to the features' correlation structure. As said, removing the variable x_j destroys its association to Y and all other variables x_i, $(i \neq j)$. The decrease in accuracy can then imply the dependence of Y on x_j or the dependence of any x_i on x_j. Conditional permutation framework allows to reduce this effect (Strobl et al.[21]).

Suggested property 1 *Auren et al.[1] suggest the following property: Tree ensemble variable importance measures are less accurate (more likely to rank unimportant variables as important ones) when input variables are correlated.*

Influence of Random split selection parameter: In their 2010 paper[7], Genuer et al. suggested two observations:

1. The optimal number of features to consider at each split (K) is different for model accuracy and for variable importance purposes. This means that the sequence of splits that reduces the most the variance in leaves and therefore provides the best model accuracy isn't necessarily the sequence that will bring out the most important variables.

2. The authors also mentioned that larger values of K increases the magnitude of variable importance for truly important variables. This conforms the intuition that when K is bigger, the model can better choose a splitting variable among the important ones. Choosing K small with regard to the set of possible features may give a higher importance than merited to less-important/irrelevant variables or even correlated variables.

Suggested property 2 *Auren et al.[1] suggest this second property for variables importance measures: importance measures are more accurate for ensemble of trees which include more randomly sampled split variables at each split.*

2.4 Related Work: Local Feature Ranking

2.4.1 Local-MDI

The Local-MDI method is formalized by A. Sutera, G. Louppe, V.A. Huyn-Thu, L. Wehenkel and P. Geurts in *From global to local MDI variable importances for random forests and when they are Shapley values.* The method is defined as the local version of Global-MDI and a straight relation exists between the definition of both methods.

Let T be a forest of trees t, $\mathbf{x}$ a sample and x_j a variable from the feature space X. The local-MDI importance associated to a feature x_j for a sample $\mathbf{x}$ corresponds to the sum of all differences in impurity (i) over all nodes of a tree traversed by $\mathbf{x}$ that uses x_j as splitting variable. t_{x_j} is the successor of node t traversed by $\mathbf{x}$ as feature x_j is tested at the split. For a forest, the importance of x_j is computed over all trees and averaged on the forest's size ($|T|$).

$$VIMP_{localMDI}(x_J, \mathbf{x}) = \frac{1}{|T|} \sum_{T} \sum_{t \in T: v(s_t) = x_M} i(t) - i(t_{x_j}) \tag{2.10}$$

The method collects all differences in impurity caused by x_j along all trajectories of $\mathbf{x}$ in the forest. The bigger the difference in impurity, $i(t) - i(t_{x_j})$, the better x_j contributed in distinguishing $\mathbf{x}$ from samples with different values.

Such x_j is good at discriminating $\mathbf{x}$ and therefore is a relevant feature to predict Y from sample $\mathbf{x}$.
Local-MDI is very efficient if the node impurities that are measured during the training process are stored.

Subsequently, the link between Local-MDI and Global-MDI consists in inferring the global measure from the local one by averaging the importance of each feature among all samples of the learning set. For a learning set $\mathbf{X}$ of N samples $\mathbf{x}_i$ $i = 1, ..N$, the importance associated to x_j corresponds to:

$$VIMP_{MDI}(x_j) = \frac{1}{N} \sum_{i}^{N} VIMP_{localMDI}(x_J, \mathbf{x}_i) \tag{2.11}$$

Note that unlike Global-MDI, the importance the local method associates to a feature can be negative. The negative importances associated to a sample are mitigated by the greater positive importances that the same feature has in other samples as the importances measured by the global methods are exclusively non-negative. A negative importance would mean that after using x_j in a split, the impurity of the successor node is bigger than its parent. This means that x_j does not participate at discriminating $\mathbf{x}$ at a particular node and therefore, *helps predicting other instances.* Negative importances are difficult to interpret locally as, when $K > 1$, the chosen splitting feature is the one that discriminates the most instances globally i.e. over the training set.

2.4.2 Shapley Values

Shapley values comes from the cooperative game theory where each game is a pair of a set of players and a function that assigns a value to any coalition of players. The method defined by Lloyd Shapley consists in assigning payouts to players depending to their individual contribution to the total[16]. In the context of model-interpretation, the players are the features, the game is the prediction task and a coalition is a subset of built on features values.

The goal of Shapley values is to *explain* how each feature of a model contributes to its prediction. The importance of each feature is measured as the contribution of the feature to the discrepancy of the prediction with regard to the average output of the model (average output on training data) i.e. Shapley values explain how each feature contributes in a prediction that is different than the average value. Christoph Molnar [16] provides a one sentence definition: *the Shapley value of a feature is the average marginal contribution of a* ***feature value*** *accross all possible coalitions.*

Practically, from the feature space X of p features, $x_1, ..., x_j \in X$. To study the impact of $v(x_j)$ - value of feature x_j - to the subset of feature values S (coalition), one first randomly samples values for all features that are not in the coalition & x_j which provides a first prediction: P_1. Another prediction, P_2 is obtained by randomly sampling another value for x_j (which could still

be the same as in P_1). The contribution of x_j to the coalition is defined as $\phi_j = P_1 - P_2$. The sampling step is often repeated to get better estimates. The Shapley value is defined as a contribution accross all possible coalitions i.e. summed and averaged over all coalitions, the complexity of the method increases exponentially with the number of features. A common approximation consists in computing contributions on a subset of all possible contributions [16].

Christoph Molmar presented the following formalism:

$$\phi_j(v) = \sum_{S \subseteq \{x_1,\ldots,x_p\} \setminus \{x_j\}} \frac{|S|!(p-|S|-1)!}{p!} (v(S \cup \{x_j\}) - v(S)) \tag{2.12}$$

with $v(S)$ the prediction for the feature values in the coalition S that are marginalized over the features that are not included in S.

The book of Interpretable Machine Learning [16] emphasizes that Shapley values are interpreted as an average contribution and not as difference of prediction when a feature is removed from a model. Shapley values can either be used in classification problems as in regression tasks. The term feature value corresponds to a numerical or categorical value of a feature.

The Shapley value is the only solution to the payout attribution problem that satifies the properties of **Efficiency, Symmetry, Dummy (Null-player)** and **Additivity**.

1. **Efficiency:** the method decomposes the total payout among all features.
2. **Symmetry:** features with the same contribution get the same Shapley value.
3. **Dummy:** feature with no contribution over all coalition receives a Shapley value of 0 - null importance.
4. **Additivity:** for a game resulting in combined payouts p and p^*, Shapley value of features sums too: $\phi_j + \phi_j^*$.

Using Shapley values as indicators of features importance is straightforward. By essence, relevant variables contribute a lot to define Y and the distinguishing ability of a feature is here measured with regard to how much it intrinsically defines the value prediction instead of how much it separates the target prediction from others (impurity reduction of MDI).

2.4.3 SHAP

SHapley **A**dditive Ex**p**lanation (SHAP) is a model agnostic method defined by Lundberg et al [14] in 2017. To explain any model, SHAP builds a surrogate model that consists in a linear function: a linear sum of features' contribution (SHAP values) that equals the prediction. Any explanation of the original model's prediction is seen as a model itself called the *explanation model*. SHAP

is said to derive feature importances through the decomposition of the prediction

SHAP values are defined as the *Shapley values of a conditional expectation function of the original model*:

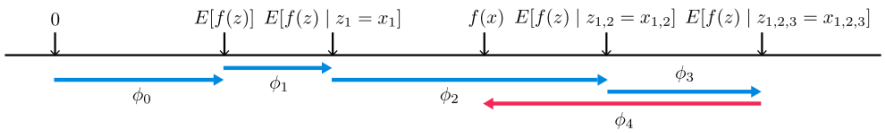

Figure 2.1: **SHAP additive decomposition of the prediction[14]:** Each prediction is decomposed as the sum of a base value and the contribution each features of a coalition Z.

Using the formalism of Lundberg et al. [14], the prediction of the model f of a sample $\mathbf{x}$ using the features z of the coalition S can be written as: $f_x(z) = \mathbb{E}[f_x(z)|z_S]$. The prediction is explained using a subset of features and as models usually cannot handle arbitrary patterns of input features, $f_x(z)$ is conditionally approximated.

As shown in Figure 2.1, SHAP values explain how to successively find the prediction by successively adding knowledge on features starting from the base value (average value: $\mathbb{E}[f(z)]$). The order in which the expectation is built i.e. in which features are added matters and SHAP values are built from averaging contribution across all orderings.

The python package *SHAP* provides the *TreeExplainer* method which implements the exact algorithm to compute SHAP values on trees and random forests.

2.4.4 SAABAS

The Saabas method is described in a blog post[18] dating from October 2014 and is implemented in the python package *TreeInterpreter*. This method is not model agnostic as feature importances are extracted from the structure of the tree. In short, Saabas decomposes the output into a sum of contributions measured at each node, from the root till the leaf.

Saabas brought the novelty of breaking down the prediction in terms of value changing along the prediction path. The method defines the prediction as the sum of the features' contributions and the bias (mean of the training set i.e. value at root node). And the prediction $f(x)$ of a tree can be written as:

$$f(x) = Bias + \sum_{k=1}^{K} contrib(\mathbf{x}, k) \tag{2.13}$$

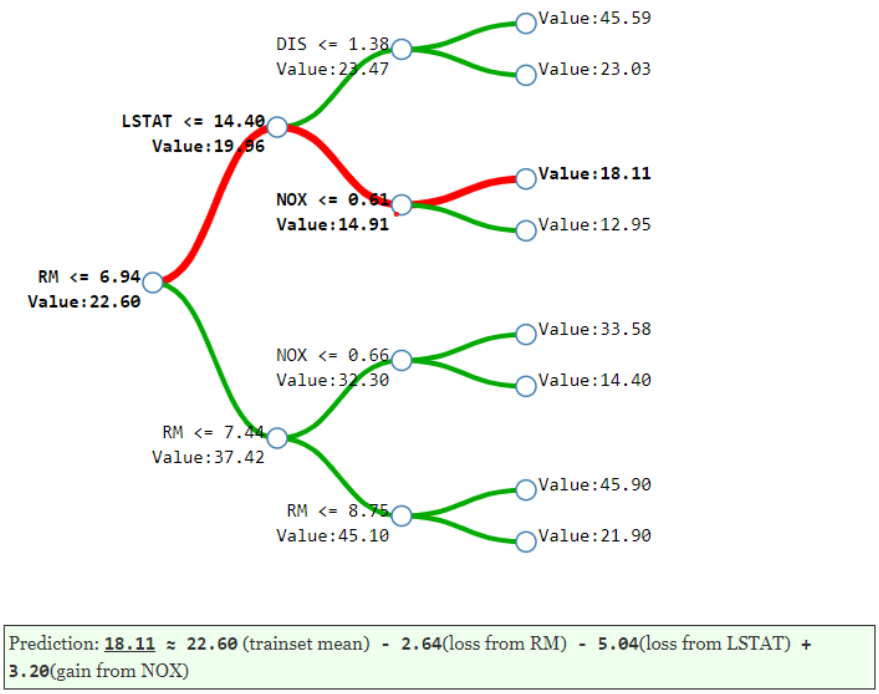

Figure 2.2: **Saabas output decomposition[18]:** Each prediction is decomposed as the sum of feature contributions and the output's mean value.

Figure 2.2 shows the output decomposition within the structure of the tree for a particular sample[18].

With K the number of features and $contrib(\mathbf{x}, k)$ the contribution of the kth feature to the sample $\mathbf{x}$. This definition is superficially similar to a linear regression[18] - $f(x) = a + bx$ - where the weight vector (b - features' contributions) isn't fixed but depends on the decision path and therefore depends on the other features. Saabas depends on the structure of the tree.

The application of Saabas to an ensemble of trees is straightforward as the prediction of the ensemble is the average prediction over all trees. The importance of a feature linked to a sample $\mathbf{x}$ corresponds to the average contribution of the feature over all trees.

The method explicitly provides a specific explanation for each prediction. As mentioned in previous sections, relevant variables should be mainly used in the trees and be responsible for splits that reduce the most the impurity (they help distinguishing the predictions). The same logic applies to induce that relevant variables should be responsible for the biggest contributions.

The hyperparameter K is known to have an impact on MDA and MDI but it

is unclear how it influences the rankings issued by Saabas on deep/shallow trees and big/small forests. The method proposed by Onda Saabas is poorly studied in the literature and mostly used to compare importance with other methods.

The method is implemented in the python package *TreeInterpreter* and can be applied to decision trees and random forests of the *scikit-learn* package.

2.5 Datasets

Several datasets are imported from the *scikit-learn* library. 20% of the data set is used to test the model and the other 80% is used to train the model. All results shown in this work come from the evaluation of a model on the test set.

2.5.1 Friedman1

The Friedman1 dataset is defined by Friedman 1991[6] and consists in 10 independent features where only the 5 first features are used to define the output:

$$y(X) = 10\sin(\pi x_0 \times x_1) + 20(x_2 - 0.5)^2 + 10x_3 + 5x_4 + noise * N(0,1) \quad (2.14)$$

500 samples are generated from this dataset.

2.5.2 Friedman2

The Friedman2 dataset is defined by Friedman 1991[6] and consists in 4 independent features uniformly distributed on the intervals:

1. $0 < x_0 < 100$
2. $40\pi < x_1 < 560\pi$
3. $0 < x_2 < 1$
4. $1 < x_3 < 11$
5. $y(x) = (x_0^2 + (x_1 x_2 - \frac{1}{x_1 x_3})^2)^{0.5} + noiseN(0,1)$

100 samples are drawn from this dataset generator.

2.5.3 Boston

The Boston dataset is loaded from sklearn, it contains 506 samples and has 13 features that are all real and positive.

2.5.4 Iris

The iris dataset is also loaded from sklearn, it contains 150 samples and has 4 features. The dataset is used for classification problems.

2.5.5 Dataset for the linear regression problem

This dataset is designed for the purpose of an experiment. It contains 5 variables that are sampled uniformly in the range $[0, 1]$ and that are uncorrelated with each other. The output is computed as a linear combination of the features. The dataset defines a linear regression problem. The dataset generator defines 4 parameters:

1. weights: weights of the linear regression $= [3, 5, -7, -2, 6]$
2. b: real value, intersection with y-axis, the value is set to 0
3. feature_range: multiply the values of the randomly sampled feature with a fixed real value r. This extends the range of a feature to $[0, r]$. The range is set to $[2, 2, 2, 2, 2]$
4. feature_translate: add the values of a feature by a fixed real value t. The final range of a feature x_j corresponds to $[t, t + r]$. The translation term is set to $[0, 0, 0, 0, 0]$.

Chapter 3

Local Mean Decrease of Accuracy

This chapter introduces a new method to derive local feature importances. The method is inspired by the global Mean Decrease in Accuracy method. The local MDA method assigns to each feature an importance that corresponds to its impact on the model's prediction when the feature has been removed from the model.

3.1 Method

3.1.1 Model-Agnostic Method

A straight relation exists between the local and global MDA methods - this section shows that the local method is a component of the global one. Consider a model f, a sample $\mathbf{x}$, its prediction $\tilde{y} = f(\mathbf{x})$, true value y and a loss function L. One defines the importance of a feature x_j with regard to $\mathbf{x}$ as the decrease in accuracy of the model when the value of x_j is noised (permuted/shuffled) in $\mathbf{x}$ producing a new sample x^j. The following equation uses the OOB context as OOB samples are available to test the model. The second equation shows the difference in accuracy between the model's predictions of $\mathbf{x}$ and its shuffled version $\mathbf{x}^j$.

$$\begin{aligned} VIMP_{local-MDA}(x_j, f, \mathbf{x}, \mathbf{x}^j)^1 &= L(f(x^j), y) - L(f(\mathbf{x}), y) \\ VIMP_{local-MDA}(x_j, f, \mathbf{x})^2 &= L(f(\mathbf{x}), f(x^j)) \end{aligned} \tag{3.1}$$

A classical error function E consists in the squared error: $E(a, b) = (a - b)^2$ and the feature importance is averaged over several permutations of x_j[22]:

$$VIMP_{local-MDA}(x_j, f, \mathbf{x})^1 = \mathbf{E}_{j*}[VIMP_{local-MDA}(x_j, f, \mathbf{x}, \mathbf{x}^{j*})^1] \tag{3.2}$$

A feature's local importance in a forest is the averaged sum of its importance value over all trees. The global method is derived as we average each sample importance over the whole dataset.

Note that the permutation of a variable aims at artificially removing that feature from the model[3]. The prediction of a model divested by a relevant feature should be strongly different from the original prediction. This effect is captured locally by the MDA method and the detected important variables are those whose removal causes an important change in the model's prediction.

Unless explicitely specified, MDA corresponds to the local MDA method likewise MDI, it refers to the local MDI method.

3.1.2 Tree-specific local MDA

The agnostic method uses feature values permutations to mimic the impact of removing a feature from the model. Depending on the base predictor used, model-specific implementations of MDA can differently implement the artificial removal of a feature. The method proposed in this thesis is specifically designed for trees as it uses their structure to artificially remove a feature from the tree model. The tree-based MDA method is referred to as MDA in the next sections.

In order to simulate the removal of a feature, MDA modifies the trajectory of a sample in a tree at all nodes whose splitting variable is the target feature. Say one studies the local impact of feature x_j on a sample $\mathbf{x}$: each time the sample navigates through a node n testing x_j, it is propagated in both child nodes, i.e. right and left with weights corresponding to $\frac{N_{right}}{N}$ and $\frac{N_{left}}{N}$. N_{left} being the number of samples from the learning set that crossed n and propagated in the left sub-trees and N the total number of learning samples that navigated through n. The sample is then propagated in both sub-trees the same algorithm applies in all sub-trees visited by the sample. As a conclusion, the algorithm builds a probability distribution on the tree's leaves that corresponds to the likelihood of the sample to reach the leaf if x_j is removed from the sample. This procedure provides a different result than retraining a tree on the feature space X excluding x_j and is much more efficient. The probability distribution on the leaves depends on the distribution of x_j in the training data.

The importance of feature x_j at predicting $\mathbf{x}$ is measured as the difference between the prediction $f(\mathbf{x})$ and the prediction built as the sum of leaves values weighted by the distribution derived after removing x_j from the sample and propagating $\mathbf{x}$ in the tree.

3.2 Implementation

The application of MDA to a set of S samples and F features provides an array of size $(S \times F)$, each of the S rows contains the feature importances of

the F features. As the importances are averaged over all trees, the algorithm propagates all samples in each tree and builds an array where each value $a_{i,j}$ corresponds to the importance of x_j for the sample $\mathbf{x}_i$.

For each tree, the method loops over all samples and successively computes all leaves probability distributions that result from propagating a sample in the tree of which a feature as been removed.

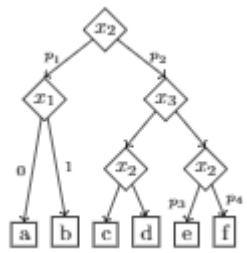

Figure 3.1: **Example of decision tree:** The sample's decision path is x_2 - x_3 - x_2 and the sample propagates in the right child of the node testing x_1.

Figure 3.1 shows how a sample whose feature x_2 has been removed propagates in the decision tree. The sample reaches a with the a probability p_1 and it reaches e & f with the respective probabilities p_2p_3 & p_2p_4. The prediction of the model corresponds to $p_1a + p_2p_3e + p_2p_4$ and $p_1 + p_2p_3 + p_2p_4 = 1$.

3.3 MDA - Regression Task

Once the algorithm computed a prediction for $\mathbf{x}$ and the removed feature x_j, one still needs to compare the prediction of the disrupted sample with the non-disrupted prediction of tree/forest. This section presents the two functions that compute the difference of the predictions at the scope of the forest. The result of this evaluation is the importance associated to x_j.

3.3.1 Error Measurement

The python module computing the MDA importances implements two loss functions: the Mean Square Error over all trees and the squared error between the forest prediction and the average prediction over disrupted trees.

Let's consider a random forest T of N_T trees (t_i), the prediction of the forest corresponds to the average predictions over the trees: $Y = \frac{1}{N_T}\sum_{i=1}^{N_T} y_i^*$, y_i^* being the prediction of t_i for a sample $\mathbf{x}$. The prediction of t_i resulting from the disrupted sample $\mathbf{x}_j$ is $\tilde{y}_{i,j}$.

1. Mean Square Error - MSE: $VIMP_{x_j} = \frac{1}{N_T}\sum_{i=1}^{N_T}(y_i^* - \tilde{y}_{i,j})^2$

2. Squared Error: $VIMP_{x_j} = (Y - \frac{1}{N_T}\sum_{i=1}^{N_T}\tilde{y}_{i,j})^2 = (\frac{1}{N_T}\sum_{i=1}^{N_T} y_i^* - \frac{1}{N_T}\sum_{i=1}^{N_T}\tilde{y}_{i,j})^2$

The following equations compare both expressions of the loss function and derive a link between them. Let's first extent the definition of both measures:

$$\frac{1}{N_T}\sum_{i=1}^{N_T}(y_i^* - \tilde{y}_{i,j})^2 = \frac{1}{N_T}\sum_{i=1}^{N_T}[(y_i^*)^2 + (\tilde{y}_{i,j})^2 - 2 \times \tilde{y}_{i,j} \times (y_i^*)]$$

$$(\frac{1}{N_T}\sum_{i=1}^{N_T} y_i^* - \frac{1}{N_T}\sum_{i=1}^{N_T}\tilde{y}_{i,j})^2 = \frac{1}{N_T^2}(\sum_{i=1}^{N_T} y_i^*)^2 + \frac{1}{N_T^2}(\sum_{i=1}^{N_T}\tilde{y}_{i,j})^2 - \frac{2}{N_T^2}\sum_{i=1}^{N_T}\sum_{k=1}^{N_T}(y_k^* \times \tilde{y}_{i,j}) \tag{3.3}$$

Let's now generically compare the squared terms:

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} x_i^2 \geq \frac{1}{n^2}(\sum_{i=1}^{n} x_i)^2 \quad \text{Derived from Cauchy-Schwarz inequality} \\
&\frac{1}{n}\sum_{i=1}^{n} x_i^2 = \frac{1}{n^2}(\sum_{i=1}^{n} x_i)^2 + \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - x_j)^2 \\
&\frac{1}{n}\sum_{i=1}^{n} x_i^2 = \frac{1}{n^2}(\sum_{i=1}^{n} x_i)^2 + \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x_n})^2 \quad \text{with } \bar{x_n} \text{ being the average value}
\end{aligned} \tag{3.4}$$

$\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x_n})^2$ represents the variance of x among all trees - i.e. it corresponds to the variance in models' predictions and the variance among disrupted predictions.

The MSE loss function contains slightly more information than the other measure as it integrates the variance coming from the forest.

3.3.2 Variations

In addition to the loss functions, other parameters can be tuned. Their impact on the rankings will be studied in the next chapter.

Graph propagation weights - Uniform parameter

In addition to permuting features values, Breiman[2001] [3] proposed, in its description of the tree-Global-MDA, to uniformly propagate samples among child nodes when the removed feature is the splitting criterion. Each trajectory in both sub-trees is therefore weighted by 0.5 and not weighted by the proportion of training samples that navigated through the child nodes.

The parameter *uniform* releases the weighting from the dependence of x_j's distribution in the training sample when set to *True*.

Willingness to propagate

This variation implements a lazy version of MDA that does not remove the feature x_j from the sub-trees that are not part of the decision path of the sample. When $v(n)$[1] $= x_j$, the sample is propagated in the two sub-trees of its child nodes. Further node-tests that use x_j in the sub-tree that does not belong to the sample's decision path don't propagate the sample in both child nodes anymore. The sample recovers its lost feature and the sub-tree is explored as $\mathbf{x}$ wasn't modified.

When the parameter *Split* is set to False, the lazy propagation scheme is implemented and child-nodes propagation only occurs at the nodes testing x_j that belong to the sample's decision path.

Weighting on the Tree Ensemble

Depending on the hyperparameter K, it might be that a feature x_k is never used in a tree. Removing such feature from the sample does not affect the prediction and a null importance is associated to this feature in a particular tree. Features that are not selected are less important than others to discriminate the output and therefore might also not be locally relevant i.e. important for the sample. However, when the feature importance is summed over all trees and divided by the forest size, one considers trees that brought a null contribution to the sum.

When the parameter *weighted* is set to anything but *'n'*, the sum of importances is divided by the number of trees that bring a positive contribution to the sum. A null-importance however is not synonym of null-information as it brings the important information that x_k is useless to predict $\mathbf{x}$ within a tree.

3.4 MDA - Classification Task

To simply recall the MDA algorithm: each sample of a set is propagated in each tree of the forest - one feature being removed at a time. Each time a sample reaches a node testing the removed variable, the sample is propagated in both child nodes with a uniform probability or a weight: $\frac{N_{left}}{N}$ or $\frac{N_{right}}{N}$. N being the number of samples crossing that node and N_{left} the number of samples reaching the left child of the node.

Within the code, probabilities are propagated till the leaves. In a classification task, a leaf is not a real value but a probability distribution built from the classes of the samples reaching the leaf. These probability distributions are averaged over all leaves and the distributions are weighted by the probability of the sample to reach them. Therefore, the result of predicting sample $\mathbf{x}$ with the removed feature x_j is a probability distribution over all possible classes.

[1] Splitting feature at node n

3.4.1 Error Measurement

One loss function is defined to measure the disturbance that removing a feature induced on the model's prediction. The loss function directly provides the importance of a variable and should therefore indicate how sensible the model is to output a different forecast when a feature is removed.

1. If altering a variable leads to a big error, say the prediction of another class or a big reduction in the likelihood associated to class previously predicted by the model - the feature should be marked as important.

2. If the probability of the model's prediction (class with biggest likelihood) remains similar - the feature should be marked as poorly relevant.

3. Note p^* the probability of the model's prediction (class c^*) and $\tilde{p}$ the probability associated to c^* after removing x_j from the sample $\mathbf{x}$. The feature importance is defined as:

$$VIMP_{MDAClassifiction}(\mathbf{x}, x_j) = (1 - p^*) - (1 - \tilde{p}) \tag{3.5}$$

3.4.2 SHAP and SAABAS rankings

The *TreeInterpreter* and SHAP methods provide for each sample $\mathbf{x}$ the contribution of a feature to the prediction of each class. It takes the form of an array with as many contributions as they are classes in the classification problem.

In order to compare methods, it is the contribution to the model's predicted class that summarizes the importance of a feature: the feature importance of x_j for $\mathbf{x}$ is the contribution of x_j in the class predicted by the original model.

Chapter 4

Experiments and Results

4.1 Results and Experiments - Regression task

4.1.1 Spearman Rank Correlation

This section especially compares MDA with SHAP, SAABAS and MDI. The aim of this section is to observe how MDA rankings compare to other methods when the parameters of the MDA are modified.

The previous chapter defined the parameters:

1. *Uniform*, which propagates samples in the child nodes with a probability of 0.5 after a split involving the target feature x_j.

2. *Split* controls the propagation of the sample in the sub-trees that do not belong to the decision path.

3. *Error_function*, that selects the MSE function when set to True.

The forests are composed of 300 trees, K is set to 2 and the models are trained on the Boston dataset. The average Spearman correlation coefficients resulting from comparing the MDA rankings with MDI, SHAP and Saabas are provided in Table 4.1. A variable whose name is crossed in the following table is set to False.

avg Corr, MSE	Uniform, split	Uniform, ~~split~~	~~Uniform~~, split	~~Uniform~~, ~~split~~
MDI	0.815	0.814	0.769	0.752
$SHAP$	0.808	0.804	0.789	0.768
$SAABAS$	0.829	0.824	0.8	0.782

avg Corr, ~~MSE~~	Uniform, split	Uniform, ~~split~~	~~Uniform~~, split	~~Uniform~~, ~~split~~
MDI	-0.771	-0.785	-0.792	-0.806
$SHAP$	-0.756	-0.774	-0.784	-0.782
$SAABAS$	-0.779	-0.797	-0.765	-0.801

Table 4.1: **Average Spearman correlation:** Computed between all methods and averaged over the test samples.

The Table corresponding to MSE shows that activating the *Uniform* and the *Split* parameters always increases the correlation with the other methods. However, *Uniform* has a much bigger importance than *Split* as it is responsible for the biggest change in correlation.

The results for ~~*MSE*~~ are truly surprising as the rankings are extremely anti-correlated with MDI, SHAP and Saabas. A possible explanation comes from the fact that the measure of the square error doesn't capture the variance in the prediction of the trees that results from both the normal and the perturbed samples. It might then change the intuition about the measure and provide variables that are important to interpret the prediction with the smallest 'importance' and give irrelevant feature with a bigger 'importance'. The average Spearman correlation detects a strong anti-monotonic relation between MDA and the other rankings. This result is open for further research and experiments.

The MDA rankings are computed in the rest of the work with the following parameters: ~~*Uniform*~~, *Split* and *MSE*.

4.1.2 Results on dataset with unused variables

This section uses the Friedman1 dataset to train the model. This dataset has 10 features and only the 5 first features are used to derive the output. The rankings provided by the local method are studied with regard to different forest's hyperparamaters.

Impact of N_T The number of trees in an ensemble is critical for its predictive ability [17]. Therefore, it is interesting to compare the rankings inferred by the different local methods and observe to what extent, methods provide similar information on the interpretation of a prediction as the number of trees (N_T) varies.

Table 4.2 shows the Spearman correlation between the different methods for different values of $N_T = [50, 100, 500]$. The parameter K is set to 2 in the right part - the forest is therefore not considered as a perfectly randomized predictor and K is set to 1 in the left part of the table. Note that trees are fully expanded.

avg Corr, $N_T = 50$	$MDA_{K=2}$	$MD_{K=2}I$	$SHAP_{K=2}$	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$
MDI	0.65	–	–	0.32	–	–
$SHAP$	0.73	0.66	–	0.29	0.52	–
$SAABAS$	0.70	0.66	0.85	0.32	0.51	0.79

avg Corr, $N_T = 100$	$MDA_{K=2}$	$MDI_{K=2}$	$SHAP_{K=2}$	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$
MDI_1	0.73	–	–	0.36	–	–
$SHAP_1$	0.77	0.71	–	0.39	0.57	–
$SAABAS_1$	0.75	0.70	0.88	0.40	0.54	0.84

avg Corr, $N_T = 500$	$MDA_{K=2}$	$MDI_{K=2}$	$SHAP_{K=2}$	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$
MDI	0.79	–	–	0.42	–	–
$SHAP$	0.79	0.79	–	0.45	0.65	–
$SAABAS$	0.80	0.80	0.92	0.46	0.65	0.93

Table 4.2: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples.

From Table 4.2, one can see that increasing the size of the forest results in increased correlation coefficient. Methods' interpretation become more similar as there are more trees. SHAP and Saabas are very close in all models and despite the model on the right being fully randomized ($K = 1$), their interpretation of the model remains similar. MDI and even more MDA seem to be very dependent on the number of features considered at each split. $K > 1$ increases the likelihood of the model to choose a splitting features that explains the model i.e. features: x_0, x_1, x_2, x_3 or x_4 which are the relevant features of the model as the output is only built from the 5 first features.

Only the first 5 variables are known to be relevant for the output and the Jaccard Index is measured between two rankings on their respective top5 features. We won't observe the propensity of the methods to detect the 5 relevant features as the most important for their local predictions but we will observe how similar the set of the best 5-features of the rankings are.

avg Jaccard Index, $N_T = 100$	$MDA_{K=2}$	$MD_{K=2}I$	$SHAP_{K=2}$	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$
MDI	0.74	–	–	0.49	–	–
$SHAP$	0.73	0.78	–	0.48	0.63	–
$SAABAS$	0.75	0.78	0.90	0.50	0.63	0.88

Table 4.3: **Average Jaccard Index:** The Jaccard Index computed over the top5 features of each ranking.

Table4.3 shows that the average proportion of features detected by two rankings to be in the top 5 is also slightly dependent from the parameter K and that methods gain in similarity as K increases.

4.1.3 Study of the impact of K on rankings

Suggestion 2, described in the **Global MDA** section, mentions that higher values of K provide more accurate variable importances. The motivation of this section is to study how rankings computed by the same method change as K becomes greater using the Spearman correlation coefficient and the Jaccard Index. Results are also compared between methods to see whether bigger values of K unify the rankings methods.

Parameters

The dataset *Friedman2* (4 features) is used in this section and the models consists in two random forests with 500 trees and $K = 1$ or $K = 4$ (4 being the total number of feature in Friedman2). The MDA local rankings are computed with the following parameters: $Split = True$, $Uniform = True$ and $MSE = True$. The rankings with subscript 1 are computed on the totally random forest and rankings with subscript 4 are inferred from the trees where the splitting variable is chosen among all 4 features.

Spearman Correlation and Jaccard Index

Table4.4 and Table4.5 display the methods' pair-wise Spearman correlation coefficient average on all samples. The correlation coefficients from Table4.4 are computed on the forest where $K = 1$ and the coefficient of Table4.5 are computed on the forest built with $K = 4$.

We observe that, indeed, the coefficient K has a strong influence on the rankings. The different methods output rankings that are more similar (correlated) as more variables are being considered at each split. The correlation measured in Table4.5 shows that all rankings are on average very correlated.

To observe how much the rankings change and become similar as K increases, the Jaccard Index is used to compare the two most important variables identified by each method for each sample i.e. the two features with the highest importance. The index is the size of the intersection divided by the size of the

union. The Jaccard Index takes a finite number of values and for the comparison of vectors of size 2: $J(Frank1, Frank2) \in [0, 0.33, 1]$ and $Frank$ being the features' id of the two most important variables.

Table4.6 and Table4.7 display the results for restpectively $K = 1$ and $K = 4$. One can observe that Saabas and Shap methods always identify the same two features as the most important ones and that as K increases the similarity of the set compared increased too. Indeed, when $K = 4$, the similarity score is close to 1 between all methods. The test set associated to Friedman2 contains 20 samples so a Jaccard similarity value of 0.96 means that for one sample, the two methods only share one similar feature among their top-2 and that the top-2 of the 19 other samples contain the same features. The position of the variables in the top-2 being not considered.

avg Corr	MDA_1	MDI_1	$SHAP_1$
MDI_1	0.64	–	–
$SHAP_1$	0.73	0.63	–
$SAABAS_1$	0.68	0.62	0.96

Table 4.4: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples. Forest with $K = 1$

avg Corr	MDA_4	MDI_4	$SHAP_4$
MDI_4	0.89	–	–
$SHAP_4$	0.94	0.89	–
$SAABAS_4$	0.94	0.91	0.96

Table 4.5: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples. Forest with $K = 4$

avg Jaccard Index	MDA_1	MDI_1	$SHAP_1$
MDI_1	0.73	–	–
$SHAP_1$	0.73	0.73	–
$SAABAS_1$	0.73	0.73	1

Table 4.6: **Average of the Jaccard Index:** The Jaccard Index is measured on the id of the two most important features detected by each method. The Index is measured on each sample and averaged over all test samples. Forest with $K = 1$

avg Jaccard Index	MDA_4	MDI_4	$SHAP_4$
MDI_4	0.96	–	–
$SHAP_4$	0.96	1	–
$SAABAS_4$	0.96	1	1

Table 4.7: **Average of the Jaccard Index:** The Jaccard Index is measured on the id of the two most important features detected by each method. The Index is measured on each sample and averaged over all test samples. Forest with $K = 4$

How much did rankings changed?

Previous section shows that on Friedman2, methods output very similar results when K is set to its maximum value - allowing the model to always choose the best possible split at each node. This section studies how the rankings inferred by the same method change as K increases.

Table 4.8 shows the Jaccard measure of similarity measured between the top-2 features identified by each method for $K = 1$ and 4. Saabas and Shap seems to have only slightly changed as K increased while MDA is the method whose ranking evolved the most.

–	MDA_1 & MDA_4	MDI_1 & MDI_4	$SHAP_1$ & $SHAP_4$	$SAABAS_1$ & $SAABAS_4$
avg Jaccard Index	0.73	0.83	0.89	0.89

Table 4.8: **Average of the Jaccard Index:** The Jaccard Index is measured on the id of the two most important features detected by each method. The rankings of the same methods are compared for $K = 1$ and $K = 4$.

It seems, in this experience, that the methods that are the most influenced by the internal composition of the tree (split variable, depth of node, impurity reduction) - MDA & MDI - are also the ones that are the most affected by K. SHAP and Saabas depend indirectly on the tree structure as the predictive power of the tree is crucial in determining variable importance. By definition, MDA & MDI are more affected than SHAP & SAABAS if the position of a variable changes within a tree while only slightly affecting its prediction.

One can also see that, on Friedman2, SHAP and Saabas provide more "stable" top2-rankings when $K = 1$: Table 4.8 shows that the top-2 only slightly evolve as K increased and Table 4.5 shows that for $K = 4$ all rankings are extremely similar. So the rankings of MDA and MDI converge close to the rankings of SHAP & Saabas as K increases.

Comparison with Forest Global MDI

Modifying the parameter K leads to changes in splitting variables chosen at each node and as it modifies the results of local methods, global measures might also have been changed. This section compares the global MDI (G-MDI) ranking with local methods for the two different values of K. The global ranking is obtained through the *feature_importances* parameter of *Scikit-Learn*-based random forests.

The feature importances and the ranking of G-MDI is different within the two models M_1 and M_4:

1. $G\text{-}MDI_{M_1} = [0.13, 0.31, 0.42, 0.12] \quad \rightarrow ranking : [2, 1, 0, 3]$
2. $G\text{-}MDI_{M_4} = [0.012, 0.41, 0.55, 0.017] \rightarrow ranking : [3, 1, 0, 2]$

–	MDA_1 & $GMDI_1$	MDI_1 & $GMDI_1$	$SHAP_1$ & $GMDI_1$	$SAABAS_1$ & $GMDI_1$
avg Corr	0.62	0.67	0.80	0.81

–	MDA_4 & $GMDI_4$	MDI_4 & $GMDI_4$	$SHAP_4$ & $GMDI_4$	$SAABAS_4$ & $GMDI_4$
avg Corr	0.80	0.84	0.83	0.83

Table 4.9: **Average Spearman Correlation:** The Spearman correlation is computed between G-MDI and local methods and averaged over all samples. M_1 and M_4 are studied.

Table 4.9 displays the Spearman correlation between the different methods for $K = 1$, 4. The correlations between SHAP/Saabas and G-MDI remain similar for both values of K and is relatively high - around 0.8. The correlations between MDI / MDA and G-MDI increase with K and also reach 0.8 which makes sense as we have seen all local methods are very correlated when $K = 4$.

Note that the Jaccard Index computed with regard to G-MDI on the top-2 features equals 1 with MDI, SHAP and SAABAS and 0.96 with MDA when $K = 4$. The G-MDI importances of M_4 indicate that :

1. the two least important variables are much less important than the two others ;
2. they both contribute to a very small and similar decrease in impurity on the forest and are therefore poorly distinguishable within the rankings.

Knowing which feature ranks 3rd or 4th does not bring lots of information and might not be relevant, even locally.

4.1.4 Comparative example with Linear Regression

The local feature importance methods presented in this work either use the structure of the tree (Saabas, MDA and MDI) or use its predictive capacity (SHAP) in order to *understand its predictions*: it is all about model interpretability. The methods aim at extracting some of the complex knowledge that is held in trees - in the context of this work, identifying which variables are found to be important by the forest - through different means: decomposition of the prediction and decreases in impurity & accuracy. Previous sections have shown that the different methods provide either similar or different rankings depending on the dataset and forest's hyperparameters.

This section aims to compare the rankings derived by MDI, MDA, SHAP and Saabas with ground-truth rankings derived from a linear regression problem. Through this experience, one will observe the capacity of a random forest to use the variables similarly as the linear model and therefore rank the features closely to the local feature rankings inferred from the weights of the linear regression.

Results are studied for different values of the Forest hyperparameters in order to quantitatively track their influence on the structure of the Forest and observe how the feature rankings derived from the tree evolve.

Important: The feature importance methods explain the model. By definition, relevant variable should be important within the tree. The forest's feature importances first explain the model, one can't directly infers that these features are relevant for the outcome - they are only detected as important for the prediction of the tree.

Linear Regression parameters

The ground truth local importance of each feature x_j with linear regression weight w_j for a sample $\mathbf{x}$ corresponds to the absolute value of the *effect* of x_j weighted by the sum of absolute effects.

$$VIMP_{local_GT} = \frac{|w_j \times \mathbf{x}_j|}{\sum_i |w_i \times x_i|} \tag{4.1}$$

Parameters of the random forest

The forests trained for this section are made of 100 estimators and the *min_samples_split* parameter is set to 2. The differences between results obtained for forests of size 20, 100 and 500 trees are not significant and are not reported in this report.

The MDA method used has parameter *Split* set at True, *Uniform* set as False and feature importances are computed with the MSE loss function.

Example1 - Global ranking similar to Ground Truth

In this first example, the parameters of the Linear Regression are:

1. weights = $[3, 5, -7, -2, 6]$
2. b = 0
3. ranges = $[2, 2, 2, 2, 2]$
4. translation = $[0, 0, 0, 0, 0]$

All 5 features are completely uncorrelated and the random forests trained on the linear regression data globally rank the features in the same order as the absolute values of the linear regression weights. The ranking issued by Global-MDI is equivalent to the linear regression weights (global ranking of the linear problem).

Table 4.10 displays the correlation between local methods and the global feature ranking for different hyperparameters of the random forest. One can observe that the correlation of the local ranking with the model global ranking always increases as K increases and that the standard deviation tends to decrease. The correlation between MDA and the global ranking significantly increases as $K = 3$, the correlation increases by 0.3 and reaches 0.7. Limiting trees to a depth of 3 also increases the average correlation of the local ranking with the global one. A greater value of K has a big impact on the correlation of SHAP & Saabas on shallow trees. Their correlation with the global ranking increases from 0.45 to 0.82 and the standard deviation is reduced by a factor of 2. The MDA method is also highly influenced by the depth of the trees and its average correlation with the global ranking is high: 0.74 and 0.89 for both values of K. It is surprising to see that the local MDI ranking is nearly the same as the global ranking for nearly all samples. As local MDI is based on the decrease in impurity, it seems that no splits especially favor specific samples with regard to the impurity measure so that no feature emerges to be more important to predict a sample. One cannot conclude on the quality of the methods using Table 4.10 as the local methods are compared with the global one. However it is interesting to see that local rankings tend to get closer to the global ranking as some parameters are modified.

LR weights	MDI_n	MDA_n	SHAP_n	SAABAS_n	MDI_3	MDA_3	SHAP_3	SAABAS_3
avg $\text{Corr}_{K=1}$	0.98	0.40	0.52	0.52	0.96	0.74	0.45	0.45
avg $\text{Std}_{K=1}$	0.045	0.45	0.38	0.38	0.05	0.20	0.41	0.42
avg $\text{Corr}_{K=3}$	1.0	0.71	0.58	0.58	0.99	0.89	0.82	0.82
avg $\text{Std}_{K=3}$	0.0	0.24	0.35	0.34	0.01	0.11	0.20	0.20

Table 4.10: **Average & Std Spearman correlation coefficients:** The Spearman correlation between local methods' rankings and the global feature ranking is measured for all samples. The average correlation and the standard deviation are reported in the table. The global ranking corresponds to the weights of the linear regression problem. The subscript attached to the row elements corresponds to the depth of the trees ($'n'$ stands for no depth-limit). Results are displayed for $K = 1$ and 3.

Table 4.11 shows the correlation of the tree-based local methods with the ground-truth local ranking directly inferred from the linear regression problem. Previous table has shown that local MDI provides the same ranking for most samples - independently from the different tree parameters. It is therefore not surprising to observe that local MDI keeps nearly constant correlation scores among all models. For this example, MDI failed to provide sample-specific information/interpretation on the forest's predictions. Although the other methods are less correlated with the local feature, they are more expressive as their rankings are influenced by the structure of the forest and therefore, they each better explain the forest's predictions than MDI.

Quality of the ranking can be assessed on this table as local methods are compared with ground-truth data. MDA, SHAP and Saabas are all very influenced by the depth of the trees and the parameter K. All methods perform better i.e. they are more correlated to the ground-truth as the depth of the trees decreases and as the number of variables considered at each split increases. The MDA method performs generally better than SHAP and Saabas. The best results are obtained on the shallow-trees which suggest that:

1. as stated in the **Background** section, the accuracy performance of a model is not always linked to the capacity of the model to make local decisions based on the most relevant variables. We observe in Table 4.11 that the model better uses/detects relevant features when K and depth equal 3 but the most accurate model corresponds to the model built with $K = 3$ and no depth limit. One observes the propensity of a model to use relevant variables to make prediction by analysing local feature rankings - 4 of them being studied in this work.

2. the importance of a feature (decision) is influenced by its depth with regard to the root node. Results show that the model tends to better identify relevant variables at the early stages of all trees. By successively increasing the depth of the forest, I observed that the average correlation of local methods with the ground-truth ranking decreases.

Ground-Truth	MDI_n	MDA_n	SHAP_n	SAABAS_n	MDI_3	MDA_3	SHAP_3	SAABAS_3
avg $\text{Corr}_{K=1}$	0.503	0.066	0.139	0.144	0.49	0.34	0.10	0.10
avg $\text{Std}_{K=1}$	0.41	0.533	0.52	0.526	0.43	0.48	0.51	0.51
avg $\text{Corr}_{K=3}$	0.51	0.29	0.20	0.20	0.505	0.43	0.38	0.38
avg $\text{Std}_{K=3}$	0.41	0.49	0.51	0.51	0.405	0.47	0.48	0.48

Table 4.11: **Average & Std Spearman correlation coefficients:** The Spearman correlation between local methods' rankings and the local feature ranking inferred from the linear problem is measured for all samples. The average correlation and the standard deviation are reported in the table. The subscript attached to the row elements corresponds to the depth of the trees ($'n'$ stands for no depth-limit). Results are displayed for $K = 1$ and 3.

Example2 - Global Ranking different than Ground Truth

In this second example, the parameters of the linear regression problem are slightly modified such that the random forests learned on the linear regression data globaly rank the feature in a different order than the linear regression weights. In this second example, forests fail to identify the global hierarchical relationship between variables. The impact on the local rankings are observed as well as the capacity of the forests to locally identify the relevant variables. Global-MDI switches the position of the fourth and first features and more importantly of the second and third features. The feature with weight of -7 is considered to be globally less relevant than the x_2 with value 5. The parameters of the linear problems are the following:

1. weights $= [3, 5, -7, -2, 6]$
2. b $= 0$
3. ranges $= [1, 5, 2, 3, 2]$
4. translation $= [0, 0, 0, 0, 0]$

Table 4.12 shows in 3 tables the average correlation between the local methods and linear regression weights, the ground-truth rankings and the Global-MDI where the local rankings are derived from 4 different forests. One observes again that the local-MDI rankings are independent from the variations in the trees' structure: indeed, varying the depth or the K does not change the extent to what MDI is correlated with the other rankings. SHAP, Saabas and MDA are all three very influenced by the depth and by K. The biggest correlation with the ground truth data is obtained on shallow trees and $K = 3$. MDA and MDI perform well on this dataset and their correlation with the ground truth feature ranking respectively reached 0.64 and 0.66.

The tables show that MDI is extremely correlated to its global version. The correlation between Global-MDI and other local methods increases with K and inversely proportionally to the depth. The model that locally uses the best relevant features (i.e. that correlates most with ground-truth rankings) is also the model whose local rankings are very similar to the global ranking. Finally, it is also the model that correlates most with the linear regression weights. We also observe that the correlations of the local methods with the linear regression are similar to the values of the correlation between local methods and ground-truth samples.

LR-weights	MDI_n	MDA_n	$SHAP_n$	$SAABAS_n$	MDI_3	MDA_3	$SHAP_3$	$SAABAS_3$
avg $Corr_{K=1}$	0.60	0.29	0.42	0.41	0.59	0.48	0.35	0.35
avg $Std_{K=1}$	0.05	0.43	0.38	0.36	0.04	0.23	0.37	0.37
avg $Corr_{K=3}$	0.60	0.54	0.46	0.46	0.6	0.57	0.57	0.58
avg $Std_{K=3}$	0.003	0.23	0.30	0.31	0.012	0.04	0.17	0.16

GT-rankings	MDI_n	MDA_n	$SHAP_n$	$SAABAS_n$	MDI_3	MDA_3	$SHAP_3$	$SAABAS_3$
avg $Corr_{K=1}$	0.64	0.20	0.34	0.34	0.66	0.57	0.32	0.34
avg $Std_{K=1}$	0.33	0.47	0.44	0.45	0.33	0.37	0.44	0.44
avg $Corr_{K=3}$	0.66	0.49	0.40	0.39	0.66	0.64	0.57	0.57
avg $Std_{K=3}$	0.32	0.40	0.42	0.42	0.32	0.34	0.37	0.36

Global-MDI	MDI_n	MDA_n	$SHAP_n$	$SAABAS_n$	MDI_3	MDA_3	$SHAP_3$	$SAABAS_3$
avg $Corr_{K=1}$	0.96	0.50	0.66	0.65	0.98	0.87	0.63	0.64
avg $Std_{K=1}$	0.053	0.40	0.32	0.33	0.04	0.13	0.35	0.34
avg $Corr_{K=3}$	0.99	0.82	0.72	0.72	0.99	0.97	0.89	0.90
avg $Std_{K=3}$	0.003	0.18	0.28	0.28	0.01	0.04	0.15	0.14

Table 4.12: **Average & Std Spearman correlation coefficients:** The Spearman correlations between the rankings of the local methods and the linear regression weights, the ground-truth local rankings and the Global-MDI ranking inferred from the forest are measured for all samples. The average correlation and the standard deviation are reported in the table. The subscript attached to the row elements corresponds to the depth of the trees ($'n'$ stands for no depth-limit). Results are displayed for $K = 1$ and 3.

The average correlation between the linear regression weights and the ground-truth rankings corresponds to 0.40. The average correlation between the ground-truth ranking and the forest global-MDI equals to 0.66. Finally, the correlation between both global rankings corresponds to 0.6. These values correspond to the forest trained with $K = 3$ and a maximum depth of 3. It seems from these statistics that the local ground-truth rankings are on average closer to the forest global ranking than the linear regression weights. The fact that many samples have a different ground-truth ranking than the regression weights might partly explain why Global-MDI (an aggregate version of the local method) identifies a global ranking which does not correspond to the linear regression weights but to a local ranking that predominates in the training samples.

Conclusion

Due to their simplicity, it is very easy to interpret the predictions of linear models. The weights of the regression and the samples values are sufficient. The artificial problem described in this section allows to derive ground-truth feature rankings and the data drawn from uniform distributions is used to train a forest that should model the linear regression function in the range of the input data.

Since the forest should model the linear regression, one can freely compare both methods.

Firstly, it is interesting to see how rankings issued by local methods evolve as the parameters of the forest change. Table 4.12 and Table 4.10 show that local ranking become more similar to the global ranking as K increases and the depth decreases.

Secondly, the extent to which forests are able to use features in a similar way as the linear regression function depends on the forest hyperparameters which is observed in the *GT-rankings* section of Table 4.12. For $K = 1$, the correlation between ground-truth ranking and rankings issued by SHAP, Saabas and MD is relatively small, which means that the variables that are important for the forest are in fact poorly relevant for the output.

The forest that relied the best on relevant features ($K = 3$ and depth= 3) is also one of the less accurate estimator as the forest developed without depth limit provide the best results. Therefore, the most accurate forests don't necessarily identify the best relevant features.

4.1.5 Ranking Expressivity

This section studies the values of the importance associated to each feature and observes how different the importance associated to the most important feature is from the value associated to the least important feature. For each sample the difference between the maximum and minimum value is reported. The *l1*-norm is used to regularize the rankings so that the importance of all features sums to 1.

The expressivity of all methods are shown for a variation for $K = 1$ and 2 and for a tree of maximum depth set to 3. The graphs are in the Appendix A. One can observe that the expressivity of all methods are very similar despite tuning parameters. These parameters influence the correlation between the rankings but we observe that the importance difference between the most important and least important variables is not affected. This was observed on other datasets such as *Friedman1*.

4.1.6 Performances

The MDA method is fully implemented in python with loops and therefore doesn't benefit from a fair comparison compared to the other methods. Saabas, SHAP and the MDI methods are partly implemented in python. The computation time is measured in seconds for forests of 100 and 500 either fully expanded ($\not{d}$) or with a maximum depth set at: $d = 3$. Results are obtained on a 11th Gen Intel(R) Core(TM) i7-1165G7 processor.

Computation time in seconds	100 Trees, ~~d~~	500 Trees, ~~d~~	100 Trees, $d = 3$	500 Trees, $d = 3$
MDA	0.0039	0.0119	0.0029	0.0079
MDI	4.44	22.70	2.0321	10.19
$SHAP$	0.382	1.81	0.0179	0.092
$SAABAS$	0.842	4.38	0.009	0.0479

Table 4.13: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples. The table shows the results for trees built with $K = 1$ and 3 and with no limit on the depth of the trees - using the *Iris* dataset.

The computation-complexity of all methods scales linearly with the number of trees which is expected as trees are studied independently from each other. The local MDI method is by far the simplest and the most efficient.

4.2 Classification Task

This section studies the rankings provided by the local methods: MDA, MDI, SHAP & Saabas with regard to different values of the forest's hyperparameters. The absolute values of all feature importances are considered in order to compare rankings. Rankings are then normalized with the *l1*-norm to study their expressivity.

4.2.1 Study of the Spearman correlation coeficient on the Iris dataset

Table 4.14 provides the average Spearman correlation coefficients between all local methods for a forest built with no constrain on its trees' depth and $K = 1$ and 3 on the *Iris* dataset.

avg Corr	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$	$MDA_{K=3}$	$MDI_{K=3}$	$SHAP_{K=3}$
MDI	0.80	–	–	0.85	–	–
$SHAP$	0.86	0.84	–	0.886	0.84	–
$SAABAS$	0.81	0.85	0.91	0.90	0.87	0.9

Table 4.14: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples. The table shows the results for trees built with $K = 1$ and 3 and with no limit on the depth of the trees - using the *Iris* dataset.

Table 4.15 provides the average Spearman correlation coefficients between all local methods for a forest made of trees of maximum depth 3 and $K = 1$ and 3 on the *Iris* dataset. The trees of the models trained in Table 4.14 had a maximum depth of 9.

avg Corr	$MDA_{K=1}$	$MDI_{K=1}$	$SHAP_{K=1}$	$MDA_{K=3}$	$MDI_{K=3}$	$SHAP_{K=3}$
MDI	0.69	–	–	0.83	–	–
$SHAP$	0.78	0.92	–	0.91	0.92	–
$SAABAS$	0.73	0.90	0.94	0.94	0.92	0.96

Table 4.15: **Average Spearman correlation:** The correlation between the feature rankings is computed for each sample and averaged over all samples. The table shows the results for trees built with $K = 1$ and 3 and trees are expanded till depth 3 - using the *Iris* dataset.

All models presented in Table 4.14 and Table 4.15 contain 500 trees and trained on the *Iris* dataset. The first important observation is that the local rankings are very correlated among all forests. Increasing the value of K for unconstrained-depth trees shows very little impact on the correlation between ranking on the *Iris* dataset. The same observation applies to the SHAP, Saabas and MDI methods on the forests containing trees of depth 3: increasing K has a marginal effect on the correlation. However, it seems that the rankings obtained from the MDA method are more sensible to the depth of trees. Indeed, for $K = 1$ in Table 4.15, MDA has lost around 0.10 point in correlation with regard to the other methods compared to the fully expanded forest. Increasing K to 3 restore a high correlation between MDA and the other methods.

Results show that all methods tend to interpret the prediction of the tree in a similar way. Indeed, a sample's feature ranking is nearly the same among all methods. One cannot conclude that the methods successfully retrieved the most relevant variables but it is equally interesting that all methods identify the same - or at least a very similar - interpretation to the model's local predictions.

Ranking Expressivity

This section studies the values of the importance associated to each feature and observes how different the importance associated to the most important feature is from the value associated to the least important feature. For each sample the difference between the maximum and minimum value is reported. The *l1*-norm is used to regularize the rankings so that the importance of all features sums to 1.

Figure 4.1: **Ranking Expressivity:** Expressivity measured among the 30 rankings of the test set. The rankings are derived from the random forest containing trees unlimited-depth trees and trained with $K = 3$ on the *Iris* dataset.

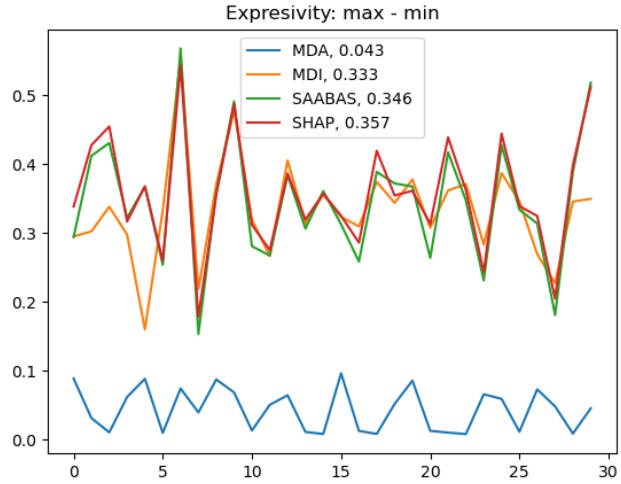

Figure 4.2: **Ranking Expressivity:** Expressivity measured among the 30 rankings of the test set. The rankings are derived from the random forest containing trees limited to depth 3 and trained with $K = 1$ on the *Iris* dataset.

Figure 4.1 and Figure 4.2 respectively show the expressivity of each method a forest trained with $K = 3$ and no limit on trees' depth and a forest trained with $K = 1$ and trees' depth limited to 3. One can see that MDI, SHAP and Saabas are very similar and tend to discriminate the most important feature from the least important feature with nearly the same importance gap among all test samples. Additionally, the rankings of these methods are only slightly influenced by the forest's hyperparameters. However, as seen in the previous section, the rankings derived by MDA are much more influenced by the tree parameters that are tweaked in this experiment. Once can see that in Figure 4.2, MDA harldy distinguishes any particular hierarchy among the features while MDI, SHAP and Saabas already figured out a strong hierarchy. An average gap of 0.33 is already high as the *l1*-normalization makes importances sum to 1.

In addition to providing similar rankings (MDI, SHAP and Saabas are very correlated), the methods also provide similar importance values. Note that, despite MDA does not reflect a strong hierarchy in the feature importance values, it remains - surprisingly - highly correlated to all methods as shown in the left part of Table 4.15 with an average correlation of 0.75.

As usual, the results from the Saabas and SHAP methods are very similar. Both methods decompose the prediction to assign a contribution, i.e. importance score, to each feature. One could expect major differences between the two methods as SHAP is model agnositic and as Saabas directly uses the structure of the tree to assign feature importances. This major difference seems to have little impact on the rankings provided by the two methods as they have always been highly correlated in all experiences realised in this work.

Chapter 5

Application to Gene Network Inference

This section of the report aims to compare local ranking methods applied to a tangible application from the biomedical study domain. The motivations for this section are manifold: the methods described in the Background and the second chapter are here studied in a practical context, their results are compared to ground truth data and the performances of the methods are contrasted with state of the art algorithms from the field of study.

The attention is high on the benefits machine learning techniques can bring to medicine as models are being used for both the purpose of improving diagnosis quality - e.g. through medical imaging - and for the purpose of better understanding the human body and grasp information about the numerous interactions that occur in an organism.

This section is about inferring the regulatory interactions that exist between genes of the same cell. Such study is made possible through the availability of large datasets of genes expression obtained by advances in genomics in the field of DNA microarrays.

5.1 Gene Network Inference Problem

5.1.1 Gene Regulatory Network

The concept of Gene Regulatory Network (GRN) has grown popular over the last decade as an approach for describing transcriptional interactions. The transcriptional regulation is the control mechanism by which cells respond to intra- and extracellular signal by monitoring the retranscription rate of several genes. Observing transcriptional regulations and thus changes in *gene expression levels* summarizes in studying the genes' orchestration among a cell.

GRN can be global and therefore be inferred from a population of cells. However, since single-cell expression become more available, cell-specific GRN can be inferred from biological techniques which open the door to local studies, here: using local inference methods to approximate a cell-specific regulatory network.

Gene Regulatory Network inference consists in building a network - seen as a graph - in which nodes are genes and the weight of edges indicates the level of regulatory interaction between them. Note that the interaction between two genes in a GRN does not necessarily imply physical interaction but can also refer to indirect regulation via proteins. Previous works have shown that GRN are relatively sparse[20]: many poorly connected nodes and few highly connected ones. In the network inference problem, the ground truth data is usually considered as boolean: 0 denoting the absence of regulation between two genes and 1 representing a regulatory interaction. The Boolean problem of inferring interactions between genes is thus a highly class-imbalanced problem as the number of possible interactions largely exceeds the number of connected nodes in the true network.

The quality of GRN inference models is crucial to understand the development, functioning and pathology of organisms studied at cellular level.

5.1.2 Deriving gene regulation through genes' expression

As seen in the previous section, cell-specific Gene Regulatory Network explains the interactions between genes in a single cell and therefore the expression level of the genes. GRN are typically obtained by reverse-engineering i.e. GRN are inferred from the computational analysis of the genes' expressions. These algorithms rely on the assumption that the existence of a strong statistical relationship between genes' expression is an indication of a potential functional relationship (regulatory interaction) [25].

Algorithms inferring genes interactions usually compute a score for each pair of gene. The first[15] methods used the Pearson correlation coefficient on the expression of genes but they failed to capture non-linear dependencies between genes. In this report, ensembles of regression trees are used to capture relationships from the expression of genes. The tree-based method can possibly capture higher-order conditional dependencies than the Pearson correlation coefficient as it makes less assumptions on the input data.

5.2 Background - Dyngen

The paper *"dyngen: a multi-modal simulator for spearheading new single-cell omics analyses"*, released in June 2020 is the main reference of this section. The authors developed a tool to generate databases of single cells expressions through a simulator from which they infer *ground truth* measurements.

This paper brought a significant contribution to the field as the ability to generate datasets allow to perform better quantitative assessment of the quality of singe-cell analysis that was often jeopardized by the lack of data. Indeed, in addition to providing models' input data, the dyngen simulator can be used to benchmark single-cells analysis such as cell-specific network inference, trajectory alignment methods and RNA velocity predictions.

5.2.1 Generation of Genes' expressions

The multi-modality simulator of single-cell expression described in the paper uses an improved version of Gillespie's stochastic simulator algorithm [4]. Dyngen can simulate 14 different types of dynamic processes using different experimental conditions for each. The simulator described in the manuscript is able to generate 42 different datasets of single-cell gene expressions.

The dataset generation starts by the definition of a global gene regulatory network i.e. a set of rules between genes that is not cell-specific. The different types of simulated cells develop over time according to this GRN and additional experimental conditions. The design of this GRN has a crucial impact on the cellular development processes of the simulation.

Figure 5.1 graphically showcase the functionality of dyngen. The genes' expressions being inferred at the end of the simulation. A dataset of genes' expressions consists of N rows (cells) and F features (expression value of each gene for each cell) and with each column of the database representing the expression of a gene.

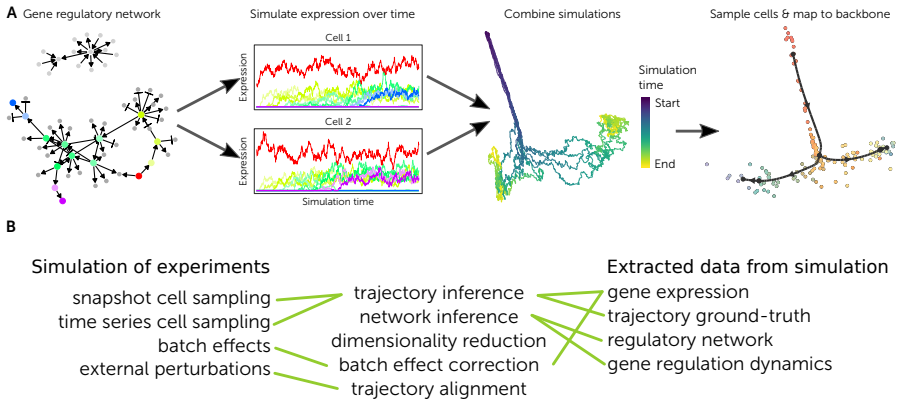

Figure 5.1: **Showcase of Dyngen functionality[4]. A:** The typical process of generating a dataset with Dyngen. **B:** Evaluating different types of computational tools requires simulating different types of experiments and extracting different layers of information from the simulation.

5.2.2 Cell-specific Gene Regulatory Network

The authors defined a method to determine the cell-specific ground-truth regulatory network. The regulatory effect of a regulator gene R on a target gene T is measured by observing the contribution of R in the propensity transcription of T i.e. observing by how much the transcription[1] (expression level) of T is affected by the expression level of R. Indeed: the regulatory effect of R on T is defined as the change in propensity transcription of T when the expression of R is set to 0. The interactions are computed among all pairs of the regulator and target sets of gene to build a cell-specific GRN.

The regulatory effect of gene R on gene T at a particular state S is the difference of propensity transcription weighted by a factor wpr_T corresponding to the pre-mRNA production rate of T.

$$regEffect_T = \frac{propTrans_T(S) - propTrans_T(S[y_R \leftarrow 0])}{wpr_T} \tag{5.1}$$

The regulatory effect lies between $[-1, 1]$, the extremums represent complete inhibition or maximal activation and 0 is synonym of an inactive regulatory interaction in the $\{R, T\}$ pair. Note that the simulator studies the trajectories of cells over time, the propensity transcription rate is a metric of a dynamic problem while the expression of a gene is a static measure made at each step.

5.2.3 Methods

The ground-truth data extracted from the simulator is used to further compare the performance of cell-specific network inference (CSNI) methods: LIONESS, SSN and pySCENIC. These methods are shortly described in the following section as their results are compared to local feature ranking methods in the *Experiments* section.

LIONESS + Pearson

LIONESS[11] stands for Linear Interpolation to Obtain Network Estimates for Single Samples. It consists in a methodology that can be used with different (gene regulatory) network construction algorithms i.e. methods that output an adjacent matrix whose values correspond to the weight between two edges of the network.
LIONESS is based on the ideas that:

1. Each cell/sample has its own regulatory network
2. Each edge of the global regulatory network is a linear combination of that edge across all individual samples' network. The global network is seen as an aggregated network

[1] Transcription corresponds to the replication of the RNA copy of a gene's sequence i.e. it refers to the quantity of some proteins within a cell - US National Human Genome Research Institute

3. It is then possible to measure the contribution of a single sample in the aggregate network

Through a network construction algorithm, LIONESS builds a network on the entire cell population and a second network on all samples but the target cell. The method compares both networks and uses a linear equation to estimate the network of the selected sample. The value of an edge between nodes (genes) i and j in the target cell q is defined by:

$$e_{ij}^{(q)} = N(e_{ij}^{(\alpha)} - e_{ij}^{(\alpha-q)}) + e_{ij}^{(\alpha-q)} \tag{5.2}$$

With $e_{ij}^{(\alpha)}$ being the value of the edge in the global network and $e_{ij}^{(\alpha-q)}$ the value of the edge in the network that withheld q.

This algorithm is repeated for all edges and all samples to build cell-specific networks.

In this implementation of LIONESS, networks are generated using the Pearson correlation formula applied on the genes expression data (correlation is measured between all pairs of genes).

SSN

The methodology of the Sample Specific Network[12] (SSN) is very close to LIONESS as the SSN of each sample (cell) is constructed through a statistical perturbation analysis of this sample against the sample population. The Sample Specific Network method also uses Pearson correlation coefficients (PCC) to build the regulatory network.

Three networks are constructed by PCC:

1. A network that excludes the target cell, the *reference network* : PCC_n
2. A network that includes the target cell, the *perturbed network* : PCC_{n+1}
3. The differential network is computed on every edges. ΔPCC is a first sketch of the *sample's SSN*. The SSN of a sample is based on the contrast that this sample brings to the expression of genes.
   Differential network: $\Delta PCC = PCC_{n+1} - PCC_n$[12]

The PCC values of some edges of the perturbed network show significant discrepancies with the reference network if they are evident differences between the expression patterns of the target sample and the reference samples. ΔPCC would then be significant and should be kept in the SSN of the target sample.

The SSN method then filters the ΔPCC and only keeps the edges with significant differential correlations. The statistical significance of a differential edge is established by comparing the Z-score of the ΔPCC to a threshold $P-value$ ($P = 0.05$). The Z-test has the null hypothesis that a distribution can be approximated by a normal distribution. The authors[12] of the method have

shown that the ΔPCC's of a network follow a symmetrical distribution (volcano distribution) whose tails regions are similar to those of a normal distribution and therefore confirms the relevance of using the Z-test.

$$Z = \frac{\Delta PCC - \mu_{\Delta PCC}}{\rho_{\Delta PCC}} \tag{5.3}$$

The resulting SSN corresponds to the GRN of the target sample (cell).

pySCENIC

PySCENIC is the python implementation of the SCENIC method[23]. The SCENIC method consists in a workflow of several steps[4]:

1. Step1: A global Gene Regulatory Network is inferred from the gene expression data using tree-based methods similar to GENIE3[10]. GENIE3 consists in training an ensemble of tree on the gene expression data and inferring regulators from the feature ranking of the ensemble. Because of genes' co-expression, the regulatory network contains many false positives and indirect regulations such as downstream effect.

2. Step2: The 10 main regulators of each target are selected and interactions are grouped in modules - each module containing a regulator and all its targets i.e. all genes for which the regulator is important. The main regulators are selected by the RcisTarget method which aims to detect direct regulatory links inside a GRN.

3. Step3: Using AUCell, an activity score for each module is determined in each cell. This step provides a matrix with a score for each module in each cell. It is then possible to filter modules through a module-specific threshold value that decides whether or not the regulatory interactions represented by the module exist within a cell. This last step provides a cell-specific gene regulatory network.

5.3 Experiments

5.3.1 Methodology

The purpose of this section is to compare the cell-specific gene regulatory networks derived from Random Forest using local feature ranking methods to the 3 methods described above: SSN, LIONESS + Pearson and SCENIC. A model is built for each target gene and each model is trained on the expression data of the regulatory genes - *regulators*. Random Forest are trained on genes expression with the assumption that complex relations at the cellular level between some regulators and the target gene can be captured by the model and that these relationships can be retrieved by drawing the ranking of the genes locally i.e. for each cell.

In this section, one observes which genes are important to predict a target gene with the assumption that genes that are measured to be important (rank high in feature importance) are also the genes that regulate the target gene. Using a global ranking method on the random forest provides a global GRN (GENIE3 and SCENIC) i.e. a summary (aggregation) of the genes' interactions among all cells of the input dataset. However, cells can be of different types and result from different trajectories and thus, pairwise genes interactions are different for each cell and a cell-wise feature ranking method might succeed at reproducing these specific interactions.

Comparing pySCENIC with local ranking methods results in comparing two kinds of methods: one that directly provides cell-specific GRN's and the other that infers a cell-specific network from the analysis of a global feature ranking.

In order to compare methods, 9 datasets were generated using the dyngen simulator. Each simulation computes the ground truth network and provides specific prior information such as the regulator (R) and target (T) sets of genes, usually: $R \subset T$. These priors are used by all methods and the Random Forest are trained on the set of regulators from which the target gene is excluded.

Each dataset describes the genes' expression over 1000 cells (samples). The results of applying any local ranking method on a model_i is a matrix of shape: $1000 \times size(R)$, that provides for each cell and target gene_i the feature ranking of the regulatory genes. Each feature importance corresponds to a confidence score of the interaction of a regulatory gene and the gene_i in a specific cell.

5.3.2 Scoring Method

A feature importance value is obtained for each $\{R, T\}$ gene pair of each cell. The ground truth data is a binary vector with $0's$ being the absence of regulation and $1's$ being the presence of a regulatory interaction. The regulatory network inference problem is strongly class-imbalanced as the ground truth data is a sparse vector. Indeed, very few direct interactions are found to exist withing the genes of a cell. The local methods provide $O(T.R)$ feature importance values for each cell, among all datasets: $O(T) = O(R) = O(100)$, while the ground-truth data lists an average of $O(100)$ regulatory interactions[2] per cell. The large number of false positive i.e. wrong regulatory interactions, is a weakness of all methods. Note that pySCENIC filters its modules and therefore reduces its propensity to detect false links.

In order to prevent the analysis to depend on the choice of a threshold applied on feature importance values, the local ranking methods are compared using the same methodology as most network inference tools. The AUROC and AUPR values are computed for each cell. The PR and ROC curves are drawn from all $\{R, T\}$ pairs and the area under the curve is used as a comparison metric. The metric is averaged among all cells of the dataset and the reported

[2]This information was computed among the 9 datasets. Note that $O(R) \approx O(T)$

metrics are the meanAUPR and meanAUROC. The propensity of the methods to contain false positive links is (partially)[3] monitored by the meanAUPR score and the propensity of the methods to detect most of the correct interactions is (partially) monitored by the meanAUROC.

The cell-wise GRN computed by Dyngen sometimes gives a regulatory interaction within the same gene i.e. a non-null cyclic edge. Genes are directly regulated by proteins and RNA sequences that are replicated from or stimulated by themselves or other genes. However, none of method studied in this section is able to detect self-regulatory interactions.

5.3.3 Methods Comparison

Tree parameters

Each Random Forest consists of 40 trees, $R - 4$ variables are randomly sampled among the R regulators at each split and trees are expanded till a minimum of 5 samples per leaf. The local methods (MDA, MDI, SHAP and SAABAS) are then used to infer a feature ranking for each cell and each target gene.

Remark on AUPR & AUROC:

The background section introduces the AUPR and AUROC metrics and describes the metric values for a model with little or no distinguishing power between predictions: class-wise (AUROC) and regarding to positive and false positive (AUPR). A random classifier has a AUROC score close to 0.5 and AUPR score corresponding to the average precision of a random classifier: $\frac{P}{P+N}$ (proportion of positive in the dataset). Over the 9 datasets, the proportion of positive examples approximately corresponds to 0.02.

Results 1 - Graph visualisation

Figure 5.2 exhibits the scores of the methods that the reference paper used to infer regulatory networks. The python implementation of the SCENIC method provides better results than the two Pearson-Correlation-based algorithms with regard to both the AUROC and AUPR metrics. Note that on these datasets, the SSN performs very poorly with meanAUROC close to 0.5 and therefore exhibits very little distinguishing power between predictions. The LIONESS-Pearson algorithm ranks intermediate and shows better scores than SSN while its implementation only slightly differs from SSN. Finally, SSN's meanAUPR scores are close to 0.02.

Figure 5.3 presents the scores of the local feature ranking methods whose weights are used to infer the regulatory interactions within cells.

[3]"Partially" because both the AUROC and AUPR scores have a more complex definition - please refer to the Background section

SHAP and SAABAS show, as their similar implementation induces, alike results. The most interesting result comes from the observation that local methods clearly outperform pySCENIC with regard to both metrics. One could think that by filtering regulatory modules, pySCENIC could maintain a better AUPR score than the other methods that do not implement any filter. However, results show that one can better set a threshold on the predictions of the local ranking methods to distinguish true positives from false positives (AUPR[4]). All three based methods show better AUPR score, the reason might be inherent to the tree structure that is directly used to infer cell-specific ranking in the local methods. The number of feature considered at each split being close to the number of regulators, many features can be skipped at the level of the ensemble and as they are not used in the tree, no regulatory interaction can be inferred from them. This reduces the number of false positive edges and seems not to affect the amount of true interactions detected by the method as the AUROC score of MDA and MDI is generally higher than pySCENIC.

Note that pySCENIC is also based on ensemble of trees but the hyperparameters of the random forest being used are not shared by the authors of Dyngen. A second important difference is that pySCENIC uses the global feature ranking to build a global GRN from which local networks are infered.

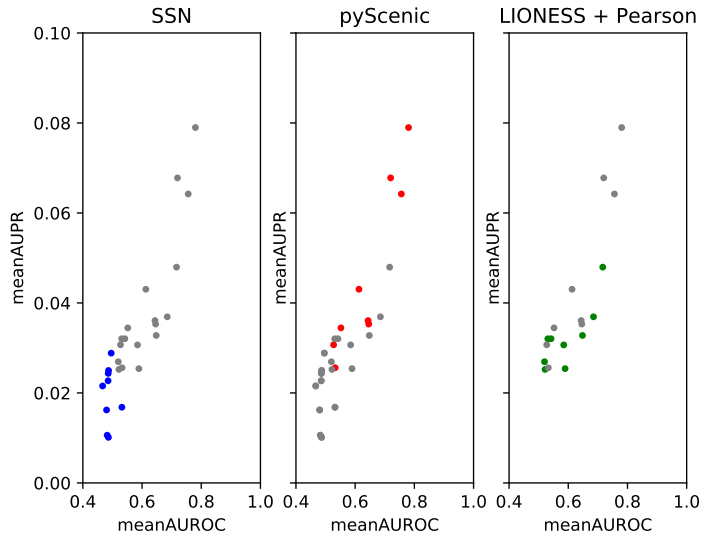

Figure 5.2: **Showcase of cell-specific GRN inference tools on 9 test datasets:** The meanAUPR and meanAUROC scores for each dataset are reported.

[4]AUPR shows how effectively one could set a threshold on feature values (predictions) to distinguish false positive from real positives

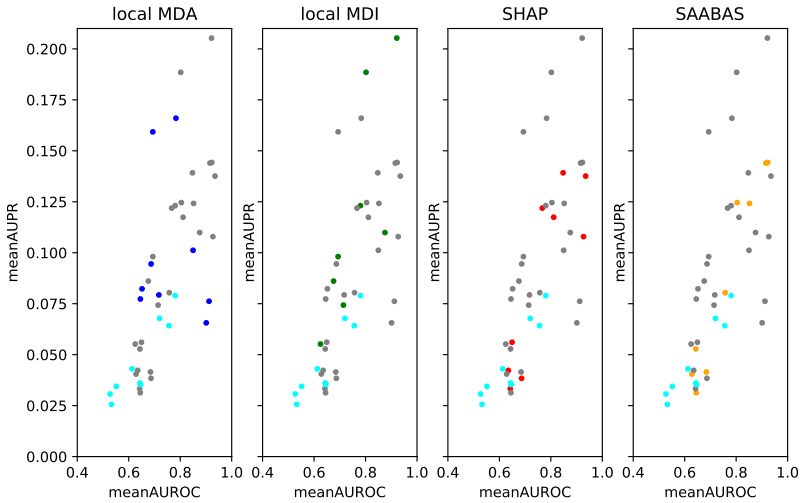

Figure 5.3: **Showcase of local feature ranking methods on 9 test datasets:** The meanAUPR and meanAUROC scores for each dataset are reported. The cyan dots correspond to the scores of pySCENIC and the grey dots is the mask of the other local methods

Results 2 - In depth tabular visualisation

Table5.1 and Table5.2 show the meanAUROC and meanAUPR score of all methods on each dataset and Table5.3 and Table5.4 show the standard deviation of each score for all methods and datasets.

meanAUROC: The local MDA method provides the best AUROC score on 2 datasets and MDI & SHAP score better than other methods on 3 datasets each. pySCENIC only performs a little better than all methods on the 7th dataset. The highest mean scores and the smallest standard deviation of the scores are highlighted in blue for each dataset.

In addition to providing very similar results, the AUROC scores of each method are similarly spread with regard to their mean. As one can see on Table 5.3, the standard deviation values are similar among all methods for each dataset.

meanAUPR: Table 5.2 shows that MDA scores best on 3 datasets, MDI on 4 datasets and SHAP & SAABAS on one dataset each. The tree-based local methods have significantly better scores than SSN, LIONESS and pySCENIC. The meanAUPR scores of the local methods seems to be relatively close to each other for each dataset. The main difference occurs between MDI and MDA on

the 5th and 6th datasets where MDI reaches scores close to 0.2 and MDA has scores of around 0.07.
As no method provides significantly better scores on all datasets, it could be interesting to study juxtapositions of local feature rankings and try optimizing the AUPR score while maintaining the high AUROC scores local methods provide.

meanAUROC score	SSN	LIONESS	pySCENIC	MDA	MDI	SHAP	SAABAS
Dataset 1	0.5319	0.6479	0.7560	0.8498	0.8755	0.8475	0.8519
Dataset 2	0.4957	0.5309	0.5273	0.6451	0.625	0.6351	0.6283
Dataset 3	0.4852	0.5216	0.5325	0.6868	0.714	0.6864	0.6842
Dataset 4	0.4857	0.5197	0.5519	0.6516	0.6759	0.6423	0.6448
Dataset 5	0.4822	0.7165	0.6436	0.901	0.9215	0.9269	0.9155
Dataset 6	0.4864	0.6852	0.72	0.912	0.9345	0.9352	0.9229
Dataset 7	0.4801	0.5892	0.7803	0.717	0.7764	0.7667	0.7572
Dataset 8	0.4868	0.5418	0.6128	0.7834	0.8019	0.811	0.8042
Dataset 9	0.4667	0.5846	0.6458	0.6934	0.6863	0.6493	0.6433

Table 5.1: AUROC scores tabular summary. The reported scores correspond to the average over all cells of the AUROC precision score. The method with the highest score is highlighted in blue for each dataset.

meanAUPR score	SSN	LIONESS	pySCENIC	MDA	MDI	SHAP	SAABAS
Dataset 1	0.0168	0.0327	0.0642	0.1012	0.1132	0.124	0.1506
Dataset 2	0.0288	0.032	0.0307	0.0773	0.055	0.0579	0.0.0684
Dataset 3	0.0226	0.025	0.0255	0.0945	0.0739	0.0679	0.0774
Dataset 4	0.0243	0.0269	0.0344	0.0823	0.0826	0.0618	0.0629
Dataset 5	0.0106	0.0479	0.0361	0.0656	0.1987	0.1199	0.1341
Dataset 6	0.0101	0.037	0.0678	0.0762	0.2086	0.1412	0.1533
Dataset 7	0.0162	0.0254	0.0789	0.0793	0.1195	0.1571	0.1186
Dataset 8	0.0245	0.032	0.0430	0.166	0.1805	0.1433	0.1752
Dataset 9	0.0215	0.0306	0.0353	0.1595	0.0927	0.0633	0.077

Table 5.2: AUPR scores tabular summary. The reported scores correspond to the average over all cells of the AUPR precision score. The method with the highest score is highlighted in blue for each dataset.

std AUROC score	MDA	MDI	SHAP	SAABAS
Dataset 1	0.0552	0.0487	0.0423	0.0517
Dataset 2	0.0348	0.0331	0.032	0.035
Dataset 3	0.0312	0.0252	0.0335	0.0317
Dataset 4	0.0341	0.0296	0.0342	0.0361
Dataset 5	0.0185	0.0144	0.008	0.0135
Dataset 6	0.012	0.0104	0.0066	0.009
Dataset 7	0.0437	0.026	0.0293	0.0271
Dataset 8	0.0297	0.0314	0.0312	0.0312
Dataset 9	0.0419	0.0701	0.0463	0.041

Table 5.3: std AUROC scores tabular summary. The reported values correspond to the standard deviation of the AUROC values among all cells. The methods that provide AUROC scores that spread the least with regard to the mean are highlighted in blue.

std AUPR score	MDA	MDI	SHAP	SAABAS
Dataset 1	0.0544	0.052	0.0506	0.0546
Dataset 2	0.0749	0.0463	0.0507	0.0674
Dataset 3	0.1086	0.0609	0.0672	0.0744
Dataset 4	0.0975	0.066	0.0712	0.0737
Dataset 5	0.0307	0.0436	0.0629	0.0681
Dataset 6	0.0251	0.0376	0.0422	0.009
Dataset 7	0.0677	0.039	0.0647	0.0505
Dataset 8	0.0681	0.0659	0.0773	0.0679
Dataset 9	0.1173	0.0563	0.054	0.071

Table 5.4: std AUPR scores tabular summary. The reported values correspond to the standard deviation of the AUPR values among all cells. The methods that provide AUPR scores that spread the least with regard to the mean are highlighted in blue.

Figure 5.4 show a box plot comparison of the meanAUPR scores of the local methods and pySCENIC. MDA is the local method that has the lowest median AUPR score and MDI has the highest median score. It is again clear that local methods perform better that pySCENIC and share similar performances.
Box plots of the meanAUPR scores on all datasets shows that MDI is the technique that usually provides higher AUPR scores and 50% of MDA's scores are included between 0.075 and 0.1 on Figure5.5.

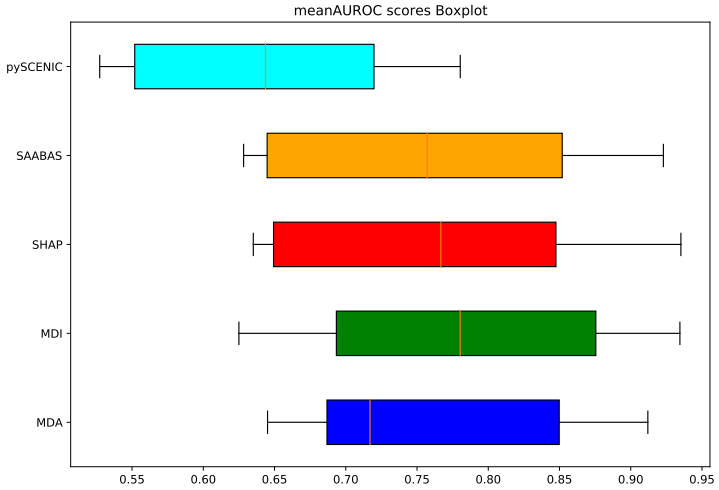

Figure 5.4: **meanAUROC - Boxplots comparison of local methods and pySCENIC.** The vertical line corresponds to the median and the box extends from the first to the third quartile

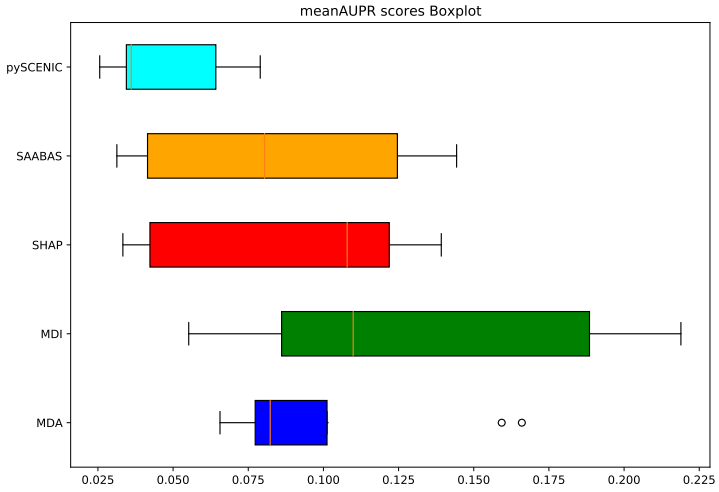

Figure 5.5: **meanAUPR - Boxplots comparison of local methods and pySCENIC.** The vertical line corresponds to the median and the box extends from the first to the third quartile

5.3.4 Gene Interactions Example on cell id

Figure 5.6 to Figure 5.10 show the regulatory network of the cell 145 of the first dataset inferred through the local feature ranking methods. Because MDA's rankings are exclusively non-negative, it is the only method that cannot infer whether a regulation is inhibiting or stimulating the expression of a gene. As explained, the ground-truth network stimulated by Dyngen contains very few interactions between genes.

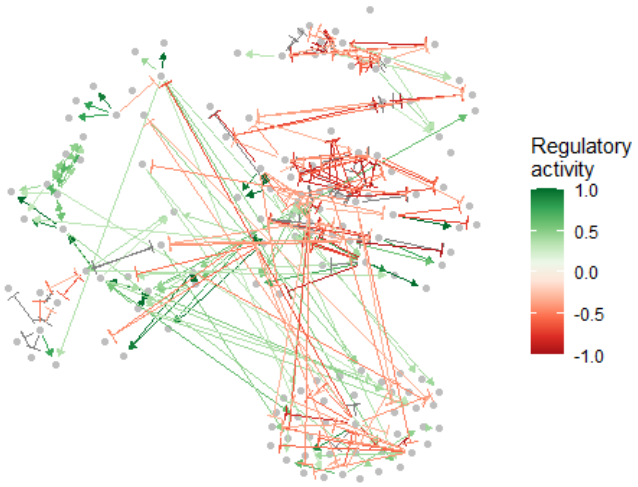

Figure 5.6: **Display of interactions predicted by SHAP:** The predictions correspond to the normalized feature importance levels (cell145 - Dataset1).

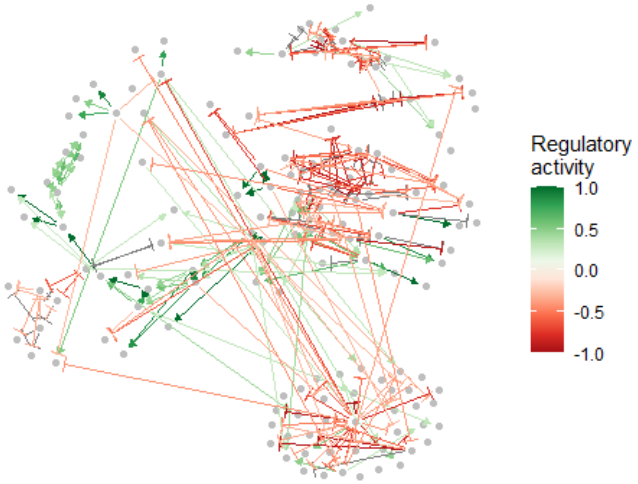

Figure 5.7: **Display of interactions predicted by SAABAS:** The predictions correspond to the normalized feature importance levels (cell145 - Dataset1).

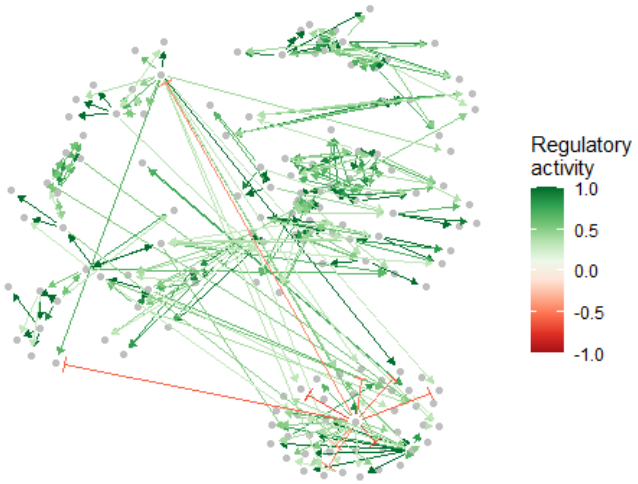

Figure 5.8: **Display of interactions predicted by MDI:** the predictions correspond to the normalized feature importance levels (cell145 - Dataset1).

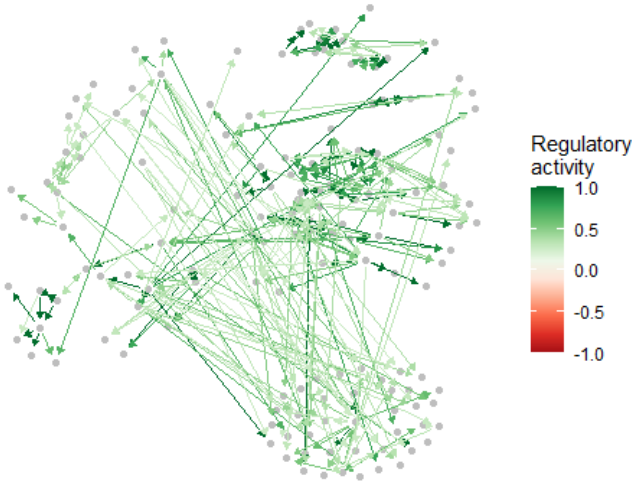

Figure 5.9: **Display of interactions predicted by MDA:** The predictions correspond to the normalized feature importance levels (cell145 - Dataset1).

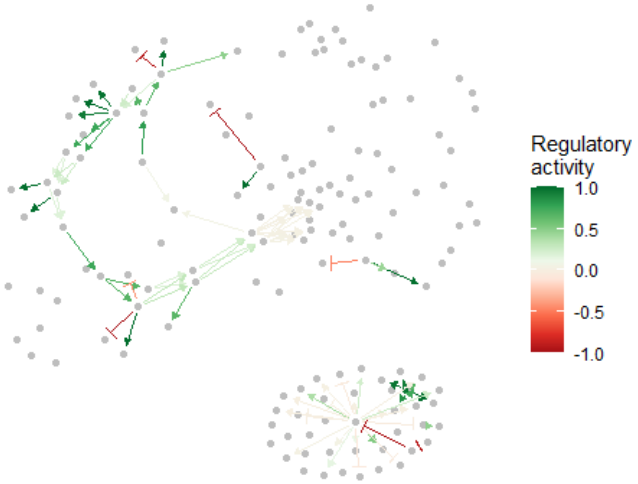

Figure 5.10: **Display of ground-truth regulatory network:** The predictions correspond to the normalized feature importance levels (cell145 - Dataset1).

5.4 Conclusion

The results from the application of local ranking methods to the Gene Network Inference problem are very promising. The four local methods studied in the thesis outperform the state of the art methods that are mainly used in the literature: SSN, $LIONESS$ and $pySCENIC$.

The local ranking methods are compared with ground-truth data, the cell-specific GRN inferred by the simulator and they have been measure to be effective. Indeed, the average AUROC score is high over all cells.

Chapter 6

Conclusion

This work studies recently developed local ranking methods. The rankings have been compared on several datasets and the influence of the forest's parameters are discussed in the fourth chapter. The methods are shown to be very correlated on non-fully randomized ensembles of trees ($K > 1$).

The methods have also been compared with ground-truth rankings through a linear regression problem and through the regulatory networks of single-cells. The latter application surely is the biggest contribution of the thesis as the studied methods outperform results previously obtained by state of the art methods.

The expressivity of the rankings derived on the classification task shows that MDA is particularly affected by K.

Throughout all experiments, SHAP and SAABAS have always been very correlated and they have been shown to provide good rankings on fully-randomized trees. MDA and MDI are shown to be much more dependent on the forest parameters. This find sense in the definition of the method that heavily rely on the structure of the tree. Additionally, increasing the value of K and the number of trees in the forest always increases the correlation between rankings. This leads to the conclusion that rankings tend to unify and therefore provide the same interpretation of a forest's predictions.

As local methods performed well on the Gene Network Inference task, it would be interesting to use the local methods on other inference tasks and see whether the good results obtained on the genes' networks can be reproduced in other fields of study.

Appendix A

Expressivity plots - Regression task

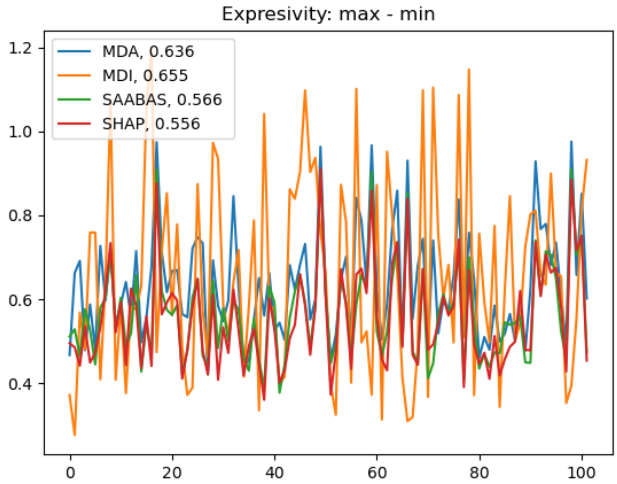

Figure A.1: **Expressivity of all methods:** $K = 1$ and max_depth $= 3$.

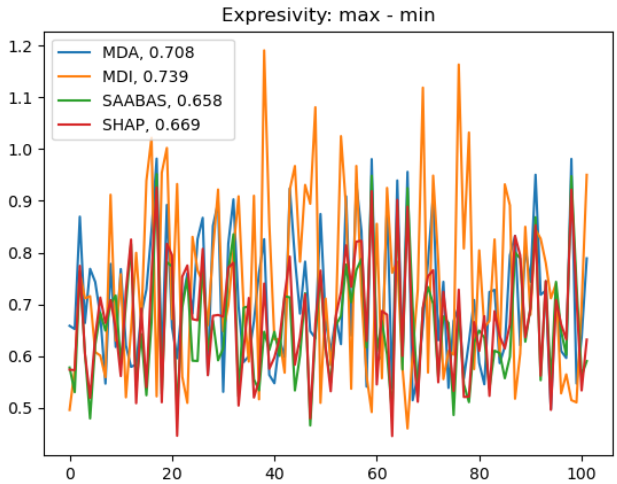

Figure A.2: **Expressivity of all methods:** $K = 3$ and max_depth $= 3$.

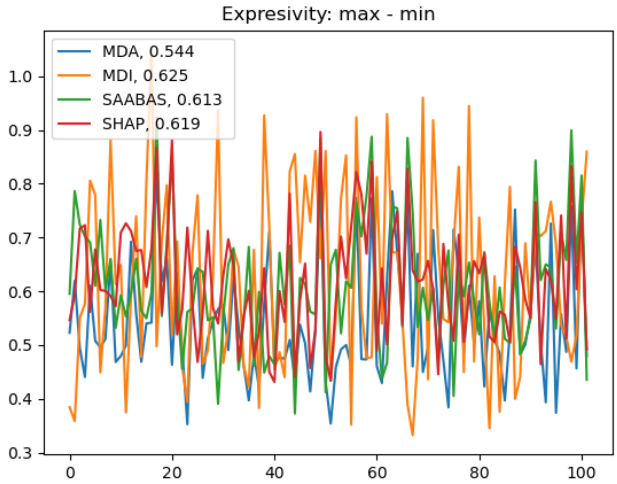

Figure A.3: **Expressivity of all methods:** $K = 1$ and no depth limit.

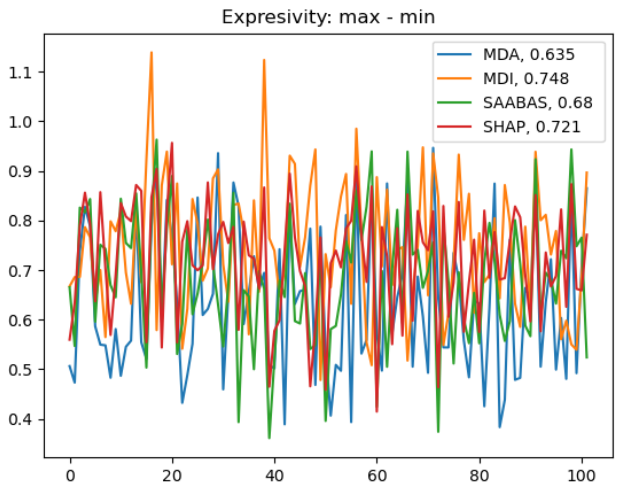

Figure A.4: **Expressivity of all methods:** $K = 3$ and no depth limit.

Bibliography

[1] Lidia Auret and Chris Aldrich. Empirical comparison of tree ensemble variable importance measures. Chemometrics and Intelligent Laboratory Systems - CHEMOMETR INTELL LAB SYST, 105:157–170, 02 2011.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. Classification and regression trees. Wadsworth and Brooks, Monterey, CA, 1984.

[3] Leo Breiman. Importance measures derived from random forests. PhD dissertation, 6 2001.

[4] Robrecht Cannoodt, Wouter Saelens, Louise Deconinck, and Yvan Saeys. dyngen: a multi-modal simulator for spearheading new single-cell omics analyses. bioRxiv, 2020.

[5] Shuonan Chen. Evaluating methods of inferring gene regulatory networks highlights their lack of performance for single cell gene expression data. BMC Bioinformatics, 19(none), June 2018.

[6] Jerome H. Friedman. Multivariate Adaptive Regression Splines. The Annals of Statistics, 19(1):1 – 67, 1991.

[7] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau-Malot. Variable selection using random forests. Pattern Recognition Letters, 31(14):2225–2236, 2010.

[8] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. Machine Learning, 63:3–42, 2006.

[9] Pierre Geurts. Contributions to decision tree induction: bias/variance tradeoff and time series classification. PhD thesis, University of Liege, Belgium, 2002.

[10] Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. PLOS ONE, 5(9):1–10, 09 2010.

[11] Marieke L. Kuijjer, John Quackenbush, and Kimberly Glass. lionessr: single-sample network reconstruction in r. bioRxiv, 2019.

[12] Xiaoping Liu, Yuetong Wang, Hongbin Ji, Kazuyuki Aihara, and Luonan Chen. Personalized characterization of diseases using sample-specific networks. Nucleic Acids Research, 44(22):e164–e164, 09 2016.

[13] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in forests of randomized trees. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 26. Curran Associates, Inc., 2013.

[14] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

[15] Markus Maucher, Barbara Kracher, Michael Kühl, and Hans A. Kestler. Inferring Boolean network structure via correlation. Bioinformatics, 27(11):1529–1536, 04 2011.

[16] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning – a brief history, state-of-the-art and challenges, 2020.

[17] Hastie R, Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. The Mathematical Intelligencer, CA, 2005.

[18] Ando Saabas. Interpreting random forests. https://blog.datadive.net/interpreting-random-forests - August 2021, October 2014.

[19] Marie Schrynemackers, Robert Kueffner, and Pierre Geurts. On protocols and measures for the validation of supervised methods for the inference of biological networks. Frontiers in Genetics, 4:262, 2013.

[20] Marie Schrynemackers, Robert Kueffner, and Pierre Geurts. On protocols and measures for the validation of supervised methods for the inference of biological networks. Frontiers in Genetics, 4:262, 2013.

[21] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. BMC bioinformatics, 9:307, 08 2008.

[22] Antonio Sutera. Importance measures derived from random forests: characterisation and extension. PhD thesis, University of Liege, Belgium, 2019.

[23] van de Sande, Bram and Flerin, Christopher and Davie, Kristofer and De Waegeneer, Maxime and Hulselmans, Gert and Aibar, Sara and Seurinck, Ruth and Saelens, Wouter and Cannoodt, Robrecht and Rouchon, Quentin and Verbeiren, Toni and De Maeyer, Dries and Reumers, Joke and Saeys, Yvan and Aerts, Stein. A scalable SCENIC workflow for single-cell gene regulatory network analysis. NATURE PROTOCOLS, 15(7):2247–2276, 2020.

[24] Louis Wehenkel. Classification and regression trees. https://people.montefiore.uliege.be/lwh/AIA, Uliege, Liege, Be, 2020.

[25] Jiangyong Wei, Xiaohua Hu, Xiufen Zou, and Tianhai Tian. Inference of genetic regulatory network for stem cell using single cells expression data. 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pages 217–222, 2016.