Wrocław University of Science and Technology

FIELD OF STUDY: Information and communication technology

PhD Thesis

TITLE OF THESIS:
Building transparent classification models

AUTHOR:
mgr inż. Bogdan Gulowaty

SUPERVISOR:
prof. dr hab. inż. Michał Woźniak

WROCŁAW 2024

List of Abbreviations

Acc Accuracy

AI Artificial Intelligence

BAC Balanced accuracy

CART Classification And Regression Tree

COMPAS Correctional Offender Management Profiling for Alternative Sanctions

CV Cross Validation

DT Decision Tree

EA Evolutionary Algorithms

EC Evolutionary Computing

Err Error rate

F1 F1 score

GDPR General Data Protection Regulation

GA Genetic Algorithm

GMean Geometric mean

GP Genetic Programming

Greedy Greedy rule list

IOT Internet of Things

KNN K-Nearest-Neighbours

LIME Local Interpretable Model-Agnostic Explanations

ML Machine Learning

MST Minimum Spanning Tree

NN Nearest-Neighbours

NOTE Non-overlapping Tree Ensemble

OneR One Rule

OVA One-Versus-All

OVO One-Versus-One

PCA Principal Component Analysis

Pre Precision

Rec Recall

RF Random Forest

Sen Sensitivity

SHAP Shapley Additive Explanations

Spe Specificity

SVC Support Vector Classifier

SVM Support Vector Machines

XAI Explainable Artificial Intelligence

List of Symbols

CM	Function calculating complexity metric
$\text{CM}(\mathcal{X})$	Value of complexity metric for set $\mathcal{X}$
$\mathcal{CS}$	Set of cliques in graph
d	Count of features
eval	Fitness evaluation function
indv	Genetic algorithm individual (chromosome)
j	Single instance's label
l	Cardinality of rules set
$\mathcal{LS}$	Training dataset
$\mathcal{M}$	Labels set
n	Number of data instances passed to procedure
N	Cardinality of training dataset
p	Single predicate (statement)
$\mathcal{PS}$	Set of predicates
Ψ	Classification algorithm
$\hat{\Psi}$	Ensemble of classifiers
r	Single rule
rel	Single relation in predicate
$\mathcal{RS}$	Set of rules
t	Count of competence areas
$\mathcal{TS}$	Testing dataset
$\mathcal{VS}$	Validation set
$\mathcal{X}$	Feature space
x	Training instance

$x^{(l)}$	The lth feature
y	Single instance's label

Abstract

As Artificial Intelligence (AI) and Machine Learning (ML) technologies advance and become deeply integrated into daily life, the need for transparent and interpretable models grows increasingly urgent. AI systems must be understandable, trustworthy, and ethical, especially in critical healthcare, finance, and legal sectors. The rise of Explainable Artificial Intelligence (XAI) seeks to address these challenges by providing explanations of model decisions, making AI systems more transparent. However, much of the progress in XAI has focused on deep neural networks, leaving other complex models like ensemble methods needing to be explored more regarding their interpretability.

The thesis aims to fill that gap by developing novel methods to improve the transparency and interpretability of ensemble classifiers while ensuring these models maintain competitive predictive performance and build inherently transparent models. The central hypothesis of the thesis is that it is possible to construct such transparent or explainable models that perform as well as black-box models in a wide range of classification tasks. The work focuses on three primary methods designed to either explain or replace complex models with transparent alternatives: Non-overlapping Tree Ensemble (NOTE), Optimal Centroids and Quad Split algorithms. The thesis sets out with the following objectives:

- Develop novel algorithms that produce transparent classification models.
- Extract interpretable models from complex ensemble classifiers like Random Forest (RF).
- Use data complexity metrics to assess and improve the performance of transparent models.
- Evaluate the effectiveness of these methods across datasets of varying complexities, ensuring the models are both interpretable and accurate.

The main contribution of the thesis is the introduction and evaluation of three novel algorithms in the domain of XAI:

NOTE The NOTE method is specifically designed to explain tree-based ensemble models like Random Forests. While traditional ensemble models can combine transparent base models like Decision Trees, they often produce overlapping decision boundaries, making the ensemble challenging to interpret. NOTE addresses this by selecting a subset of trees that create non-overlapping decision areas, simplifying the overall model. The proposed method applies graph analysis techniques to identify which rules, coming from Decision Trees in the ensemble, may be used in a way that does not overlap and provide the best generalizing abilities. The areas covered by selected rules are assigned Decision Trees that are trained and combined to form a transparent version of the original model, allowing users to understand individual predictions more easily. Experimental results

showed that NOTE can significantly improve interpretability without significantly losing predictive performance. In datasets of moderate complexity, NOTE performed comparably to the original RF while providing much more precise insights into the decision-making process.

Optimal Centroids The Optimal Centroids algorithm focuses on creating interpretable classification models by splitting the feature space into regions based on centroids and 1-Nearest-Neighbours (NN) classifier. In the method, the feature space is divided according to centroids found by evolutionary algorithms, such as genetic algorithms. Centroid positions are being evaluated to find their best distribution, which would maximize the model accuracy. As part of the evaluation step, transparent models are trained for decision space designated by every centroid. The model created in such a way has high interpretability and solid predictive abilities.

Quad Split The Quad Split is a novel algorithm for creating transparent classification models that has been introduced to compete with more complex models by creating interpretable decision boundaries. The idea behind the Quad Split algorithm is based on the assumption that multiple specialized models applied to parts of the whole dataset will perform better than one. To achieve that, Quad Split recursively splits the decision space based on feature values extracted from the training set. Those values serve as splitting points. The algorithm evaluates those points to find one minimizing data complexity at both sides of the split. Then, when stop criteria are reached, the interpretable model is created in every split, finally resulting in an ensemble of transparent models, which could be reduced to a list of simple rules.

Experimental Evaluation The thesis conducted a comprehensive evaluation of the three proposed methods using 16 binary datasets with varying complexities. These datasets included real-world and synthetic examples, covering a range of domains and challenges such as class imbalance, high-dimensional spaces, and noisy data. Complexity metrics from various categories, such as feature, dimensionality, and linearity-based, were evaluated. The experimental setup focused on several key evaluation metrics:

- Predictive abilities: The performances of each model, as defined by balanced accuracy, F1 score, and geometric mean score, were compared against standard classifiers like Random Forests, Decision Trees and rule models, focusing on overall prediction accuracy.

- Model complexity: Base and proposed model complexities were quantified and evaluated against each other.

- Effectiveness as explainers: The ability of the models to explain traditional black-box models like Random Forests was also assessed. The experiments evaluated how well the proposed methods could serve as surrogate explainers for these complex models.

All the above qualities were evaluated in the domain of changing internal algorithm parameters, as well as different dataset complexities - defined by the aforementioned complexity metrics.

Conclusion and Future Work The thesis contributes to the science of AI by demonstrating that transparent and explainable models can be built without sacrificing predictive abilities, particularly in moderately complex datasets. The proposed methods — NOTE, Optimal Centroids, and Quad Split — offer flexible and interpretable alternatives to traditional black-box ensemble models. The main achievements of the thesis are:

- The development of two novel algorithms that create inherently transparent models while maintaining competitive performance.
- The development of the forest ensemble explaining algorithm that is competitive to the explained RF. Extensive experimental evaluation shows that the proposed methods can explain and, in some cases, replace black-box models with transparent alternatives.
- A demonstration that data complexity metrics play an important role in determining when transparent models are most effective.
- Development of a programming library containing the algorithms mentioned above.

The three proposed methods offer significant improvements in interpretability, making them suitable for applications where trust and transparency are essential, such as in healthcare, finance, or legal systems. Along with that, several observations are made, such as that:

- Evaluated classification models behave immensely differently when applied to datasets with different complexity properties.
- Internal model complexities do not always correlate to dataset complexities.
- Evaluated explaining methods that used knowledge extracted from complex models behaved better than inherently transparent ones when used as explainers.

Future research could focus on further refining the proposed methods in various ways, such as fine-tuning the Genetic Algorithm (GA) parameters of Optimal Centroids or finding better evaluation metrics for NOTE. Additionally, the applicability of the algorithms could be checked in a wider range of complex, black-box methods, such as neural networks. Observations made in the thesis might also help develop meta-algorithms based on datasets' complexity metrics. The thesis makes significant contributions to the field of XAI by demonstrating that transparent models can be built without sacrificing predictive abilities.

Streszczenie

W miarę jak technologie Sztucznej Inteligencji (SI) i Uczenia Maszynowego rozwijają się i stają się coraz bardziej zintegrowane z codziennym życiem, potrzeba budowania przejrzystych i interpretowalnych modeli staje się coraz bardziej istotna. Systemy SI muszą być zrozumiałe, godne zaufania i etyczne, zwłaszcza te używane w kluczowych sektorach, takich jak opieka zdrowotna, finanse i systemy prawne. Rozwój Wyjaśnialnej Sztucznej Inteligencji (eXplainable Artificial Intelligence - XAI) ma na celu sprostanie tym wyzwaniom poprzez m.in. tworzenie bardziej przejrzystych i intepretowalnych modeli. Jednak większość prac dotyczących XAI koncentruje się na głębokich sieciach neuronowych, pozostawiając inne złożone modele, takie jak np. metody zespołowe bez rozwiązań w tym zakresie. Jednym z celów rozprawy jest wypełnienie tej luki poprzez opracowanie nowatorskich metod poprawy przejrzystości i interpretowalności klasyfikatorów zespołowych, przy jednoczesnym zachowaniu ich wysokiej jakości predykcji oraz budowę modeli klasyfikacji, które z góry zakładają swoją przejrzystosć. Główna hipoteza pracy stwierdza, że możliwe jest skonstruowanie takich przejrzystych lub wyjaśnialnych modeli, które będą działać równie dobrze jak modele typu "black box" w szerokim zakresie zadań predykcji danych. W pracy zaprezentowano trzy autorskie algorytmy wyjaśniania lub zastępowania złożonych modeli przez transparentne alternatywy: Non-overlapping Tree Ensemble (NOTE), Optimal Centroids oraz Quad Split. Przyjęte cele pracy obejmują:

- Opracowanie nowych algorytmów tworzących przejrzyste modele klasyfikacyjne.
- Ekstrakcję interpretowalnych modeli ze złożonych klasyfikatorów zespołowych, takich jak Random Forest (RF).
- Wykorzystanie metryk złożoności danych do oceny i poprawy wydajności przejrzystych modeli.
- Ocena skuteczności tych metod na zbiorach danych o różnej złożoności aby zapewnić, że modele są zarówno interpretowalne, jak i mają wysokie zdolności predykcyjne.

Głównym wkładem tej pracy jest zaproponowanie i ocena trzech nowatorskich algorytmów w dziedzinie XAI:

NOTE Metoda NOTE została zaprojektowana specjalnie do wyjaśniania zespołów modeli drzewiastych, takich jak RF. Tradycyjne modele zespołowe cechują się krokiem integracji predykcji z modeli będącymi członkami zespołu, co utrudnia interpretację zespołu. NOTE rozwiązuje ten problem, wybierając podzbiór drzew, które tworzą nienachodzące na siebie obszary decyzyjne, upraszczając ogólny model. Proponowana metoda stosuje techniki analizy grafów w celu zidentyfikowania reguł pochodzących z drzew decyzyjnych w zespole, które można wykorzystać do zdefiniowania nienachodzących na siebie obszarów,

zapewniając przy tym jak najlepsze możliwości predykcyjne modelu. Obszary pokryte przez wybrane reguły są przypisywane do drzew decyzyjnych, które są trenowane i łączone w komitet, tworząc przejrzystą wersję oryginalnego modelu, umożliwiając użytkownikom łatwiejsze zrozumienie indywidualnych predykcji. Wyniki eksperymentów wykazały, że metoda NOTE poprawia interpretowalność, nie tracąc przy tym znacząco na wydajności predykcyjnej. Na zbiorach danych o umiarkowanej złożoności metoda NOTE osiągnęła wyniki porównywalne z oryginalnym Random Forest, jednocześnie dostarczając dużo precyzyjniejszych informacji o procesie podejmowania decyzji.

Optimal Centroids Algorytm Optimal Centroids koncentruje się na tworzeniu interpretowalnych modeli klasyfikacyjnych poprzez podział przestrzeni cech na regiony w oparciu o centroidy i klasyfikator 1-NN. W tej metodzie przestrzeń cech jest podzielona zgodnie z centroidami, które są optymalizowane za pomocą algorytmów ewolucyjnych, takich jak algorytm genetyczny. Pozycje centroidów są oceniane w celu znalezienia ich najlepszej dystrybucji, która maksymalizyje dokładność modelu. W ramach kroku ewaluacji trenowane są przejrzyste modele dla przestrzeni decyzyjnych wyznaczonych przez centroidy. Tak wytrenowany model cechuje się wysoką interpretowalnością i dobrymi zdolnościami predykcyjnymi.

Quad Split Algorytm Quad Split to nowatorska metoda tworzenia przejrzystych modeli klasyfikacyjnych, zaprojektowana z myślą o konkurowaniu z bardziej złożonymi modelami poprzez tworzenie interpretowalnych granic decyzyjnych. Algorytm ten zakłada, że zastosowanie wielu wyspecjalizowanych modeli do części zbioru danych przyniesie lepsze wyniki niż jeden model globalny. Aby to osiągnąć, Quad Split rekursywnie dzieli przestrzeń decyzyjną w oparciu o wartości cech wyodrębnionych ze zbioru treningowego. Wartości te służą jako punkty podziału. Algorytm ocenia te punkty, aby znaleźć taki, który minimalizuje złożoność danych po obu stronach podziału. Po osiągnięciu kryteriów zatrzymania w każdym podziale tworzony jest interpretowalny model, co ostatecznie prowadzi do utworzenia zespołu przejrzystych modeli, który można sprowadzić do listy prostych reguł.

Ocena eksperymentalna W ramach pracy przeprowadzono szeroki zakres badań eksperymentalnych trzech proponowanych metod, używając 16 zbiorów danych do zadania binarnej klasyfikacji o zróżnicowanej złożoności. Zbiory te obejmowały zarówno przykłady rzeczywiste, jak i syntetyczne, dotyczyczące różnych dziedzin i wyzwań, takich jak niezbalansowane rozkłady klas, przestrzenie o wysokiej wymiarowości oraz szum danych. Eksperymenty brały pod uwagę ocenę złożoności zbiorów danych w ramach różnych kategorii, takich jak cechy, wymiarowość i liniowość. Eksperymenty koncentrowały się na kilku kluczowych wskaźnikach oceny:

- Zdolności predykcyjne: Wyniki każdego modelu, mierzone przez zbalansowaną dokładność, F1-score i średnią geometryczną, zostały porównane ze standardowymi klasyfikatorami, takimi jak Random Forest, drzewa decyzyjne oraz modele regułowe, z naciskiem na ogólną dokładność predykcji.
- Złożoność modelu: Złożoność podstawowych i proponowanych modeli została skwantyfikowana i oceniona w stosunku do siebie.
- Skuteczność jako metody wyjaśniania modeli: Oceniono zdolność proponowanych algorytmów do wyjaśniania modeli typu "czarna skrzynka", takich jak Random

Forest. Eksperymenty oceniły, jak dobrze proponowane metody mogą pełnić rolę wyjaśniających modeli zastępczych dla tych bardziej złożonych.

Wszystkie powyższe cechy zostały ocenione w kontekście zmieniających się parametrów wewnętrznych algorytmów oraz różnych złożoności zbiorów danych, zdefiniowanych przez wspomniane metryki złożoności.

Wnioski i przyszłe kierunki badań Praca wnosi istotny wkład do rozwoju AI, pokazując, że można budować przejrzyste i wyjaśnialne modele bez poświęcania ich zdolności predykcyjnych. Proponowane metody — NOTE, Optimal Centroids i Quad Split — oferują elastyczne i interpretowalne alternatywy dla tradycyjnych modeli zespołowych typu "czarna skrzynka". Główne osiągnięcia pracy obejmują:

- Opracowanie dwóch nowatorskich algorytmów, które generują przejrzyste modele przy zachowaniu konkurencyjnych zdolności predykcji i generalizacji.
- Opracowanie algorytmu wyjaśniającego dla komitetów drzew decyzyjnych, który jest konkurencyjny wobec algorytmu RF.
- Szeroką ocenę eksperymentalną, pokazującą, że proponowane metody mogą wyjaśniać, a w niektórych przypadkach zastępować modele typu "czarna skrzynka" przejrzystymi alternatywami.
- Wykazanie, że metryki złożoności danych odgrywają ważną rolę w określaniu, kiedy przejrzyste modele są najbardziej efektywne.

Trzy zaproponowane metody wnoszą znaczące ulepszenia w zakresie interpretowalności, co czyni je odpowiednimi do zastosowań, w których zaufanie i przejrzystość są kluczowe, takich jak opieka zdrowotna, finanse czy zastosowanie w problemach prawnych. Oprócz powyższego, poczyniono kilka obserwacji, takich jak:

- Oceniane modele klasyfikacyjne zachowują się znacząco inaczej, gdy są stosowane do zbiorów danych o różnych właściwościach złożoności.
- Wewnętrzna złożoność modeli nie zawsze koreluje ze złożonością zbiorów danych.
- Oceniane metody wyjaśniające, które wykorzystywały wiedzę wyodrębnioną z złożonych modeli, działały lepiej niż z natury przejrzyste modele, gdy były stosowane jako modele wyjaśniające.

Przyszłe badania mogłyby skupić się na dalszym udoskonalaniu proponowanych metod, na przykład poprzez dostrajanie parametrów algorytmu genetycznego w metodzie Optimal Centroids lub poszukiwanie lepszych metryk oceny dla NOTE. Dodatkowo uzasadnione jest sprawdzenie zastosowanie algorytmów w szerszym zakresie złożonych metod typu "czarna skrzynka", takich jak sieci neuronowe. Obserwacje poczynione w pracy mogą również pomóc w opracowaniu meta-algorytmów opartych na metrykach złożoności zbiorów danych. Praca wnosi znaczący wkład do dziedziny XAI, wykazując, że można budować przejrzyste modele bez poświęcania ich zdolności predykcyjnych.

Contents

Chapter 1

Introduction

1.1 Data as power driving world

The statement saying that "data rules the world", without much thought, hesitation, or analytical thinking, could be considered true by most people living and utilising inventions of the modern world. One could also say that it has been true for decades, if not centuries. Gathering information and potentially using them to raise some generic conclusions is a tool that has been used by *homo sapiens* since the beginning of time. What was Socrates doing, if not utilising data he collected during his lifetime about human nature to provide insight to his fellow Athenians about their day-to-day behaviours? What would military leaders do without data about enemy whereabouts, which consumed, resulted in specific moves of their armies. A well-managed country with an economy that works like clockwork was only possible by processing data and drawing conclusions from it.

Data-driven decisions have ruled the world for centuries, and now, on an unprecedented scale, various information is utilised and embedded within our day-to-day activities. Nowadays, it starts with insignificant actions such as walking somewhere with a smartphone in our pocket, entering any website or even buying apples in the grocery shop nearby. The majority of our everyday life activities may result in producing some data – let it be GPS tracking to estimate traffic in some areas of the city you live in, analysing user behaviour in a website to create new features and monetise existing ones in the optimised way or to process invoices by the accountant of the grocery shop.

The scope of stored and processed data has continued to grow exponentially during the last few years. The source of this phenomenon is not hard to grasp. Due to technological advantages, we are able to link technology to more and more spheres of our lives. Forecasts predict that up until *2030*, there will be almost 30 billion Internet of Things (IOT) devices interconnected (Figure 1.1). The global amount of data created is predicted to reach 175 Zettabytes in 2025 [1].

1.2 Ethics of AI

With the aforementioned number of developments come numerous worries. Those worries apply to us humans from a broad spectrum of angles. With an extended amount of image recognition research come concerns for privacy and increased surveillance. It is becoming harder to control and perceive who processes or stores our data, which became *defacto* currency of the digital world. The issue has become so significant that to counter

[1]Source: statista.com

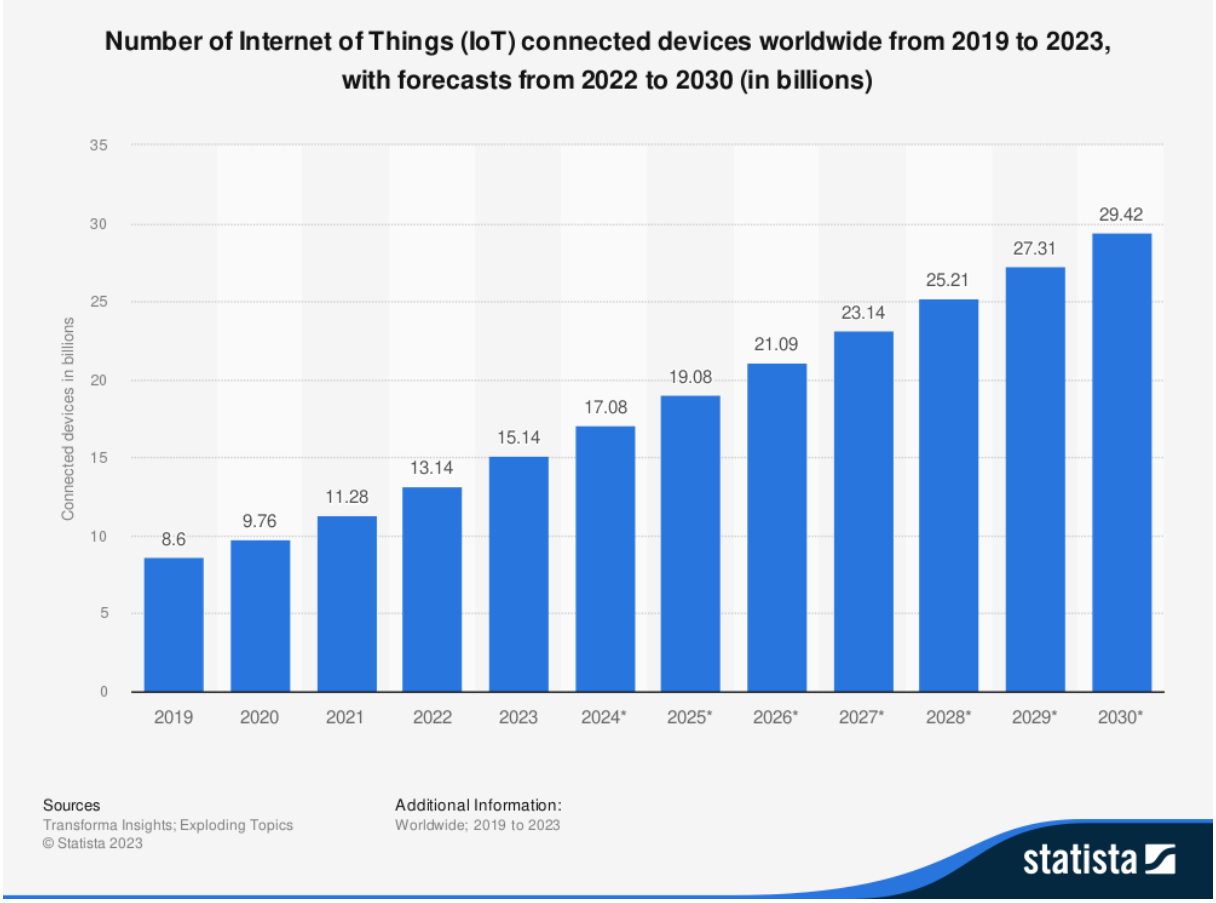

Figure 1.1: IOT connected devices worldwide from 2019 to 2023 with future forecasts[1]

that (and create a business out of it), companies that delete our digital fingerprints have started to emerge.

The risk of manipulation leading to the controlling behaviour of social groups is imminent. A recent example of such behaviour can be found in the Israeli-Hamas war, where participants manipulated images posted on social media to undermine the other side [2]. Currently, developments in generative imaging, voice, and video generation have come to such quality that it is already hard for consumers to distinguish them from reality. This, apart from providing the ability to create actual value for humanity, provides a great toolset for fake news [3]. Even the fact of open-sourcing those tools raises ethical concerns, as from one perspective, it supports freedom and allows individuals to participate and share global wealth of development. But from others – it allows evildoers to mislead and gain profit in doubtful ways.

Even though Artificial Intelligence (AI) backed systems lead to increased productivity and operating cost reduction, AI developments result in computer systems taking on jobs that do not require specialised knowledge. In return, it may lead to increased unemployment and forced reduction of job places in different areas. Currently observed is the polarisation of workplace distribution toward increased demand for highly skilled technical jobs and low skilled service positions, but decreasing demand for mid-qualification occupations, such as factory or office workers [4].

All those concerns are just scratching the tip of the iceberg – as new developments arise, so do new ethical issues. Last but not least, whether the concept of Artificial General Intelligence is possible is still up to debate [5], [6], one more ethical question remains –

who and why should gather and control vast amount of data that is currently created in the global sphere and how to incorporate individual privacy and safety into the picture. [7]

1.3 Explaining behavior of classification models

Explainable Artificial Intelligence (XAI) is currently one of the most important trends of AI [8]. There are practical reasons behind the need to build explainable systems, primarily related to the need to ensure that the solutions used are ethical [9], not biased [10]. We can fully trust them [11]. Many well-known examples may be cited to support the above observations, such as the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) algorithm for predicting recidivism, which issued decisions that were biased against Afro-Americans by recognizing that the risk of re-offending in this group is highest [12]. It is also important to be able to use the information to explain how the model works as it is being built, including to guard against the influence of so-called adversarial examples [13], or to respond to a change in task parameters when concept drift occurs. Regulations are also an important aspect. Currently, the European Union defines solutions classified as so-called *Trustworthy Artificial Intelligence*, as one that realizes the "Right to Explanation" [14].

Biecek and Burzykowski [15] pointed out that one of the critical bottlenecks in predictive modelling is the need for more explanation tools for the predictors. We may take one of two possible directions in the model explanation. Either we may build models that are, by their nature, *glass-box* models, which generally use a symbolic knowledge representation. This group includes models like decision trees, k-nearest-neighbors, linear/logistic regressions, rule-based models and general additive models [16]. Alternatively, construct *black-box* models, which have complex structures and are not interpretable by humans, e.g., due to the use of complex mathematical transformations, and treat the explanation of the model as a separate task. This class includes both models that focus on explaining behaviour for single observations such as Local Interpretable Model-Agnostic Explanations (LIME) [17] and Shapley Additive Explanations (SHAP) [18], or approaches that offer model translation as a whole, e.g., by building surrogate models, e.g., extracting interpretable models from complex models such as RF [19], or class models visualization [20].

1.4 Need for this thesis

The dynamic growth of XAI landscape can be observed for some time now, having dedicated conference tracks dating back to as far as 2017 [21]. As of the time span of the development of this thesis, the scientific literature in the field of Machine Learning (ML) and AI has received increasing focus in areas of Explainability and Interpretability. In particular, a major research effort was undertaken in order to utilise the predictive power of various types of neural networks whilst tackling the issue of its lack of transparency [8], [22]–[24]. Rudin [25] points out noteworthy issues with the development of the XAI scenery as a whole. Very often, one can either use an existing high-performing black-box model with use some kind of explainer or (given that training data is available) build the new transparent model. Rudin argues that one of the issues is the fact that explainable ML models provide explanations which are not faithful to the explained model. Having this in mind, the fact – that, to some extent – a surrogate model created on a black

box will make mistakes makes it harder to trust it and thus makes its application to real-world problems more difficult. Trust, often needed in high-stakes decision-making, is considered one of the major drivers in the development of the XAI domain [26]. Nowadays, in public domains, application qualities attributing to trust are being enforced by various government-level policies and regulations, such as General Data Protection Regulation (GDPR) [27]. The next important factor is that explanations often do not provide enough details to understand what the explained model is doing. A surrogate model, created as a black-box explainer, might use completely different features and have different internal decision-making mechanics. An example of this is *saliency map*, which points out where a neural classification is looking in the image. In the literature, it is often considered as one of XAI methods, while all it does is pointing out where the model is looking – without providing reasoning or explanation behind the decision. Having a surrogate explainer poses an additional issue. When presented with potentially faulty data (for example, as inputted to the system by human error), debugging the whole requires examining not one but two models. It is also worth noting that in contrast to neural network XAI developments, much less research is being done in terms of explaining other complex models, such as meta ensembles. Additionally, in the scientific community, there is a lack of work analysing the utilisation of proposed classification methods under specific conditions of classified datasets, defined by their complexity.

Given all the above – mainly the need for the development and employment of increasingly better transparent classification models – this work strives to connect all the dots by stating the following hypothesis:

For a given classification task, it is possible to build such a transparent or explainable model whose quality is not worse than a similarly applied black-box model

To support or dismiss the above claim, the following objectives were formulated:

- Development of novel ensemble "glass-box" model extraction method
- Development of novel transparent, "white-box" models
- Utilization of data complexity metrics in developing novel transparent classification method
- Experimental evaluation of proposed algorithms using a wide array of datasets attributed with different complexities
- Designing and implementing a programming library

Completion, and as a natural part, development of new methods will hopefully bring value to the scientific community and be a building block of further research.

1.5 Structure of thesis

This thesis is structured as follows. Its core consists of three chapters presenting developed and verified during PhD study methods. The first method, called NOTE, presented in chapter 3, was developed strictly as a way of explaining tree ensemble models. It bases its internal work on extracting trees from the explained model and using a graph analysis toolkit to find the best possible combination of the final model. The following

ones (chapters 4 and 5) were developed as novel transparent models (named respectively Optimal Centroids and Quad Split), which – after some modifications – are also used as model explainers. To set some common ground for the reader, in the next chapter, shared background knowledge is presented that – to some extent – is needed to understand each of the core chapters' algorithms. Finally, methods are compared to each other in terms of their performance. A common experimentation protocol toolkit is also presented in chapter 2, along with details on used datasets. Then, methods are discussed and evaluated altogether in chapter 6, which is followed by the last part containing conclusions and further research proposals.

Chapter 2

Background and related works

The preliminary knowledge needed to proceed further into understanding of this thesis spans across multiple areas: supervised learning, building and using ensemble models, data complexity measures and explainable artificial intelligence. In the following sections those topics will be expanded upon in scope that is shared among proposed methods background. The detailed literature review of specific research relating to described, proposed methods and algorithms will follow in next chapters.

2.1 Supervised learning classification

In supervised learning, the model (estimator) is trained on the basis of the labelled learning set $\mathcal{LS} = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ where x is d-dimensional feature vector $x = [x^{(1)}, x^{(2)}, ..., x^{(d)}]^T \in \mathcal{X}$, and $y \in \mathcal{M}$ is its label, coming from discrete and finite $\mathcal{M} = \{y_1, y_2, ..., y_N\}$. Therefore, single training dataset sample consists of d features, which can be qualitative of quantitative.

Subset of qualitative data is "nominal" (coming from Latin word "nomen" which means name), that is one where categories/labels are used without any meaningful order. Examples of such data could be hair colour - "red", "black", or "blonde". When the order of labels brings information, we talk about the ordinal features. Examples of such can be grades - "A", "B+", and "C-".

When it comes to quantitative data, we mean numbers that can either be discrete (cost of product) or continuous (height of person).

A classifier Ψ is such a function, that maps the feature space into the set of labels:

$$\Psi : \mathcal{X} \to \mathcal{M} \tag{2.1}$$

Methods of building predictive models can be divided into parametric and non-parametric. In the first case, an assumption is made about the functional form of the estimated relationship. For example, given training data, a linear relationship could be observed and thus, linear model coefficients(parameters) will be estimated using the least squares algorithm.

Non-parametric methods do not make assumptions about the shape of the estimated function. Instead of that, those methods try to find such a model that fits all the data points as closely as possible.

Most used supervised classification algorithms include Classification And Regression Tree (CART), K-Nearest-Neighbours (KNN), Support Vector Machines (SVM), *Naive Bayes* and some of them will be described later in this chapter [28].

2.1.1 Ensemble learning

Ensemble learning is a term for a group of methods that combine multiple base models to make a decision. A base model is classification algorithm that produces a predictive model. This approach is based on the assumption that when combining multiple models, the error of a single classifier in the pool will be mitigated by others, thus creating a stable, accurate model with low variance. Nowadays ensembles are considered a state-of-the-art approach for solving machine learning problems [29].

A classifier ensemble $\hat{\Psi}$ employs a pool of base classifiers:

$$\Pi = \{\Psi_1, \Psi_2, ..., \Psi_k\} \tag{2.2}$$

and makes a decision based on a function $\mathcal{F}$, that combines predictions provided by the base classifiers:

$$\hat{\Psi}(x) = \mathcal{F}(\Psi_1(x), \Psi_2(x), ..., \Psi_k(x)) \tag{2.3}$$

There are several reasons behind why such approach works [30]:

- It reduces over-fitting by averaging different hypotheses induced by multiple base learners
- It reduces the possibility of being stuck in local optima when learning due to fact of having multiple learners
- Hypotheses solution space for combined classifier can be greater than for single learners due to combination step mixing multiple outputs

Many factors exist that could be manipulated when building ensemble models. The building process may vary for example by how initial training dataset is being split, which features are being fed to what learners, how outputs are combined (if they are weighted), if samples for learning are drawn with replacement and finally, whether the outcome ensemble is homogeneous (consisting only of models of one induction algorithm) of heterogeneous. Ensemble learning can be applied in various areas of AI, such as Reinforcement Learning [31], regression [32] or data streams classification [33]. Below are briefly described some of the most commonly used approaches for supervised learning.

Bootstrap aggregating (bagging) is one of the simplest and most popular ways to build ensemble models. Given training set $\mathcal{LS}$ algorithm creates n so-called bootstrap datasets by sampling the original dataset uniformly with replacement. Then, those datasets are fed into the base learner, effectively creating an ensemble of multiple models. Predictions of the ensemble are made by majority voting of the models [34].

Boosting is a technique of iteratively improving weak learners. Initially, some base algorithm is fed original data. Then, higher weights are assigned to samples which were misclassified in the previous step, and the next iteration follows, where the algorithm is fed weighted samples of original data. Procedure repeats until certain stop criteria is reached (such as number of iterations). The final ensemble takes the form of weak models weighted by their performance [35].

Stacking , also known as a two-level ensemble, works by randomly training weak learners, whose output is then taken by a meta-learner that raises the final prediction. Each of the first-level learners returns some predicted label or support for given class. Those values are then fed into single final meta-learner to produce final prediction of the ensemble [36].

2.1.2 Metrics and evaluation in supervised classification

In order to measure how good a given classification model performs, several well-established metrics exist [37]. In this section, the most common of those metrics are examined and picked as a base for further experimental evaluation.

In the case of binary classification problems, typically the base output of model evaluation is *confusion matrix*. It can be generalized to accumulate any number of classes. The matrix for the binary case is depicted in Figure 2.1.

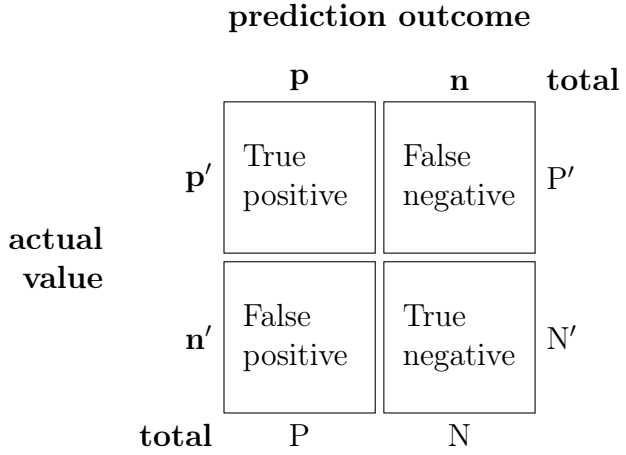

Figure 2.1: Confusion matrix.

It consists of 4 metrics, which indicate number of samples, that model classified in certain way:

- **TP** - true positive - number of positive instances classified as positive
- **TN** - true negative - number of negative instances classified as negative
- **FP** - false positive - number of instances which belonged to negative class, and were classified as positive
- **FN** - false negative - number instances which belonged to positive class, and were classified as negative

.
In the case of multiclass problems, such matrix can be either multiplied in One-Versus-All (OVA) method or expanded by rows and columns for more classes. Based on those four values, number of classification metrics can be derived. One of the most common and intuitive being:
Accuracy (Acc) defined as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

Intuitively its ratio of correctly classified instances to all instances
Error rate (Err) defined as

$$Err = \frac{FN + FP}{TP + TN + FP + FN} = 1 - \text{Acc} \tag{2.5}$$

which, in contrast to Accuracy, measures how often the model makes an error. Additionally, the confusion matrix serves as basis for:
Recall (Rec) (also known as Sensitivity (Sen)), which measures, out of all examples predicted as positive, how many are positive:

$$Rec = \frac{TP}{TP + FN} \tag{2.6}$$

Precision (Pre), which gives insight into how many of instances predicted as positive are actually positive:

$$Pre = \frac{TP}{TP + FP} \tag{2.7}$$

Specificity (Spe), which is counterpart of recall for negative class:

$$Spe = \frac{TN}{TN + FP} \tag{2.8}$$

Those more specific measures can be used to prioritize specific behaviors of the outcome model. For example, in an area where making an error in classifying an instance as negative, such as Anti Money Laundering or Transaction Screening system [38], one might prioritize maximizing specificity, therefore picking a model that would sacrifice accuracy in favour of it.

Data imbalance Some of the abovementioned fall short when classified datasets' characteristics would be, that it's strongly imbalanced. Taking an Acc as an example, let's imagine that the evaluation dataset on which a given model is tested has 99 positive examples and one negative. Having a simple model that classifies every presented instance as positive would raise Acc score of 99% and Pre as 100%. Even though it would fail to completely recognize underrepresented negative class.

Having this in mind, let's examine the next set of derived metrics.
BAC measure takes into consideration class imbalanced:

$$BAC = \frac{Rec + Spe}{2} \tag{2.9}$$

GMean squares class-wise recall:

$$\text{Gmean} = \sqrt{Rec \times Spe} \tag{2.10}$$

F1 is specific case of F-β score, which considers weighted precision and recall:

$$F1 = 2\frac{Pre \times Rec}{Pre + Rec} \tag{2.11}$$

.

All of the above-mentioned metrics may be used in multi-label classification problems by employing either micro or macro averaging. Macro-averaging assigns equal weight to instances and classes, where micro-average is preferred where potential class-imbalance is suspected [39].

2.2 Classification algorithms

In this section, a brief overview of the most used and important from the perspective of further examination algorithms is presented.

Decision Tree

Originally introduced in [40] by Quinlan, ID3 (iterative dichotomies 3) decision tree induction algorithm served as base for further research of what is today one of most popular machine learning models. It is based on splitting decision space using entropy defined as

$$Entropy(S) = \sum_{i=1}^{C} P_i log_2 P_i \tag{2.12}$$

where P_i represents probability of random variable having class i.

Another, often used for splitting measure is information gain (or mutual information). In contrast to entropy, higher values indicate better split. Intuitively it provides insight into how much knowledge random variable's value brings:

$$Gain(S, A) = Entropy(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2.13}$$

where $V(A)$ are possible values of attribute A and S_v is subset of S equal to value V.

Nowadays, many decision tree induction algorithms exist. Amongst them C4.5 (successor of ID3), CART, Chi-squared Automatic Interaction Detection (CHAID), Multivariate Adaptive Regression Splines (MARS), Generalized Unbiased Interaction Detection and Estimation (GUIDE), Conditional Inference Trees (CTREE), Classification Rule with Unbiased Interaction Selection and Estimation (CRUISE), Quick, Unbiased and Efficient Statistical Tree (QUEST) [41].

After the induction step, the output decision tree usually takes a very simple and interpretable form consisting of a root, decision nodes and terminal nodes, an example of which is depicted in Figure 2.2.

Random Forest

RF is an ensemble learning algorithm where the final output is so-called "forest" of decision trees. Each tree in the ensemble is fed a sample drawn with replacement from the training set. When splitting each node during the construction of a tree, the best split is found through a search of the feature values of either all input features or a random subset defined by a learning parameter. The aim of such approach is to decrease the variance of the forest estimator. While individual decision trees typically display high variance and tend to overfit (when not pruned), by combining their output, Random Forest reduces those phenomena.

Those two sources of randomness in forests provide decision trees with reduced prediction errors. By taking an average of those predictions, some errors may cancel out. Random forests achieve a reduced variance by combining diverse trees, In practice the variance reduction is often significant hence yielding an overall better model [42], [43]. The visualization of final model created by RF is present in Figure 2.3.

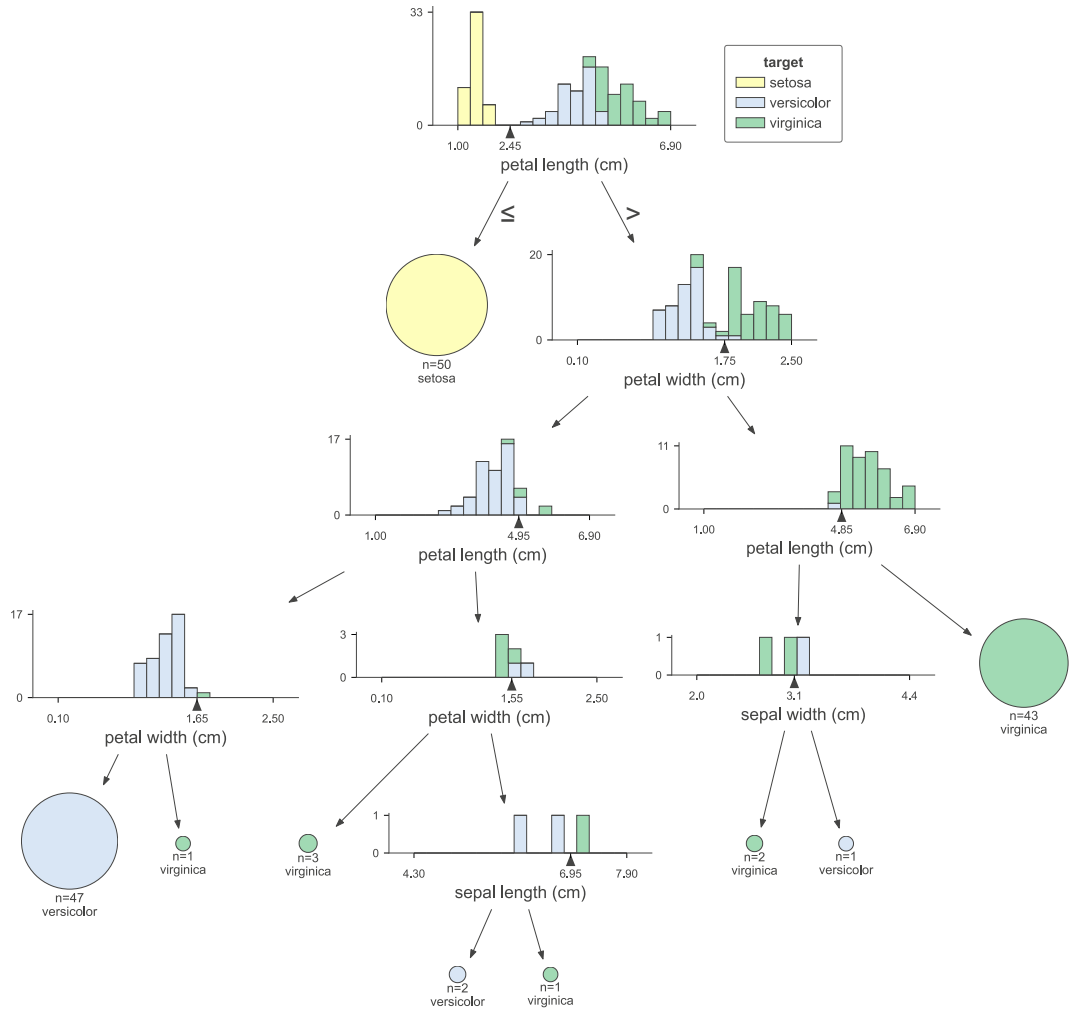

Figure 2.2: Decision tree visualization.

Greedy rule search

Greedy algorithms are a family of optimization algorithms that search for locally optimal choices at each step in order to find a globally optimal solution. Greedily rule search splits on one feature at a time along a single path. It tries to find rules which maximize the probability of the majority class.

RuleFit

RuleFit algorithm learns sparse linear models that include automatically detected relations between input variables in the form of decision rules. The learning procedure consists of two components. The first step is rules generation. Gradient boosting is used to fit an ensemble of decision trees. Each tree in the ensemble is then converted into multiple rules. *RuleFit* learns trees of different depths so that many diverse rules are generated. Then, in the second step, the linear model is fitted. Not only rules are used, but also features from the original dataset are used in the linear model. The result is a linear model that has weights for all of the original features and for the rules. [44], [45]

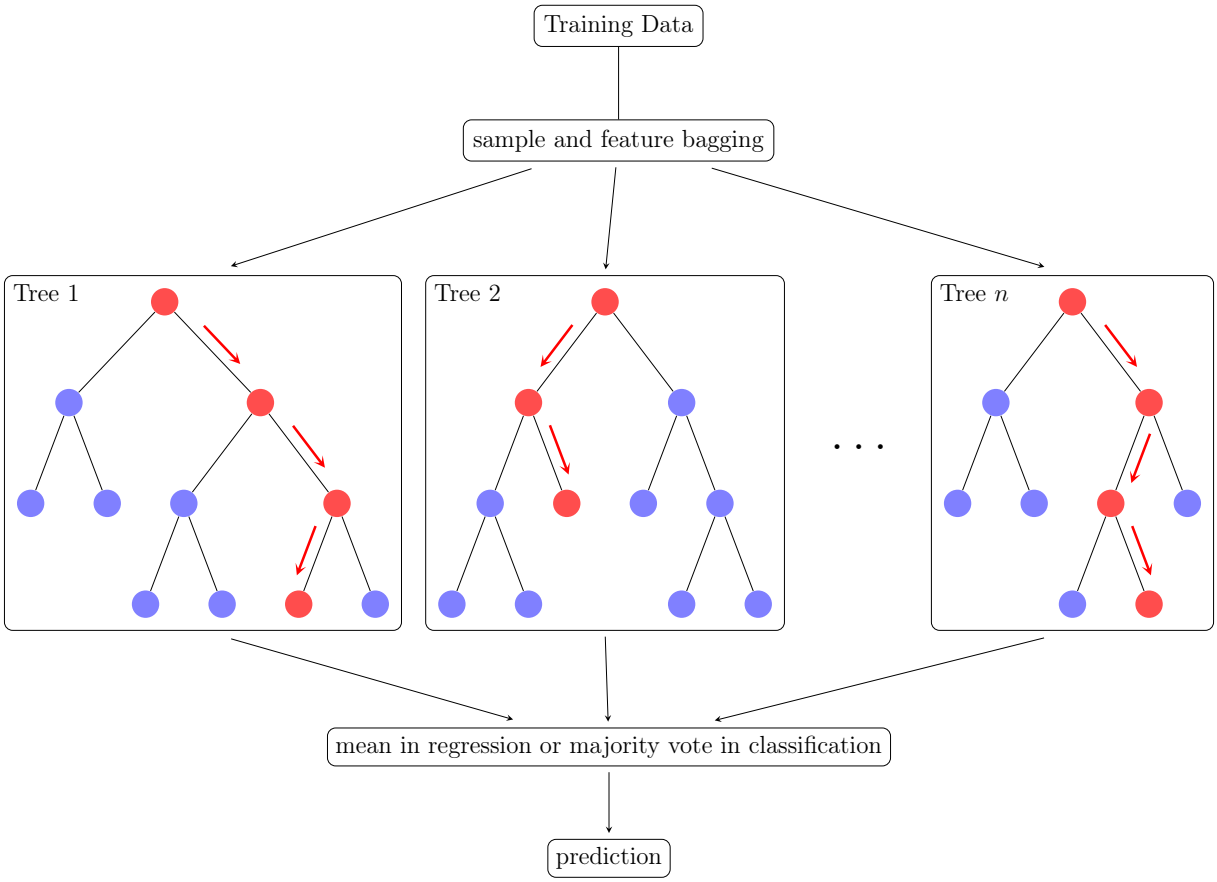

Figure 2.3: RF of n trees visualization.

2.3 Data complexity

Essentially, most real-life classification problems find their sources in processes, which can be described by the physical or behavioural models. Even though such sources might have some non-deterministic random element within them, the distributions of features or labels have some consequential structure which differentiates such sources from random labelling. The latter is, at its core, a difficult problem because assigning labels randomly to data points makes it impossible to find any valuable patterns.

The difficulty of the problem could be the result of various factors. Label distribution may be intrinsically ambiguous or due to feature measurements of not sufficient quality. The dataset might have been underrepresented by the lack of a number of samples describing the concept. Some concepts might have complex and multidimensional decision boundaries, making it impossible to find a compact description of the boundary. Datasets might have non-zero Bayes errors, making it impossible to classify concepts present within them accurately. A sampling of the problem might have been performed in a way that failed to describe the complexity of class distribution. In many complex real-life processes, high-quality gathering samples might be problematic. It may apply to the whole described phenomena in general, as well as to only a subset of classes, making the label distribution imbalanced. Overall, real-world problems usually contain a mixture of such difficulties.

With a sample completely describing the problem, the decision boundary could be described by Kolmogorov complexity [46] or the minimum length of a computer algorithm needed to reproduce it [47]. In this principle, a problem is complex if it takes a long

algorithm to describe the class boundaries. Given the knowledge that Kolgomorov complexity is algorithmically uncomputable [48], other metrics describing data complexity were created. They will be described in the following sections. [49]

The problem of estimating dataset complexity has been considered in literature widely in the area of meta-learning. It is part of artificial intelligence that studies how the learning system can become flexible enough to adapt itself to a specific environment. It differs from *base-learning* in sense that in the latter the bias is fixed *apriori*, where with *meta-learning* the learner has enough knowledge to chose bias dynamically. In a typical learning scenario, bias is fixed as a result of choosing a specific model where it's embedded. The meta approach aims to discover ways to dynamically search and apply the most suitable model and learning strategy based on the specific domain it operates in. Thus, learning takes place not only at the instance level but also across-datasets (meta) level [50].

Building a system that uses a meta-learning approach utilises so-called meta-features. Those are measurable characteristics of datasets, which could form patterns that will eventually predict specific learners' performance in a given problem domain. Those features vary from as simple as dataset size to more complex ones containing structural info of features or width of decision tree induced using the instances. The key component behind such features is that they should be fairly easy to compute, so they may be utilised as predictors of specific learner accuracy instead of the accuracy itself [51].

Literature on the subject matter provides a consistent approach to assigning meta-learning features into categories. Lorena et al. [52] deals with complexity measures split into a feature, linearity, dimensionality, neighbourhood and class balance based. Such approach is persistent [49], [51], being extended upon with simple statistical measures (like quantiles or number of instances), model-based features (height of tree or number of leaves), land-marking features (which use some simple, usually fast-trained model to set some based level) and structural info measures (where two feature vectors are being generated from dataset and their statistics are employed as meta-features).

In the following sections, the data complexity categories will be expanded, and specific measures' definitions will be presented. As mentioned before, the scope of this doctoral thesis covers the usage of linearity metrics and thus, more focus will be put on describing those.

For further description, n will serve as a number of instances for which the complexity metric is being calculated.

Linearity based measures

All following measures, to some extent, check whether the classification problem is linearly separable.

L1. Sum of the error distance by linear programming The sum of the distances of incorrectly classified examples to a linear boundary used in their classification is computed. If the value is zero, then the problem is considered to be linearly separable and thus may be considered simpler.

Having the SVM hyperplane, error distance can be computed by summing up the ϵ_i values. For instances correctly classified with a margin larger than 1, ϵ_i, otherwise it will take the value of distance to the linear boundary. It is described by SVM optimization

process:

$$\text{SumErrorDist} = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i | \Psi(x_i) \neq y_i \tag{2.14}$$

Then, the L1 value can be computed as:

$$L1 = \frac{\text{SumErrorDist}}{1 + \text{SumErrorDist}} \tag{2.15}$$

L1 does not allow the checking of whether one linearly separable problem is simpler than another one with the same property. The dataset for which data are distributed narrowly along the linear boundaries will have a $L1 = 0$ value, but so will the datasets where classes are apart with a large margins. The complexity computation cost of this measure is dependent on that of linear Support Vector Classifier (SVC), and can take $O(n^2)$ in the worst case ([53].

L2. Error rate of linear classifier The L2 measures complexity by calculating the error rate of the linear SVM classifier. When Ψ denotes the obtained linear classifier, the metric is given by:

$$L2 = \frac{\sum_{i=1}^{n} I(\Psi(x_i) \neq y_i)}{n} \tag{2.16}$$

A higher L2 value indicates more errors and, therefore, great complexity. When it comes to problems that are linearly separable, it has a sample issue as L1 - it is unable to tell which problem is more complex (barely separable - with a narrow margin) and which is far apart. The computation complexity is the same as in L1 - $O(n^2)$

L3. Non-linearity of linear classifier This metrics computation is based on generation of synthetic samples. Randomly chosen pairs of training examples of the same class are interpolated, and then, using a random coefficient, a new sample is created along the interpolation line. Then, a linear classifier is trained on the original data, and its error is measured on synthetic data points. Thanks to such an approach, the metric is sensitive to how the data from the class is distributed in the border regions and how much the classes overlap. When Ψ denotes the linear classifier trained using original training data the L3 measure may be expressed by:

$$L3 = \frac{1}{l}\sum_{i=1}^{l} I(\Psi(x_i^{'}) \neq y_i^{'}) \tag{2.17}$$

where l is count of interpolated examples $x_i^{'}$ and their corresponding labels are denoted by $y_i^{'}$. The asymptotic complexity cost of this measure is dependent on both the induction of a linear SVM and the time taken to obtain the predictions for the l test examples, resulting in $O(n^2 + mln_c)$.

Neighborhood metrics

These metrics' aim is to capture the shape of the decision boundary and quantify class overlap by analyzing local neighbourhoods. All of them work by utilizing a distance matrix of every point pair in dataset. Gower distance measure [54] is utilised to cover both numerical and symbolic features.

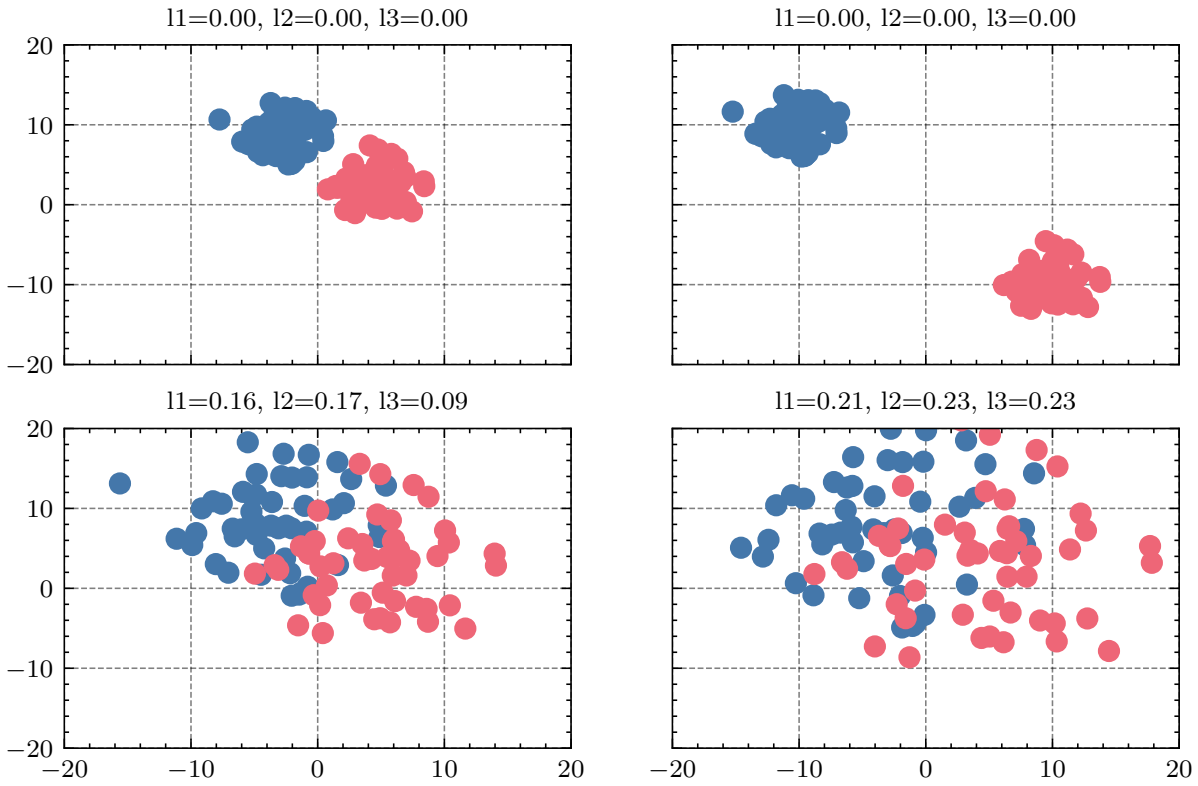

Figure 2.4: Example of L1 and L2 metric values. As can be seen, those metrics do not take into consideration the complexity of linearly separable datasets.

N1. Fraction of borderline points Minimum Spanning Tree (MST) is built from data, where edges are weighted according to the distance between data points. The metric is calculated by computing the percentage of vertices incident to edges connecting examples of opposite classes. N1 estimates the size and complexity of the required decision boundary through the identification of critical points in the dataset. Higher values indicate the need for more complex boundaries to separate classes or that there is a large amount of overlap between classes.

$$N1 = \frac{1}{n}\sum_{i=1}^{n} I((x_i, x_j) \in MST \wedge m_i \neq y_j) \tag{2.18}$$

To compute the graph from the dataset, it is required to first compute the distance matrix, which is $O(mn^2)$, and then create MST which, using *Prim's algorithm* [55] has $O(n^2)$ worst-case complexity.

N2. Ratio of intra/extra class Nearest-Neighbours (NN) distance This metric computes the ratio of two sums: the sum of distances between all pairs of examples and their closest neighbour from the same class and the sum of distances between each example and its closest neighbour from another class. N2 is expressed as:

$$N2 = 1 - \frac{1}{\text{intraExtra}} \tag{2.19}$$

where

$$\text{intraExtra} = \frac{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_i)}{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_j \neq y_i} \tag{2.20}$$

and $d(x_i, NN(x_i) \in y_i)$ is distance to the example x_i and its nearest neighbor from its own class y_i and $d(x_i, NN(x_i) \in y_j \neq y_i)$ is the distance to closes neighbor from another class. Computation of this metric again requires obtaining distance matrix $O(mn^2)$. Low values indicate simpler problems in which the overall distance between examples of different classes exceeds the distance between examples of the same class. As indicated, this metric is sensitive to how data are distributed within classes, not only to what the boundary between classes is like.

N3. Error rate of NN classifier Simply put, this metric calculates the error rate of 1-NN classifier using the leave-one-out procedure:

$$N3 = \frac{\sum_{i=1}^{n} I(NN(x_i) \neq y_i)}{n} \tag{2.21}$$

where $NN(x_i)$ denotes NN classifier's prediction for sample x_i. High $N3$ values indicate that many examples are close to examples of other classes, making the problem more complex. It complexity is $O(mn^2)$.

N4. Non-linearity of NN classifier Similar to L3 2.3 but uses NN classifier instead:

$$N4 = \frac{1}{l} \sum_{i+1}^{l} I(NN_T(x_i^{'}) \neq y_i^{'}) \tag{2.22}$$

where l is a number of interpolated points. In contrast to L3, this metric can be applied directly to multilabel classification problems without needing to decompose them (for example, using OVA or One-Versus-One (OVO) [56]) into binary sub-problems. The computation complexity of this metric is $O(mnl)$.

T1. Fraction of hyperspheres covering data The hypersphere is built and centred at each one of the samples. The radius of hyperspere is iteratively increased until it reaches example of another class. Smaller hyperspheres contained in bigger ones are eliminated. Then, T1 is defined as the ratio of remaining hyperspheres to the total number of examples:

$$T1 = \frac{\text{no. of hyperspeheres}}{n} \tag{2.23}$$

LSC. Local set average cardinality The local set of an example x_i in dataset T is defined as a set of points from T whose distance to x_i is smaller than the distance from x_i to the nearest member of different class [57].

$$LS(x_i) = \{x_j | d(x_i, x_j) < d(x_i, ne(x_i))\} \tag{2.24}$$

$ne(x_i)$ being the nearest member of other class. A big count of low-cardinality local sets in the dataset indicates that the space between classes is narrow and irregular. In other words, the boundary is more complex. The local set average cardinality measure (LSC) is calculated here as:

$$LSC = 1 - \frac{1}{n^2} \sum_{i=1}^{n} |LS(x_i)| \tag{2.25}$$

where $|LS(x_i)|$ is the cardinality of the local set for a given example. The complexity cost of LSC is related to computation of pairwise distances between all samples, being $O(mn^2)$.

Network measures

Network-based measures model the dataset as a graph and extract the statistical characteristics of network complexity. The base graph must preserve the similarities or distances between examples for modelling the data relationships. Each example from the dataset should correspond to a node or vertex of the graph, whilst undirected edges connect pairs of examples and are weighted by the distances between the examples.

As in the neighbourhood measures, Gower distance is utilised. Two nodes i and j are connected if $dist(i, j) < \epsilon$. For this work purposes, ϵ was set to 0.15. The post-processing step is applied to the graph, pruning edges between examples of different labels. A more detailed description of the graph building process can be found in [52] and [58].

Average density of the networks Number of edges that are retained in the graph created from the dataset normalized by the maximum number of edges between n pairs of data points:

$$\text{Density} = 1 - \frac{2|E|}{n(n-1)} \tag{2.26}$$

For datasets with dense regions for the same classes, lower metric value will be obtained, translating to lower complexity. When there is a low number of edges observed for a given dataset of low density, the metric will raise a higher value.

Clustering coefficient The clustering coefficient is defined as the root of number of edges between neighbors and the maximum number of edges that could possibly exist between them:

$$\text{ClsCoef} = 1 - \frac{1}{n}\sum_{i=1}^{n}\frac{2|e_{jk} : v_j, v_k \in N_i|}{k_i(k_i-1)} \tag{2.27}$$

where N_i denotes neighborhood set of vertex v_i and k_i is size of N_i. The Clustering coefficient assesses the grouping tendency of vertexes by monitoring how close to forming a clique vertexes are.

Hub score Hub score assigns each node the number of connections it has to other nodes, which is then weighted by the number of connections these neighbors have. Highly connected vertexes, which are also densely connected, will raise higher hub scores.

$$\text{Hubs} = 1 - \frac{1}{n}\sum_{i=1}^{n}\text{hub}(V_i) \tag{2.28}$$

The values of $\text{hub}(V_i)$ are given by the principal eigenvector of A^tA, where A is the adjacency matrix of the graph. In datasets in which there is high overlapping of the classes, strong vertexes tend to be less connected to neighbours. Therefore smaller values will be raised for simpler datasets.

Feature based metrics

F1 F1 calculates the ratio of overlap between values of features in different classes.

$$F1 = \frac{1}{1 + \max_{i=1}^{m} r_{f_i}} \tag{2.29}$$

where r_{f_i} is discriminant ratio for feature f_i.

F2 F2 describes the overlap of classes' feature values distribution. It is calculated by first finding the minimum and maximum feature value for each class and then finding the range of overlapping normalized intervals.

$$F2 = \prod_{i}^{m} \frac{\text{overlap}(f_i)}{\text{range}(f_i)} = \prod_{i}^{m} \frac{\max\{0, \text{minmax}(f_i) - \text{maxmin}(f_i)\}}{\text{maxmax}(f_i) - \text{minmin} f_i} \tag{2.30}$$

where:

$$\begin{aligned}
\text{minmax}(f_i) &= \min(\max(f_i^{c_1}), \max(f_i^{c_2})) \\
\text{maxmin}(f_i) &= \max(\min(f_i^{c_1}), \min(f_i^{c_2})) \\
\text{maxmax}(f_i) &= \max(\max(f_i^{c_1}), \max(f_i^{c_2})) \\
\text{minmin}(f_i) &= \min(\min(f_i^{c_1}), \min(f_i^{c_2}))
\end{aligned}$$

The values $\max(f_i^{c_j})$ and $\min(f_i^{c_j})$ are maximum and minimum values of each feature in class c_j. The asymptotic cost of the measure is $O(mnn_c)$. The higher the F2, the greater the overlap between problem classes. If there is at least one non-overlapping feature, the F2 value will be zero.

F3. Maximum individual feature efficiency This metric estimates the individual efficiency of each feature in separating the problem classes and considers the maximum value found among them. Every feature is checked for the overlap between examples of different classes. If there is overlap, the classes are considered to be ambiguous in given region. The problem is considered simpler if at least one feature shows low ambiguity between classes:

$$F3 = min_{i=1}^{m} \frac{n_o(f_i)}{n} \tag{2.31}$$

where $n_o(f_i)$ gives number of examples that are in the overlapping region for feature f_i. Small values of metric are raised when few examples overlap in at least one dimension. Complexity of the metric is $O(mnn_c)$.

$$n_o(f_i) = \sum_{j=1}^{n} I(x_{ij} > \text{maxmin}(f_i) \wedge x_{ji} < \text{minmax}(f_i)) \tag{2.32}$$

F4. Collective feature efficiency The computation of this metric happens iteratively. First, the most discriminative feature, according to F3, is selected. Next, all examples that could be separated by this feature are removed from the dataset and then the procedure is repeated. The algorithm is applied until all the features have been considered and can also be stopped when no example remains. Then the ratio of examples that have not been discriminated. F4 is computed after l rounds are performed through the dataset.

$$F4 = \frac{n_0(f_{\min}(T_l))}{n} \tag{2.33}$$

where $n_0(f_{\min}(T_l)$ measures number of points in the overlapping region of feature f_{min} for the dataset in the l-th round of iteration. Then, the most discriminative feature in dataset T can be found using:

$$f_{min}(T_i) = \{f_i | min_{j=1}^{m}(n_o(f_j))\} T_i \tag{2.34}$$

F4 uses the F3 procedure multiple times, and at most, it will iterate for all input features, resulting in a worst-case complexity being $O(m2nn_c)$.

Dimensionality measures

Those measures give an idea of dataset sparsity. They are based on the dimensionality - either original or reduced. The idea behind them is that it is harder to build a good predictive model on sparse datasets.

T2. Average number of features per point This measure divides the number of examples in the dataset by their dimensionality.

$$T2 = \frac{d}{n} \tag{2.35}$$

The measure reflects the data sparsity – if there are many features and a small amount of data points, the problem will potentially be harder to model as the samples will be sparsely distributed in input space. It can be computed in $O(d + n)$.

T3. Average number of Principal Component Analysis (PCA) dimensions per point T3 is defined as number of PCA [59] components needed to represent 95% of data variability (d'):

$$T3 = \frac{d'}{n} \tag{2.36}$$

Because T3 requires computing a PCA analysis of the dataset, its worst complexity is $O(d^2n + d^3)$.

T4. Ratio of PCA dimension to the original dimension Related to the previous two measures, this measure gives a rough estimate of the proportion of relevant dimensions for the dataset. It is measured by calculating number of PCA components needed for 95% variability and comparing it to original dimensionality:

$$T4 = \frac{d'}{d} \tag{2.37}$$

The larger the T4 value, the more of the original features are needed to describe data variability. This indicates more complex combination of the data features. The complexity cost of this measure is $O(d^2n + d^3)$.

Class Imbalance Measures

This group of measures considers the dataset imbalance ratio as an aspect that largely influences the potential performance of machine learning models. In general, algorithms tend to favour the majority class [60]

C1. Entropy of class proportions Taking p_c as proportion of examples is each classes, this measure can be expressed as:

$$C1 = 1 + \frac{1}{log(n_{c_i})} \sum_{i=1}^{n_c} p_{c_i} log(p_{c_i}) \tag{2.38}$$

This measure will express a minimum value for balanced problems. The computation complexity for it is $O(n)$.

C2. Imbalance ratio The standard metric for measuring class imbalance. We adopt here versions proposed by [61], adopted for multiclass problems:

$$C2 = 1 - \frac{1}{IR} \tag{2.39}$$

where

$$IR = \frac{n_c - 1}{n_c} \sum_{i=1}^{n_c} \frac{n_c}{n - n_{c_i}} \tag{2.40}$$

and n_{c_i} is number of instances for i-th class. These values can be computed in $O(n)$ time.

2.4 Explaining behavior of classification models

2.4.1 Ambiguity of terms in Explainable AI

Before diving deeper into the review of XAI related literature, it is worth noting and clarifying certain terms discrepancies existing within the community. As Marcinkevičs and Vogt [62] observe, synonymous terms are interchangeably used by scientists and industry. Some authors [63] define interpretability as the ability to explain or present a solution in understandable terms to humans. Lipton [64] notes, that many definitions used in existing literature lack mathematical rigour and precision. Additionally, many papers vaguely define motivations for interpretability, taking the definition of the term as granted. Rudin and Huysmans[25], [65] point out, that interpretability is domain-specific, therefore creating all-purpose definition is unnecessary and unfeasible. Rather, the interpretability is dependent on many external factors, such as target user experience, prior knowledge, and application domain. Authors also notice that the past research on interpretable machine learning does not necessarily use terms like "interpretability" or "explainability". For example, the original paper on inducing decision trees using ID3 algorithm does not use any of the aforementioned [40]. Instead, it makes use of the term "intelligibility" [25].

Marcus et al. [66] point out that heterogeneous and ill-defined terminology in the field of XAI makes the efficient search of papers impossible, therefore rendering research efforts incapacitated. Let's take a look at each of the used terms and try to figure out definitions which will be followed for the rest of this thesis. Although this is out of the scope of this thesis, for completeness, it is worth mentioning that the whole scientific area on the crossroads of linguistics and philosophy exists considering questions such as "what is explanation" in lingual aspect [67]. Originally, the term XAI was coined by Van Lent et al. 2004 paper [68] to characterize system capacity to explain actions of AI in gaming applications. Oxford Dictionary gives the following definition of the word "interpret": "explain or understand (behaviour etc.) in a specified manner" [69].

For the rest of the thesis, the following is used to define what we understand as explainability. To reach it, both fidelity and interpretability are necessary [70]. The first defines how faithful the explanation is to the model being explained, while the latter states how understandable to the human receiver is the explanation. [17] also notes how important it is to have a faithful explanation by stating that the description of the model should be at least locally faithful in order to be meaningful at all.

Interpretability will be defined as the property of explanation [16]. The explanation can be the result of the prediction of the inherently interpretable model or so-called *post-hoc* explanation accompanying existing AI model. We will dive further into the taxonomies

of interpretable methods in section (2.4.2). Taking the above into consideration, we could say that the AI system is interpretable if its model is inherently interpretable or it is supported with another model that provides interpretable and faithful explanations.

Although pondering over definitions will not be the main subject of this thesis, certain clarification should be in place for consistency. For the rest of this thesis, the following definitions are accepted, followed after Arrieta et al.[16]:

- **Understandability/intelligibility** - property of model to make humans understand how it works, without the need for explaining its inner algorithms
- **Comprehensibility** - ability of learning algorithm to represent its learned knowledge in a way understandable to human [71]
- **Interpretability** - ability to explain or provide meaning in understandable to human terms
- **Explainability** - associated with explanation, defined as interface between human and decision maker that is, equally accurate proxy (or, as Rudin [25] argued, approximation) of decision maker (model) and comprehensible to human receiver
- **Transparency** - based on the previous definition, it can be said that the model is transparent if it is understandable.

2.4.2 XAI taxonomies

There are several taxonomies proposed in the XAI field [21], [72], [73] for classifying such methods. Amongst them, three repeating categorizations can be noticed:

- *post-hoc* / *intristic* – wherever the interpretability was achieved as an internal part of model creation/training, or after it,
- *model specific* / *agnostic* – wherever the explaining method is applicable only to a certain family of models, such as Deep Neural Networks or ensembles,
- *global* / *local* explanations – global, if the methods explain the whole model or only given part of it, such as model behaviour given a particular sample.

One of the most cited and well-acclaimed works when it comes to model agnostic explanations is LIME [17]. This explanation technique utilizes local interpretable model estimation in order to provide insight into specific classification event. It can be applied to a wide spectrum of areas, including simple tabular data, text classification and image classification models. Since its introduction, many other application-specific methods have been researched based on it, in domains such as fake news detection [74] or financial risk assessment [75]. Another method uses tools coming from game theory. Lundberg and Lee [18] introduce SHAP, which makes use of *Shapley value* in order to provide local explanations via decomposing the output of a model by the sums of the impact of each feature. It was well acclaimed across the data science community and is present as a tool in many libraries. What is worth mentioning is that a whole spectrum for explaining deep neural networks exists. Among them can be found numerous methods for explaining convolutional neural networks used for image classification [76]–[78]. Usually, they provide an explanation via heat-map showing, based on which area the model made a decision. Partial dependency plots are often used as tool for introspecting feature importance learned by model [79]. Sensitivity analysis [80] is another tool very often used, which over the years was utilised to expose weaknesses of deep neural network models [81].

2.4.3 How to quantify explanation?

Since the nature of the task of explanation is very subjective, it is difficult to compose or find that would apply to a wide range of XAI research while being objective. Explainable systems are usually designed with the user in mind, which poses a thread of subjectivity and randomness. Trust, satisfaction from explanation, its goodness and understanding are generally "soft" values, hard to measure [82]. Rosenfeld [83] proposed four metrics – D,R,F,S – for measuring the explainability of human-agent systems. They are not dependent on the task performed or the algorithm itself. D quantifies the change in the agent's performance that occurred thanks to introducing explanations into the system. R measures the complexity of rules in the system, which inherently measures explainable model complexity. F measures the number of features provided by the agent to the system in order to create a viable explanation, and finally, S measures system stability when it performs under perturbed, randomized input. Authors of [84] make a point that it is common for XAI research paper to measure their methods in ways only applicable to specific types of approaches or only to the proposed method itself. It is not uncommon that explainable systems are being tested using human-generated opinions only for the given system and no other [85]. Sovrano et al. [84] propose an objective method of measuring explanation based on linguistic sets details of how explanation answers archetypal questions (how? why? what?). In detail, it requires XAI method to be able to provide a textual explanation behind some reasoning, which is then reasoned using embeddings and language models, which is also restricting and, to some degree, model-specific.

When it comes to rule-based models, which are the subject of the thesis, a couple of model-specific measures are used across literature [86], namely:

- Completeness, defined as the ratio of instances covered by rules to the total number of instances
- Correctness, defined as the ratio of instances correctly classified by rules over total number of instances
- Fidelity, defined as accuracy to explained model
- Robustness, defined as resistance to small perturbations – how explainer is able to withstand small changes to input without changing its prediction
- Total number of rules
- Average rule length

Islam et al. [87] concluded that, as purely explainability-related metrics are not yet established within science, in most of the research, state-of-the-art quality measures are used, such as accuracy or recall. Additionally, [88] et al. identified over 60 different metric notions used across papers in XAI domain. They also note that it is not uncommon to rediscover the same metrics in different papers, making it even harder to enforce common ground.

2.5 Genetic algorithms

In the following sections, an optimization problem will be raised as a way of building a transparent model. Having this said, Evolutionary Algorithms (EA) will be utilized to

find feasible solutions. In this section Genetic Algorithm (GA) will be described as part of aforementioned family of meta-heuristic. It should be noted that in this thesis, those methods will be used just as part of the toolbox created with the intention of solving an optimization problem. Therefore, fine-tuning and diving deeper into properties of EA is out of the scope of this thesis.

GA are a family of exhaustive search-based methods established on the biology-originating concepts of natural selection. GA include a group of different optimization techniques inspired by genetic recombination processes such as mutation, crossover and selection. Such solutions are evaluated using fitness functions, and the more appropriate individuals have a greater probability of moving on to the next generation. In its classical way, the procedure goes as follows. A population of chromosomes is randomly initialized. Next, the value of the fitness function of each chromosome is computed. Two chromosomes are *selected* from the population according to fitness value and then, the *crossover* operator – with some probability – is applied to produce offspring. Thereafter, *mutation* operator is applied to produce offspring again, with certain probability, to produce new offspring. The offspring returned after mutation is then placed in a new population, and the whole procedure is repeated until the new population is complete [89].

In the aforementioned process, three distinct operations can be seen. Namely selection, crossover and mutation. Scientific development for all of these is actively ongoing. Additionally, there are many variants of GA itself, which can be selected according to applied domain, such as real-coded, parallel or multi-objective variations [90].

2.6 Evaluation methodology

In order to find out whether the proposed method is applicable to a wide range of problems, evaluate its generalization abilities and estimate overall performance, a tailored experimentation protocol is required. In this section, such protocol will be proposed and reasoned about.

Across scientific literature, one of the most frequently acclaimed procedures verifying whether the model will perform well across different problems is Cross Validation (CV) [91]. Performing a task of evaluating an algorithm, an experimentator is typically present with a certain amount of learning sets $\mathcal{WS}$, such as *Wisconsin breast cancer*[1] one. Said set is then randomly sampled in order to generate training sets $\mathcal{LS}$ and testing set $\mathcal{TS}$, in a way that guarantees $\mathcal{LS} \cup \mathcal{TS} = \emptyset$ (training and testing sets are disjoint).

In CV, the learning set is split into many disjoint training sets and validation sets $\mathcal{VS}$. The term validation set is typically used within the context of CV, while test set commonly refers to the part of data that is put aside for a final model evaluation (hold-out test set). Widely adopted and known k-fold CV procedure randomly splits learning dataset into k equally sized folds k times, from which $\frac{k-1}{k}$ is used for training, and one is used for validation. Another variation is nested k-fold CV. In, for example, the case of 5-times nested 2-fold CV, the learning dataset is split into two equally-sized parts randomly five times, giving 10 learning iterations in total. Metrics raised by such procedure could then be averaged, to obtain unbiased and low-variance estimation of tested models' accuracy. Picking a number of repetitions and a number of folds (k) is a non-trivial task and was broadly studied within the literature. [92], [93].

Depending on the sample size [94] Wong et al. suggest different folds and repeat counts. 10-fold or 5x2 are widely considered. Since datasets selected for the experiments

[1] `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`

Table 2.1: Selected binary dataset details with number of instances, features and their types.

name	#instances	#features	#integer	#real	#nominal
appendicitis	106	7	0	7	0
australian	690	14	5	3	6
bands	365	19	6	13	0
breast	277	9	0	0	9
bupa	345	6	5	1	0
crx	653	15	3	3	9
haberman	306	3	3	0	0
heart	270	13	12	1	0
hepatitis	80	19	17	2	0
housevotes	232	16	0	0	16
ionosphere	351	33	1	32	0
mammographic	830	5	5	0	0
monk-2	432	6	6	0	0
saheart	462	9	3	5	1
tic-tac-toe	958	9	0	0	9
wisconsin	683	9	9	0	0

in the following chapters are of various sample counts and in order to unify the evaluation procedure, the methodology suggested in well acclaimed [93] is picked - 5 repetitions with 2 folds of equal size, replacing test with train in subsequent iterations.

Statistical significance In order to provide statistical insight in the results, statistical tests are performed to check whether the difference is significant. The methodology proposed by [95] will be followed, which uses Wilcoxon Signed-Rank test to check whether performance, validated over multiple datasets with multiple repetitions, is significant or random. Wilcoxon Signed-Rank test is non-parametric statistical test and an alternative to paired t-test. After performing metric measurements (such as accuracy) of two classifiers on batch of datasets, absolute differences of values in those measurements are considered and ranked for positive and negative scores. In the case of ties, average ranks are assigned. T statistic is calculated by taking sum of ranks where the second algorithm was better (R^+) and worse (R^-) and taking minimal value of the two. Critical values can then be read from tables, or, for larger number of datasets, z statistic can be calculated:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.41}$$

which is normally distributed and with $\alpha = 0.05$ the null-hypothesis can be rejected if z is smaller than -1.96.

In the Wilcoxon Signed-Rank test greater differences count more, which is probably desired, but the absolute magnitudes are ignored. The test is safer since from the statistical point of view it does not assume normal distributions. Additionally, the outliers have less effect on the value than on the t-test.

Additionally, Sign test is utilized to validate the performance of classifiers by comparing a number of wins, draws and losses [96]. If the two algorithms compared are,

Table 2.2: Dataset folds were assigned multiple labels according to quantile-based discretization. For each metric, each fold was assigned either *low*, *medium* or *high* value. Boundaries which set the quantiles are presented in this table.

metric	low	low/medium	mediun/high	high
F1	0.017	0.49	0.757	0.982
F2	0.0	0.215	0.659	1.0
F3	0.225	0.824	0.948	1.0
F4	0.0	0.629	0.896	1.0
F1V	0.006	0.145	0.417	0.91
C1	0.001	0.009	0.069	0.39
C2	0.001	0.024	0.172	0.658
L1	0.0	0.179	0.266	0.487
L2	0.0	0.194	0.297	0.764
L3	0.0	0.14	0.301	0.604
N1	0.034	0.111	0.195	0.269
N2	0.373	0.493	0.54	0.732
N3	0.052	0.219	0.377	0.515
N4	0.013	0.155	0.25	0.377
T1	0.264	0.588	0.822	0.956
T2	0.012	0.035	0.096	0.475
T3	0.002	0.022	0.043	0.126
T4	0.2	0.421	0.667	1.0
clsCoef	0.013	0.081	0.368	1.0
hubs	0.245	0.505	0.738	0.946
density	0.402	0.718	0.879	0.981
LSC	0.698	0.96	0.986	0.994
size	80.0	306.0	462.0	958.0
features_count	3.0	9.0	14.0	33.0

as assumed under the null hypothesis, equivalent, each should win on approximately $\frac{N}{2}$ out of N datasets. The number of wins will be distributed according to the binomial distribution. For an arbitrary number of datasets, the number of wins is under the null hypothesis distributed according to $N(\frac{N}{2}, \sqrt{\frac{N}{2}})$, which allows for the use of z-test: if the number of wins is at least $\frac{N}{2} + 1.96\sqrt{\frac{N}{2}}$ then the tested classifier is significantly better with $p < 0.05$. Ties are split evenly between the two classifiers. If there is an odd number of them, one will be ignored.

Datasets selection Standard Knowledge Extraction based on Evolutionary Learning classification datasets repository was used as source of benchmark datasets [97]. It provides a set of widely and commonly used real-world datasets. Out of those, 16 were picked in a way that would provide the widest spectrum of measured complexity metrics. It was done by calculating each interesting complexity metric and picking subsequently maximum and minimum value datasets for given metrics. Description of the picked datasets can be found in Table 2.1.

Complexity discretization In order to be able to raise conclusions about how a given

algorithm performs under certain dataset properties, complexity metrics were computed for every training fold. Complexity periods were calculated as follows. First, every complexity metric was calculated for every used dataset training fold. Then, those complexities were discretized using quantile-based approach into equal-sized buckets. Three discrete values were used: *low*, *medium*, *high*. Such quantization resulted in each experiment training instance having assigned multiple discrete values equal to the amount of calculated complexity metrics. To get a final estimation of whether the data fold is complex or not, the mode of those discrete values was taken. In the case of draws, a higher complexity value was used. This resulted in every experiment having one complexity label assigned. Specific, after discretization bin values can be found in Table 2.2.

Chapter 3

Explaining tree based ensemble models

3.1 Introduction and related works

In section (2.4) XAI segment of AI was introduced. In this chapter, emphasis will be put on a subset of XAI focused on explaining classifier ensembles.

In terms of interpretability and explainability, the downside of ensemble models is that they usually require some kind of integration step, which accumulates results from all base learners. This step, even as simple as majority voting or selection of simple best learner, often introduces a certain degree of incomprehensibility, in result rendering the whole model less interpretable. XAI methods seek a remedy to this problem.

Several methods were proposed for explaining ensemble models. Sagi and Rokach [19] proposed a framework for constructing a single *decision tree* using rule conjunctions extracted from *decision forest*. The conjunction of all possible rules is created, and then such a set is organised into a single tree structure. The proposed *Forest Based Tree* was often able to provide equally well-performing explanations of RF with deeper rules. The problem of high computational complexity was also encountered and recognised by authors, who then proposed some heuristics to tackle it. The proposed output model is a single decision tree. Deng [98] proposed a method called *InTrees* for extracting, processing and filtering rules from the tree ensemble and, as a result, creating an interpretable model based on an ordered rule list. The extraction process is based on the frequency of variable interactions in the trees, forming an ensemble as a quality measure. The frequency is defined as the proportion of data instances satisfying a given rule. The notion of rule complexity is also introduced, which is equal to the number of predicates in a single rule. Extracted rules are pruned using the relative increase of error after removing the rule from the set (called *decay*). Next, since the outcome rule set could be big, a rule selection process is undertaken where more complex (longer) rules are penalised. Finally, rules are summarised into a single tree using *Simplified tree ensemble learner (STEL)*, which exhaustively removes the best rules and covered by them instances from the set while creating the final tree. Zhou and Hooker [99] operate within the medical domain, where a RF was built to classify responses in diagnosis questionnaires. In the approach, the explained model acts as an oracle which can be queried during the construction of the resulting interpretable tree. The method focuses on ensuring that the number and quality of samples are obtained from the oracle and that the variability due to random selection of such samples is removed. Then the CART like algorithm is used on such sample instances to construct the stable tree. A statistical test is developed to compare splits made by the Gini criterion-based tree induction algorithm for the probability that the split would be the same for another randomly drawn set of samples. Such a test assures that the estimated

tree is stable while being as accurate as the oracle. The pseudogenerated samples used for learning the final tree are generated within specific split subspaces and provided labels by the explained RF. GENESIM [100] uses GA (2.5) to transform an ensemble into single decision tree. Initially, trees are extracted from the ensemble and fetched as individuals into GA. In order to merge two decision trees within the genetic algorithm recombination step, each tree is represented as a k-dimensional hyperplane. Then, the hyperplanes are intersected using a sweep line approach, which in return is converted into a single decision tree using various heuristics. Additionally, two mutations are implemented: choosing a random node in the tree, replacing its split value with a random number and swapping two random subtrees. Results were compared with different decision tree induction algorithms as well as two tree ensemble learners.

Tolomei and Silvestri's [101] mechanism is based on transforming a true negative instance into a positively predicted one given an explained random forest. More precisely, one needs to find a mapping that would transform the vector describing the original negative instance into a new one, which would allow the classification of such an instance as positive. The task is defined as the optimisation problem of choosing the best transformation among a set of possible transformations given cost function, which effectively minimises the effort of mapping (distance between old and transformed vector). In contrast to previously described ensemble explanation methods, this one does not build a whole new surrogate model but provides insight into the features' importance. Namely, authors, by using ad quality classification as an example, show that the model points out features, which change has the biggest impact on shifting the label from negative to positive. Therefore providing insight into ensemble behaviour.

Bastani et al. [102] propose a method that extracts a decision tree from a black box model and evaluates it using a random forest as an example. A set of axis-aligned Gaussian distributions is fit to training data using the Expectation Maximization algorithm. Then, the so-called greedy decision tree is constructed using active sampling, which iteratively finds decision node splits using weighted Gini impurity. Among the reviewed papers, only the last one provided an actual explanation of quality evaluation using human subjects. Apart from that, it compared the output interpretable model (decision tree) among other models of the same type but induced by different learning algorithms. Apart from that, persistent metrics among algorithm evaluation were accuracy and fidelity.

3.2 Explaining tree ensemble using graph modelling

In this section, a novel framework for simplifying tree-based ensembles is presented. Unlike methods described in section (3.1), it does not completely break up with the ensemble approach in order to gain explainability. Instead, the guarantee of high-quality classification performance is obtained through the use of local specialisation of individual decision trees in the selected subspaces. The advantage of such an approach is that it is easy to interpret knowledge structure, which could also be interpreted as a single decision tree or rules list and which is a symbolic representation of knowledge while maintaining a relatively simple model for the classifier training.

The combination mechanism used by the classifier ensemble leads to decreased interpretability. Therefore, to extract interpretable knowledge from the classifier ensemble, dividing a feature space into non-overlapping competence areas and assigning an interpretable model to each of them in the form of a Decision Tree (DT) could be fruitful. Therefore, let us divide the feature space $\mathcal{X}$ into k constituents [103]:

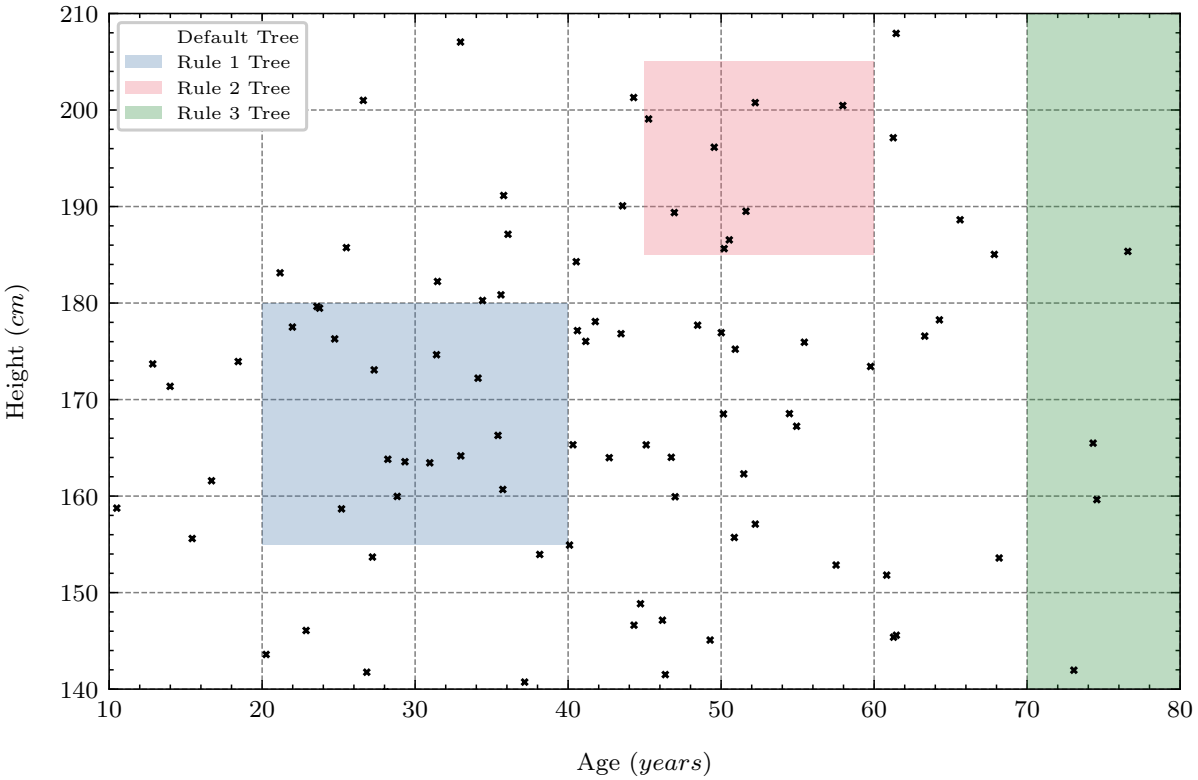

Figure 3.1: Example visualization of model's competence areas. Model consists of 4 trees in total. Each for every rule: ($20 \leq Age \leq 40$, $155 \leq Height \leq 180$), ($45 \leq Age \leq 60$, $185 \leq Height \leq 210$) and ($70 \geq Age$) plus one additional default tree, covering all other feature vectors.

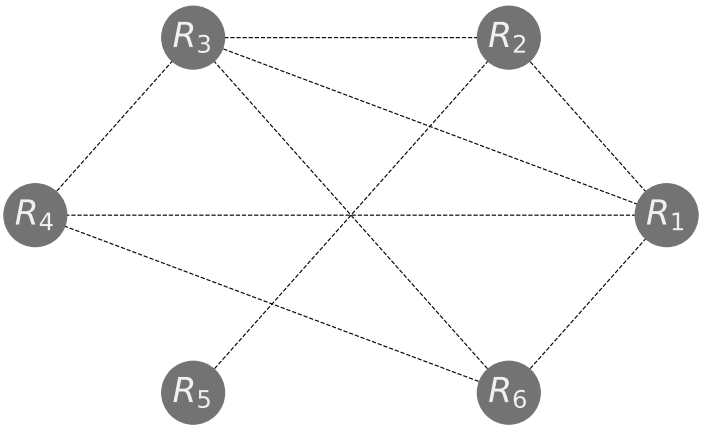

Figure 3.2: Example visualization of graph approach to modelling overlapping relation. Every two rules connected by edge do not overlap. Rules 1, 3, 4, 6 create a clique since they are densely connected. This also means, that they do not overlap one another and could be treated as possible candidates of size 4.

$$\mathcal{X} = \bigcup_{i=1}^{k} S_i \tag{3.1}$$

Furthermore, lets define a set of l input rules $\mathcal{RS}$ and single rule r as tuple consisting of set of c *conditions* (terms *statements* and *predicates* will be used interchangeably) $\mathcal{PS}$ – each of which describes subspace of whole features space – and conclusion label j:

$$\mathcal{RS} = \{r_1, r_2, ..., r_l\} \tag{3.2}$$

$$r_i = (\mathcal{PS}_i, j_i) \tag{3.3}$$

$$\mathcal{PS}_i = \{p_1, p_2, ..., p_c\} \tag{3.4}$$

where p_i is single condition applied over single feature $x^{(l)}$. Condition consists of a feature value $x^{(l)}$ and relation rel:

$$p_i = (x^{(l)}, \text{rel}) \tag{3.5}$$

where $x^{(l)} \in \mathcal{X}$ and rel $\in \{\leq, <, \geq, >, =, \neq\}$. Example of a single rule, induced from the famous titanic survivors dataset[1], is:

$$R_1 = (\{age \leq 25, sex = Female\}, survived \Rightarrow True)$$

It consists of two conditions for *age* and *sex* features and indicates that every Female under the age of 26 has survived.

Rules may be subject to *overlapping* relation. Given the features space $\mathcal{X}$, each predicate p_i designates some subspace $\mathcal{S}_i^p \subset \mathcal{X}$ and in return, so does every rule. Let us define $O_p(p_1, p_2)$ as conditions overlapping relation, and $O(r_1, r_2)$ as rules overlapping. Two predicates overlap if they are defined for the same feature and cover some common part of the subspace. Two rules R_1 and R_2 overlap, if at least one pair of their conditions overlap:

$$\begin{aligned} &\forall(R_1, R_2 \in \mathcal{RS})(\forall(p_1 \in R_1)\forall(p_2 \in R_2)) \\ &\exists(O_p(p_1, p_2)) \Rightarrow O(R_1, R_2) \end{aligned} \tag{3.6}$$

Rules overlapping relation is *reflexive* and *symmetric*. An example of such an overlapping rule pair is:

$$R_1 = (\{age \leq 25, sex = Female\}, survived \Rightarrow True)$$

$$R_2 = (\{age \leq 50\}, survived \Rightarrow False)$$

In the example above, subspace designated by conditions of R_2 covers a bigger part of feature space, which also contains R_1.

As part of the proposed method, a graph representation of rules is being used. Such an approach allows modelling the rules with respect to overlapping relations and, therefore, easily finding sets of non-overlapping rules. If a pair of rules do not overlap each other, then there is an edge between them. The exemplary graph is presented in Figure 3.2. Let us define *clique* of size c in undirected graph $\mathcal{G}$. Such a clique is a complete subgraph of size c in $\mathcal{G}$, in which any two vertices are connected. In addition, *maximal clique* is clique that cannot be extended by adding more connected vertices [104].

[1] `https://www.kaggle.com/c/titanic/data`

3.3 Non-overlapping tree ensemble

NOTE bases its inner workings on a set of rules used to construct non-overlapping competence regions for the ensemble of *decision tree* classifiers.

As the method uses prior knowledge provided by the model in the form of a set of rules created either by an expert or an external algorithm, we can classify it as post-hoc and model-specific. Moreover, the proposed framework provides global explanations, as it may explain any feature vector in $\mathcal{X}$.

The model takes the shape of an ensemble consisting of DTs, each covering a specific competence region designated by a single rule, plus an extra *default tree*, for samples not covered by any of the rules. Thanks to *default tree*, the model allows covering the whole $\mathcal{X}$, and there is no need for utilizing the default class label approach or calculating the distance to the nearest neighbour – which reduces interpretability – as in some of the rule induction algorithms [105].

The learning process is described in Algorithm 1 and possible learning parameters are shown in Table 3.1. The first step is to find all possible rule combinations of size 2, and measure them for *overlapping* (Equation 3.6). This way, every possible rule pair $R_{nm} = (R_n, R_m) \in \mathcal{RS}$ is known for the existence of an *overlapping* relation. The overlapping measurement procedure is fairly straightforward. It consists of iterating over all possible 2-sized combinations and checking for overlapping, as described in section (3.2).

Next, given that we know if every two rules are overlapping or not, candidates for defining the ensemble's competence areas are found. All rules are represented as a graph $\mathcal{G}$, where edges describe non-overlapping relation, and vertices are single rules. An example of such a graph is depicted in Figure 3.2. Let us denote all maximal cliques of rules as $\mathcal{CS}$, and all maximal cliques of size $\leq s$ as $\mathcal{CS}_c$. A single clique may be interpreted as a set of non-overlapping rules. Given the graph and learning parameter s, initially, all maximal cliques $\mathcal{CS}_s$ are considered as possible candidates for the model's competence areas. This allows adjusting the complexity to potential user's requirements and cognitive abilities, which is an important attribute of the explainable model. If no feasible candidates of size s are found, namely $\mathcal{CS}_s = \emptyset$, then the algorithm takes one step backwards and checks $\mathcal{CS}_{c-1}$ and so on until it reaches $\mathcal{CS}_1$, which are single rules.

Checking every possible clique, instead of only maximal ones, would be computationally infeasible. Experiments performed on smaller datasets have shown that it would not give significant improvement.

Having a set of cliques, where each clique is set of up to s non-overlapping rules, we evaluate each of them and assign a score. To find the most efficient model with the best possible generalization and prediction capabilities, we are using a stratified cross-validation procedure [106]. For each fold, the model is trained according to Algorithm 2. Let us denote the training and testing example sets for each fold as consecutively $\mathcal{VS}^{train}$ and $\mathcal{VS}^{test}$. At this point, having clique q, we want to consider its rules' conditions as possible competence areas for a final classifier. For each competence area S_i defined by the given rule's predicates, we train the DT using only those training samples that belong to such area. Let us denote $\mathcal{VS}_i^{train}$ as the set of training samples used to train the tree in competence area S_i, then:

$$\mathcal{VS}_i^{train} = \{x \in \mathcal{VS}^{train} | x \in S_i\} \tag{3.7}$$

To make the final model adjustable in terms of complexity induced by the tree's depth, one can pass arbitrarily parameters to the base tree models, such as maximum tree depth

Algorithm 1 Procedure of finding best model.

Input:
set of training samples $\mathcal{LS}$
set of input rules $\mathcal{RS}$
cross-validation iterations cv
maximum number of competence areas s
evaluation method eval
Output:
trained model

procedure FINDBESTMODEL($\mathcal{LS}$, $\mathcal{RS}$, cv, s, eval)
  $\mathcal{RS}_m \leftarrow$ MEASURE($\mathcal{RS}$)
  candidates $\leftarrow$ FINDNOTOVERLAPPING($\mathcal{RS}_m$)
  $\text{CV}_{\text{folds}} \leftarrow$ STRATIFIEDCV($\mathcal{LS}$, cv)
  $\text{candidates}_{\text{scores}} \leftarrow \emptyset$
  for all $c \in$ candidates **do**
    $c_{\text{scores}} \leftarrow \emptyset$
    for all $\mathcal{LS}_{\text{fold}}, \mathcal{VS}_{\text{fold}} \in \text{CV}_{\text{folds}}$ **do**
      model $\leftarrow$ TRAIN($\mathcal{LS}_{\text{fold}}$, c)
      prediction $\leftarrow$ PREDICT(model, $\mathcal{VS}_{\text{fold}}$)
      $c_{\text{score}} \leftarrow$ EVAL(model, prediction)
      $c_{\text{scores}} \leftarrow$ ADD($c_{\text{scores}}, c_{\text{score}}$)
    end for
    $\text{candidates}_{\text{scores}} \leftarrow$ ADD(c, MEAN(c_{scores}))
  end for
  $\text{candidate}_{\text{best}} \leftarrow$ FINDBEST($\text{candidates}_{\text{scores}}$)
  model $\leftarrow$ TRAIN($\text{candidate}_{\text{best}}, \mathcal{LS}$, d)
  return *model*
end procedure

or pruning coefficients. This allows potential users to adjust the model's complexity according to the required needs and serves the purpose of managing overfitting. For clarity, those specific parameters were excluded from algorithm listings.

It is not guaranteed by input rule cliques in $\mathcal{RS}$ to cover whole $\mathcal{X}$. To tackle this problem and avoid the possibility of any example not being covered by any of the cliques' rules, we introduce the notion of *default tree*. Such a tree is being trained on all possible samples from $\mathcal{LS}$ and is complementary to competence areas designated by the rules. Namely, it provides the solution for all not already covered regions of the feature space. Comparing this approach to other solutions used in rule classification systems, such as assigning default class label (for example, based on the majority class) or calculating distance to the nearest rule [107], the proposed procedure gives better predictive power than the former and more interpretability than the distance calculation. If no samples in the training dataset fall into it for a given competence area, then we do not train a tree for such area, and its samples are being classified by the *default tree*.

The prediction mechanism is described in Algorithm 3. For each sample, we check whether any competence area covers it. If it is true, then a tree assigned to such area is used to predict the sample. Otherwise, the *default tree* is being used. Thanks to the fact that only non-overlapping rules are considered, there is no possibility that any value will be classified by more than one tree. An example of the trained model with its competence

Algorithm 2 Training NOTE model.

Input:
set of samples $\mathcal{LS}$
set of nonoverlapping rules $\mathcal{RS}$
Output:
trained model $\hat{\Psi}$

procedure TRAIN($\mathcal{LS}$, $\mathcal{RS}$)
  $\hat{\Psi} \leftarrow \emptyset$
  for all r $\in \mathcal{RS}$ **do**
    $\mathcal{LS}_{covered} \leftarrow$ FINDCOVEREDSAMPLES(r, $\mathcal{LS}$)
    if $\mathcal{LS}_{covered} \neq \emptyset$ **then**
      $\Psi \leftarrow$ TRAINDT($\mathcal{LS}_{covered}$)
      $\hat{\Psi} \leftarrow$ ADD($\hat{\Psi}$, Ψ)
    end if
  end for
  $\hat{\Psi}_{\text{default}} \leftarrow$ TRAINDT($\mathcal{LS}$)
  $\hat{\Psi} \leftarrow$ ADD($\hat{\Psi}$, $\hat{\Psi}_{\text{default}}$)
  return $\hat{\Psi}$
end procedure

areas is depicted in Figure 3.1.

Furthermore, the learning process is adjustable by *cv* and eval parameters. For each aforementioned fold in $\mathcal{VS}$, eval calculates score using samples predicted in $\mathcal{VS}^{test}$. The most basic example of such a scorer is just accuracy. The final single clique's score is the evaluation method's value averaged over all folds.

The best possible rule set is selected in the final stage by choosing one with the highest score. The final model is then trained using competence areas designated by rules in the set and with all samples from $\mathcal{LS}$. Since competence areas are just simple rules and the *decision trees* within them could also be interpreted as sets of rules, each prediction the model provides can also be interpreted as given by rules.

3.4 Computational complexity analysis

Let N and d denote the number of training samples and feature space dimension, respectively. Furthermore, let us use parameter symbols as in Table 3.1 and additionally denote r as input rules count together with p as the maximal possible condition count in a single

Table 3.1: NOTE learning parameters with description.

parameter	description
s	Maximal number of subspaces that the algorithm will use. It is not guaranteed that this number will be achieved, as this depends on input set of rules.
eval	Function used for scoring each set of considered competence areas. A model with a higher score will be used.
cv	Number of cross validation folds.

Algorithm 3 Predicting with NOTE.

Input:
NOTE model
samples to predict $\mathcal{XS}$
Output:
list of predicted labels $\mathcal{M}_p$

```
1: procedure PREDICT(model, XS)
2:     M_p ← List[]
3:     for all x ∈ XS do
4:         Ψ_selected ← ∅
5:         for all Ψ, competenceArea ∈ model do
6:             if competenceArea covers x then
7:                 Ψ_selected ← Ψ
8:             end if
9:         end for
10:        if Ψ_selected is empty then
11:            Ψ_selected ← GETDEFAULTTREE(model)
12:        end if
13:        y ← PREDICTDT(Ψ_selected, x)
14:        M_p ← ADD(M_p, y)
15:    end for
16:    return M_p
17: end procedure
```

rule.

The computational complexity of the learning procedure, as depicted in Algorithm 1, can be factorized as follows:

$$\begin{aligned}\mathcal{O}(&\text{measure}(r,p) + \text{findCliques}(r) + \\ &\text{evaluate}(r, cv, s, N, d) + \text{findBest}(r) + \text{train}(N, s))\end{aligned} \tag{3.8}$$

where:

- *measure* is the complexity of measuring rules,
- *findCliques* is complexity of finding cliques,
- *evaluate* is complexity of evaluation of all cliques,
- *findBest* is complexity of finding the best rules set among already measured cliques,
- *train* is complexity of a single training iteration of the model.

Rules measuring complexity depend on the number of rules and the maximum possible count of conditions within a single rule. In the worst case, to determine if two rules are overlapping or not, we will need to compare all of their predicates against each other. To measure all rules, we need to compare each pair. Therefore, the complexity of this procedure can be estimated as:

$$\mathcal{O}((rp)^2) \tag{3.9}$$

In the proposed method's implementation, *NetworkX* library was used to model the graph and find the cliques [108]. Note that not every possible clique is considered, but all maximal cliques are. The worst-case complexity of the algorithm implemented in the library is $3^{\frac{n}{3}}$, where n is the count of nodes in a graph [104]. Since the previous measuring step can raise up to r^2 measurements in total, the complexity of finding non-overlapping rules is:

$$\mathcal{O}(3^{\frac{r^2}{3}}) \tag{3.10}$$

The evaluation procedure is the most complex part of the proposed method. It consists of training and testing procedures for up to $s+1$ DTs. From the previous steps of finding an applicable set of rules, up to r^2 candidates can be found. In addition, this procedure is carried out cv times. *Scikit-learn*[43] implementation of DT was used. Its worst-case complexity is estimated to be $Nd \log N$ for training and $\log N$ for querying. Furthermore, the matching subspace is needed to be found for each sample to be classified or trained. Consequently, the training complexity of the model could be estimated as:

$$\mathcal{O}(sNd \log N) \tag{3.11}$$

and querying procedure as:

$$\mathcal{O}(s \log N) \tag{3.12}$$

For each cross-validation fold, $N\frac{cv-1}{cv}$ samples are being used for training, and $\frac{N}{cv}$ samples are used for testing. Additionally, the score needs to be calculated, which can be estimated as $(N\frac{cv-1}{cv})^2$. Therefore, the evaluation complexity is:

$$\mathcal{O}(r^2 cv(sNd \log N + \frac{s \log N}{cv} + N^2)) \tag{3.13}$$

3.5 Experimental evaluation

The proposed method is being compared to the subject of explanation – RF – as well as other standard, transparent classification algorithms, such as DT and ones mentioned in the previous chapter. To support or dismiss the hypothesis, experiments were designed to answer the following research questions:

- RQ1 What is the influence of different selection metrics and number of subspaces on model performance?
- RQ2 How does the method perform when presented with a Random Forest of different sizes?
- RQ3 What is the influence of dataset complexity on algorithm performance?
- RQ4 How complex is the output model, and what does influence it?

3.5.1 Setup

Choice of datasets A series of experiments were conducted across selected binary classification datasets. As the source for those Knowledge Extraction based on Evolutionary Learning repository was used [97]. Datasets were pre-processed beforehand. Labels were encoded into integers. Finally, to reasonably limit scope of experiments, 16 datasets were

Algorithm 4 Procedure of finding nonoverlapping rule combinations.

Input:
set of input rules $\mathcal{RS}_m$
size of maximal clique s
Output:
set of maximal cliques

```
1: procedure FindNotOverlapping(RS_m)
2:     G ← empty graph
3:     for all rule ∈ RS_m do
4:         for rule_other ∈ {r : RS | r ≠ rule} do
5:             if rule is not overlapping rule_other then
6:                 G ← AddNodes(rule, rule_other)
7:                 G ← AddEdge(rule, rule_other)
8:             end if
9:         end for
10:    end for
11:    results ← FindMaximalCliques(G, s)
12:    return results
13: end procedure
```

randomly chosen. Although there is no consensus about a minimal amount of datasets when comparing classification algorithms, such an amount is widely considered by the research community as standard and good practice. Details of used datasets are in Table 2.1. Each dataset was shuffled and split according to the 5x2 CV experimental protocol. Reasons behind such choice were described in chapter 2.

Complexity discretization In order to be able to raise conclusions about how a given algorithm performs under certain dataset properties, complexity metrics were computed for every training fold. Complexity periods were calculated as follows. First, every complexity metric was calculated for every used dataset training fold. Then, those complexities were discretized using a quantile-based approach into equal-sized buckets. Three discrete values were used: *low*, *medium*, *high*. Such quantization resulted in each experiment training instance having assigned multiple discrete values equal to the amount of calculated complexity metrics. To get a final estimation of how complex the data fold is, the mode of those discrete values was taken. In the case of draws, a higher complexity value was used. This resulted in every experiment having one complexity label assigned. Specific, after-discretization bin values may be found in Table 2.2.

Base algorithms For comparison, state-of-the-art RF with DT in CART implementation [109] were used. For comparison, three other interpretable models were used: *RuleFit*, Greedy rule list (*Greedy*) and One Rule (OneR). Their internals were described in previous chapters.

Measuring models complexity In order to objectively compare trained model complexities, they needed to be quantified into numerical values. For the sake of this experiment, the pattern of assigning numerical values to models present in Table 3.2 was used. In the case of *RuleFit*, the linear integration step was excluded, and so was in the case of RF, as there is no standard way to quantify their complexities.

Table 3.2: Internal models complexity computation details

model	complexity formula
NOTE	# of trees + $\sum^{\text{trees}}$ # of rules
RF	$\sum^{\text{trees}}$ # of leaves
DT	# of leaves
OneR	# of rules
RuleFit	# of rules
Greedy	# of rules

Implementation and reproducibility. The described method was implemented using *Python 3.8* programming language. *scikit-learn*[43] implementation of DT and RF base models was used along with *iModels*[110] implementation of *RuleFit*, *Greedy* and OneR. Slightly modified in terms of performance complexity metric implementation based on Problexity library [58] was used. Experiments were run using *Apple MacBook Pro M1 Max* with 32GB RAM. *NetworkX*[108] implementation of graph procedures was used. Following the trend of research replicability, the method's source code has been published in the online repository[2].

Model parameters selection. The subject and base algorithms were pre-trained on a subset of datasets to choose parameters from. Four datasets for pre-training procedure were randomly selected. Grid Search method[111] was used for finding the best parameter values. Special care was taken for alpha parameters of both CART and *RuleFit* algorithms. As it follows particular pattern, exponential distribution was used for sampling those with scale of 0.1. Then, best parameters for each dataset were taken and average for continuous or mode for discrete parameters was taken as final set. DT parameters were propagated to RFs' base models. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers. After pre-training parameters for base algorithms can be found in Table 3.3.

In the following sections, results leading to answering research questions are presented.

3.5.2 What is the influence of different selection metrics and number of subspaces on model performance

The proposed model is parameterized by metrics used for scoring cliques formed by rules extracted from RF. Each clique is assigned a score, and the one with the highest is then used in the final model. Figure 3.3 shows how the model performs with different scoring functions. Variations of accuracy and balanced accuracy are present, along with – used for theoretical evaluation – test accuracy. The test accuracy is the theoretical maximum that can be achieved by the method – in other words, it is the best subspace to choose from. It can be seen that it stands out from the other measures by roughly 2% for all the metrics. This raises the observation that the algorithm is prone to metric selection, and possibly better metrics can be evaluated in the future to find and match the theoretical maximum. For the following experiments, the standard accuracy measure will be picked up as the base. The algorithm works by bounding maximal clique size with the parameter provided by the operator – the number of subspaces. It will try to look for cliques of size equal to the provided parameter, and if it fails to find such, it will iterate down until

[2] `https://github.com/bgulowaty/non-overlapping-rules-ensemble`

Table 3.3: Result of pre-training procedure for base algorithms undertaken on *wisconsin*, *australian*, *seheart* and *haberman* datasets.

Algorithm	Parameter	Value	Distribution	Pretrain iterations
DT	ccp_alpha	0.0115	exponential(0.1)	10000
	criterion	gini		
	max_features	None		
RF	size	32	range(1,100)	1000
	ccp_alpha	0.0115		
	criterion	gini		
	max_features	None		
OneR	max_depth	27		1000
	criterion	gini		
Greedy	max_depth	3	range(1,100)	1000
	criterion	gini		
RuleFit	alpha	0.1983	exponential(0.1)	1000
	size	36	range(1,100)	
	tree_size	4	range(1,100)	

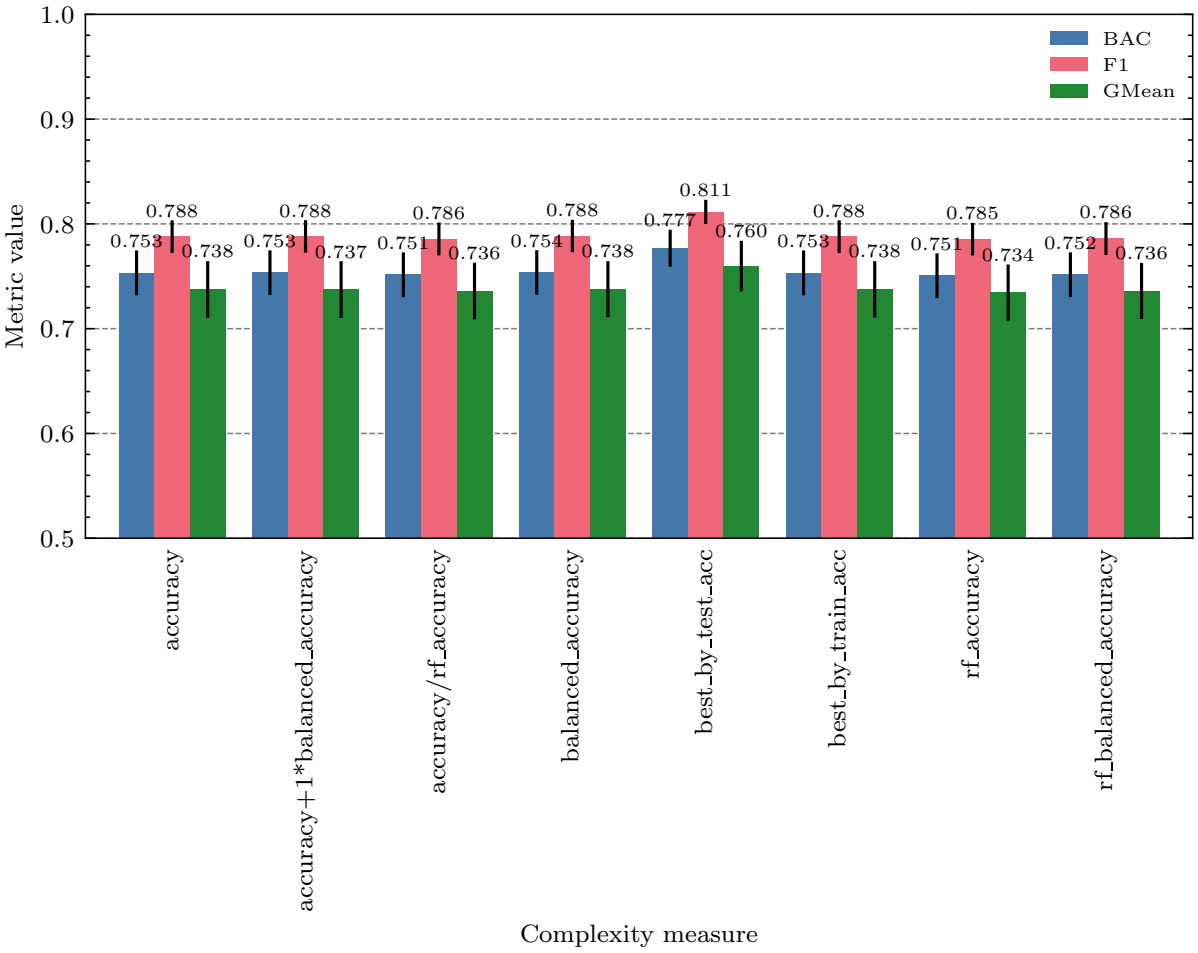

Figure 3.3: Comparison of different clique selection metrics and their performance. The numbers above the bars indicate metric values.

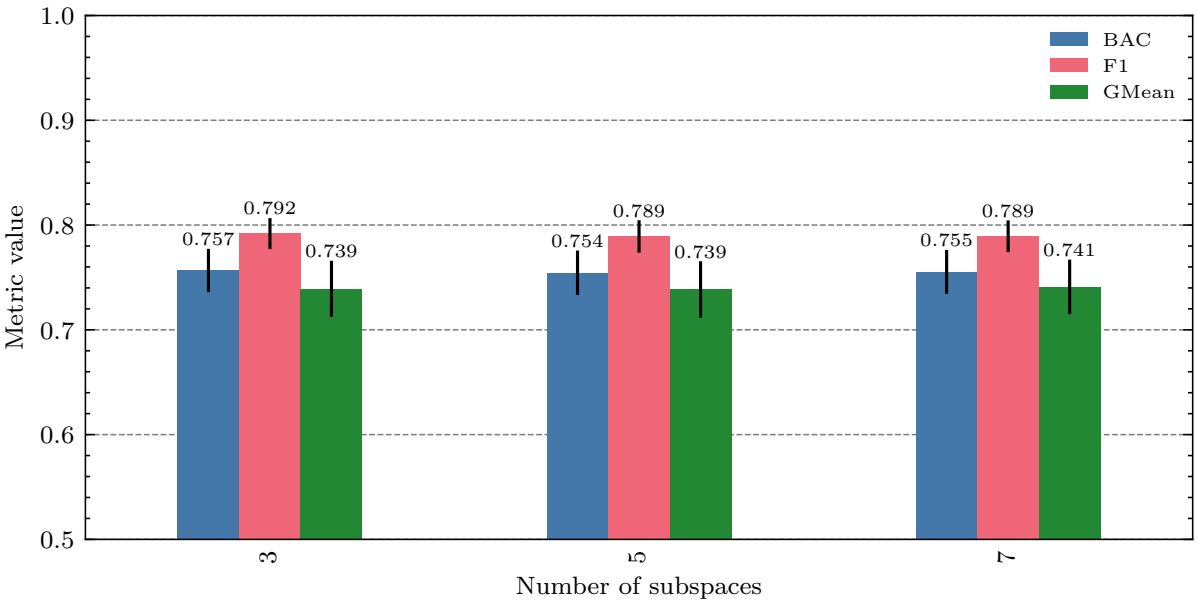

Figure 3.4: Comparison of performance metrics across different number of subspaces. The numbers above the bars indicate metric values.

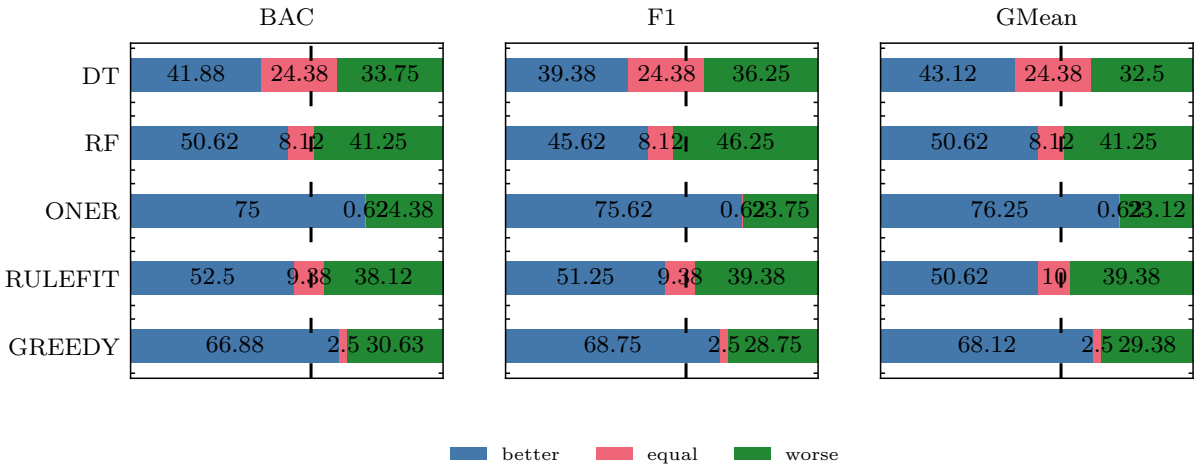

Figure 3.5: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics. Vertical dashed lines indicate sign-test values.

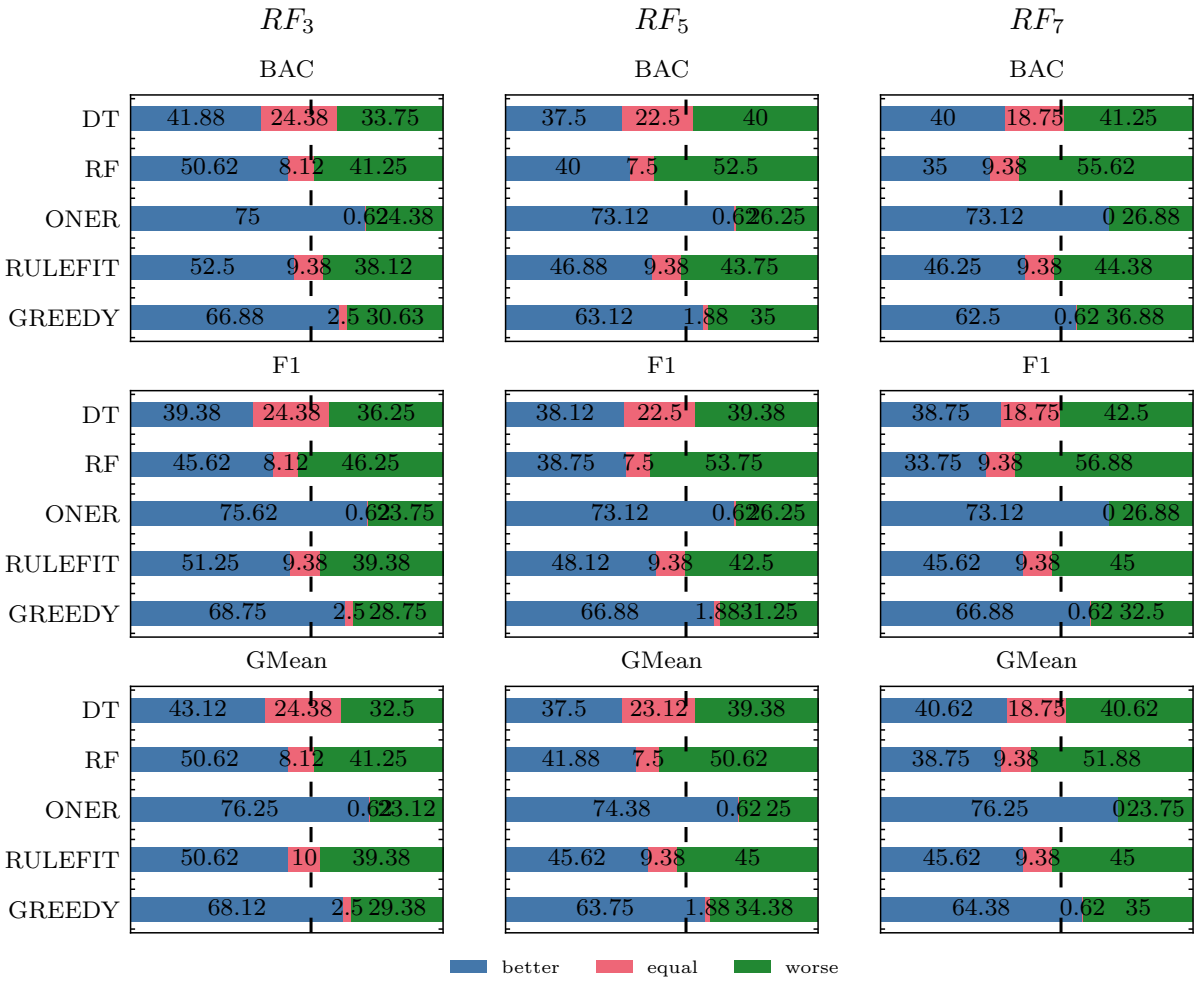

Figure 3.6: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in the domain of a different base ensemble size. Vertical dashed lines indicate sign-test values.

finding a number of subspaces that do not overlap. In extreme scenarios, it will consider only single rules.

When it comes to the number of subspaces, the experiment results may be seen in Figure 3.4. One may notice that the lesser the number of subspaces is, the better the algorithm behaves in terms of F1 and BAC scores. The only standing out metric is the GMean, which is better for seven subspaces. For further experiments, three subspaces will be taken as base.

3.5.3 How does method perform when presented with Random Forest of different sizes

Overall model performance can be examined in Figure 3.5. It is promising as Random Forest is outperformed for each metric in roughly 40% of cases. Additionally, what is important is that the model outperforms or draws the Decision Tree in 50% of cases for every metric.

Figure 3.6 presents the algorithm's performance for different metrics as an explainer for RF of different sizes. Because of significant computational complexity three, five and seven base estimators for Random Forests were considered. One of the most evident patterns

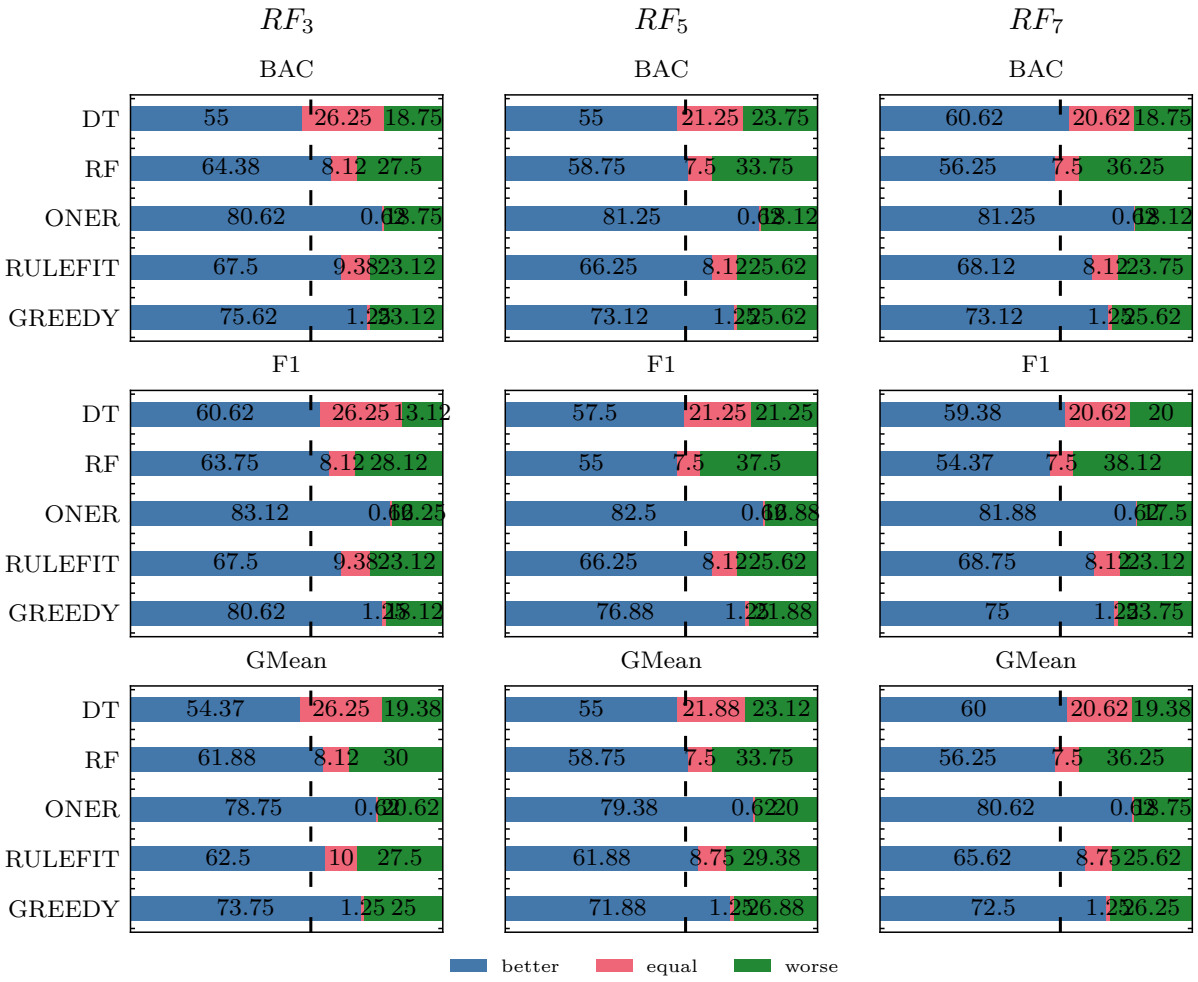

Figure 3.7: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in the domain of a different base ensemble size for cliques selected via test accuracy. Vertical dashed lines indicate sign-test values.

is the amount of wins over RF decreases as the number of base classifiers increases. For comparison, a similar barplot with test accuracy as a metric is presented in Figure 3.7. Although it's not achievable by standard scorers, with a theoretical maximum, one can see that the overall scores are much better - regardless of performance metric, all algorithms are outperformed in over 50% cases. Additionally, the quality decline as the number of estimators increases is lower.

3.5.4 What is the influence of dataset complexity on algorithms performance?

Figure 3.8 shows wins-ties-losses plot for different dataset complexities. Two interesting observations can be made. The proposed model does outperform almost all tested algorithms in 50% of cases when presented with high-complexity datasets. Furthermore, when presented with low complexity, it tends to draw with DT very often, still outperforming it in roughly 25% of cases. The method falls short when presented with moderately-complex dataset, although still achieving edge of 40% of wins over RF when it comes to GMean.

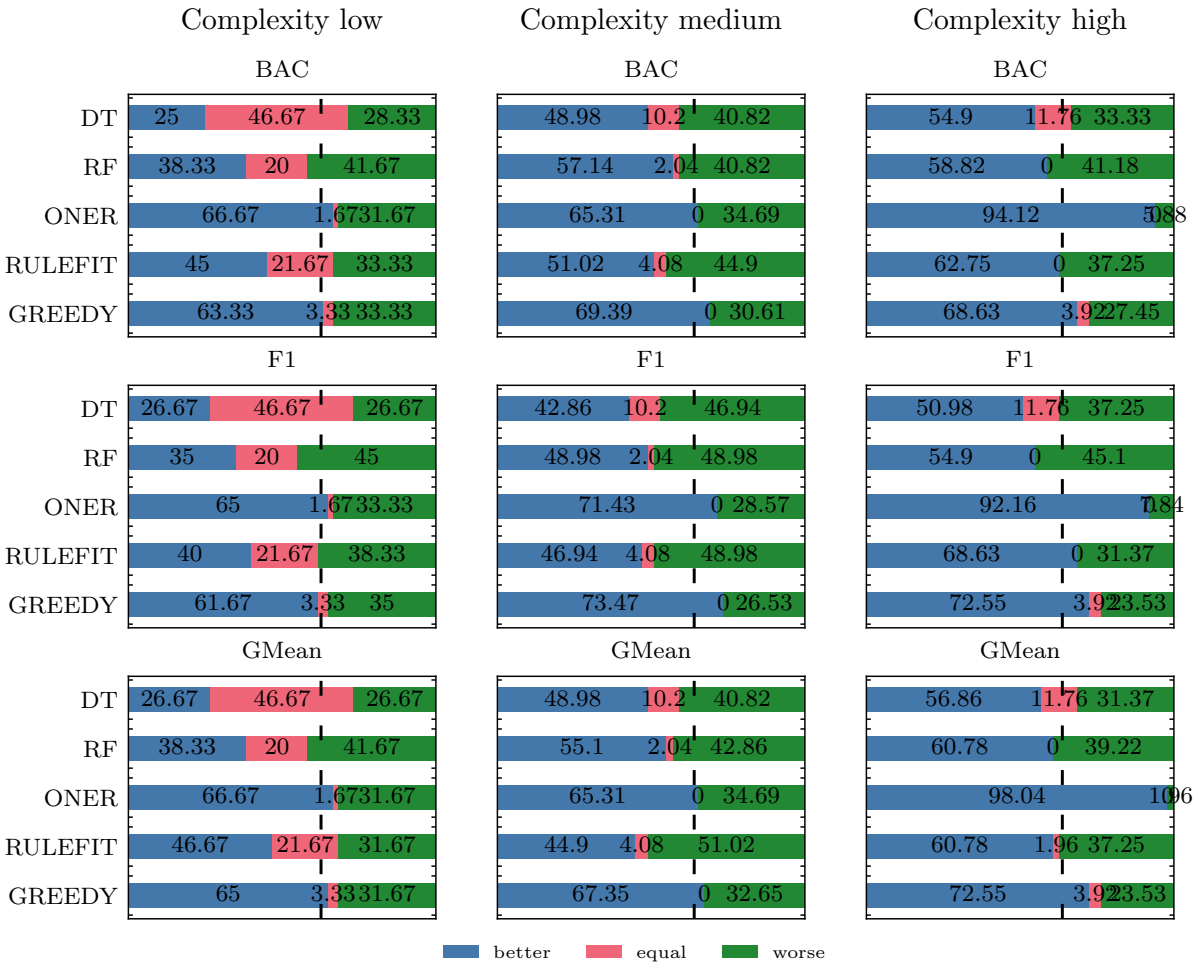

Figure 3.8: Results of overall performance comparison counted over 16 datasets in the domain of different complexities. Vertical dashed lines indicate sign-test values.

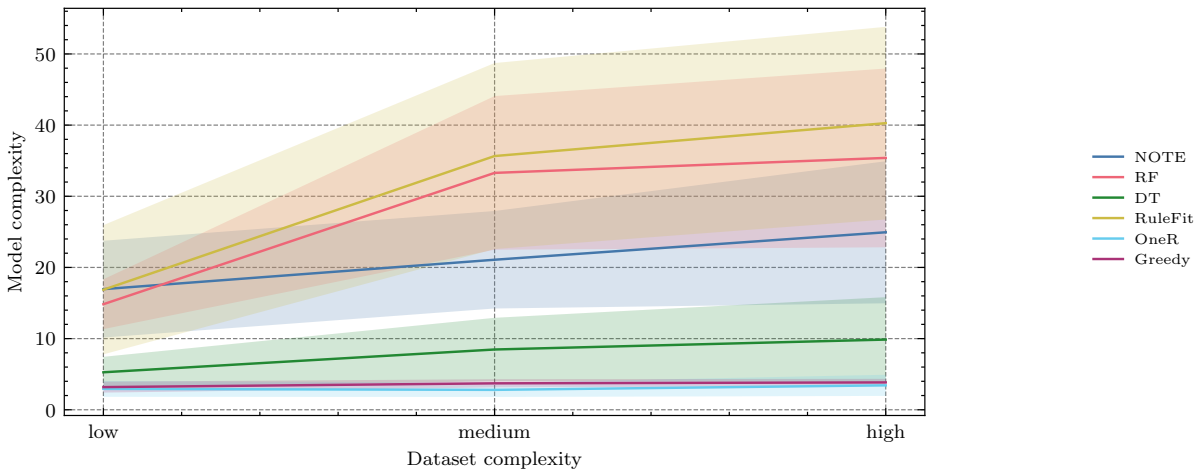

Figure 3.9: Internal complexity of compared models in the domain of dataset complexity.

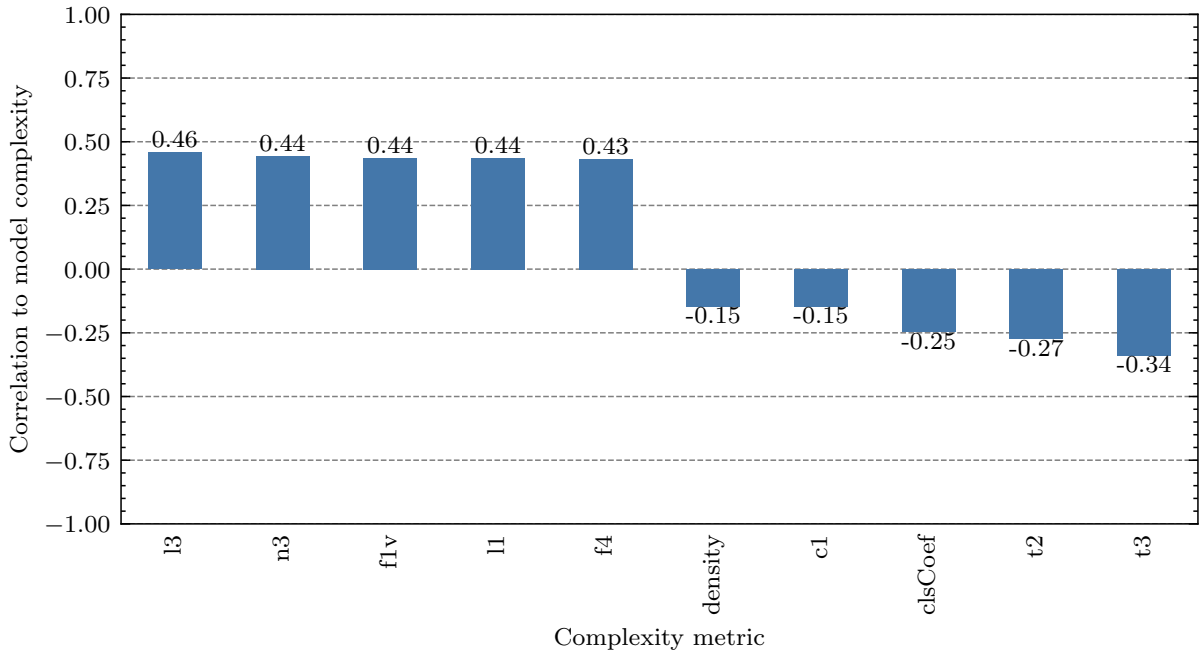

Figure 3.10: Correlation of internal NOTE model complexity to the dataset complexity metrics.

3.5.5 How complex is output model and what does influence it?

In order to compare how the proposed model complexity changes in the context of other algorithms, let's take a look at Figure 3.9. Let's recall that model complexity was computed as a number of leaves added to a number of rules. It can be seen that the model's complexity steadily rises as data complexity increases. It's also worth noting that, although it is slightly more complex than RF in lower complexities, the difference increases in its favour as data complexity increases. By taking a look at Figure 3.10, we can see that the most determinant factor for complexity increase is linearity, as defined by L3 and L1 metrics. Additionally, model complexity decreases as the average number of features per point and PCA dimensions per point increase.

3.6 Summary and lessons learned

In this chapter NOTE, a novel algorithm for explaining RF predictions was presented and evaluated in a more extended version than in its original paper [112]. The algorithm was verified for its behaviour when explaining Random Forests of different sizes and when performing predicitons under circumstances of different dataset complexities. Additionally, internal model complexities were evaluated, with an emphasis on capturing the dynamics of their change.

The proposed XAI method turned out to be an effective, from a predictive power point of view, explainer for RF. It has been shown to be competitive in terms of predictive abilities when explaining small-sized forests of three, five and seven trees. What was observed is that while the method could outperform smaller forests, its performance would show a trend of decreasing as the ensemble got bigger. Data complexity wise, there are some areas where the model would consistently outperform other classic transparent classification algorithms, such as DT or OneR. Solid predictive abilities were especially

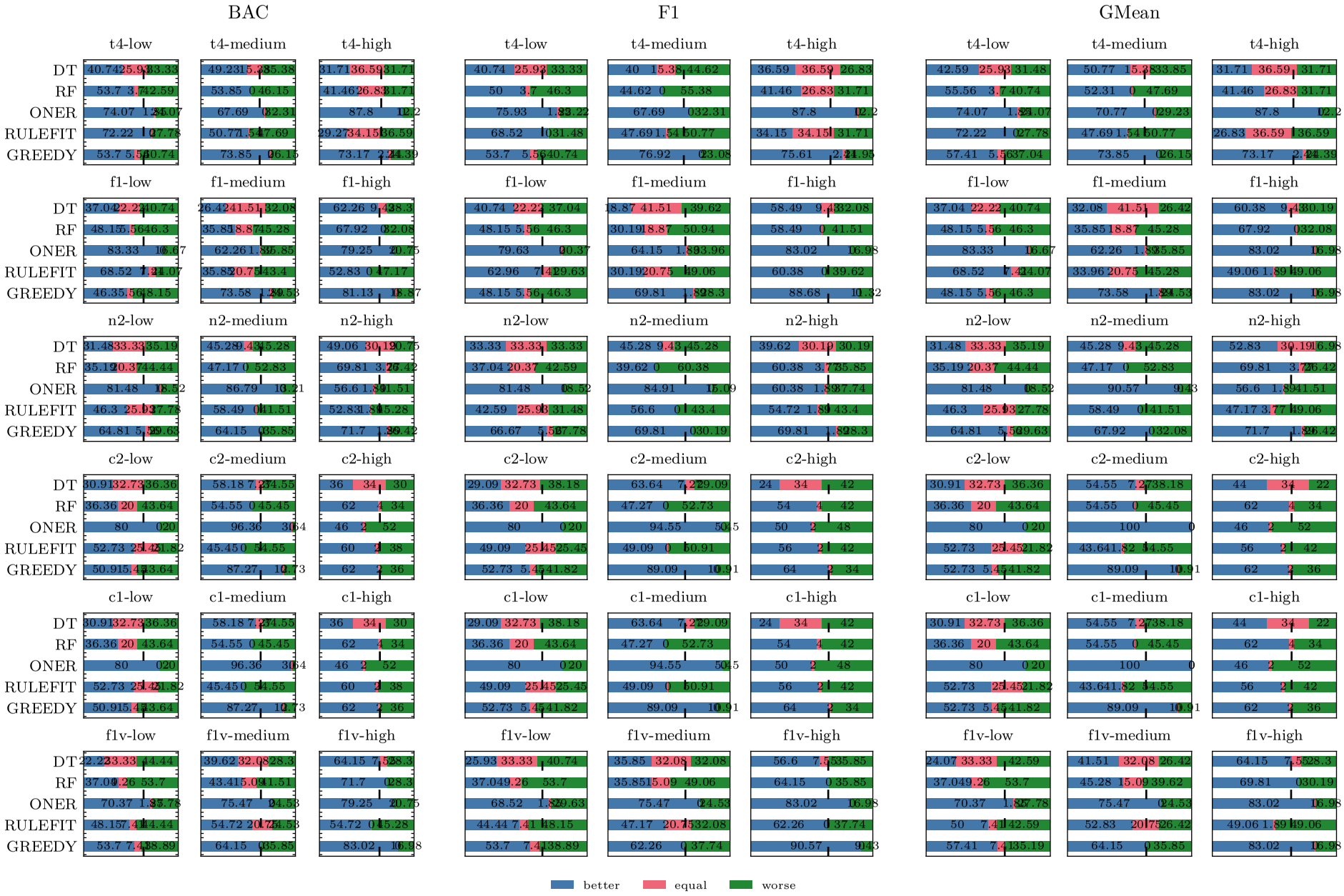

Figure 3.11: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in the domain of different dataset complexities. Vertical dashed lines indicate sign-test values.

present when:

- Data would be characterized by a big overlap of classes, as pointed out by F1 and F1V complexity values.
- When there is a moderate and low ratio of PCA dimension to the data dimensions. This property is indicated by T4 complexity measure.
- In datasets where entropy of class proportions and balance is moderate, as indicated by C1 and C2 metrics.

What is also noticeable is that, on average, the model displayed lesser complexity than RF and *RuleFit*, even though trained using the same data and having excluded integration steps in the complexity calculation of the other models. NOTE shows promises as a useful explanation method, although coming with its flaws. Computational complexity analysis has shown that the algorithm's complexity is very high, making it difficult to apply to large, multidimensional datasets in real time. It also needs to be recognized that it does not apply to some more specific domains, like continuous learning. It may still find its applicability to a wide range of areas where time is not essential and explainability of a pre-trained model is required. That said, a possible branch of further research is finding a way of reducing it. One of the possible ways to tackle this problem is to reduce the number of considered model candidates.

Apart from that, the method shows promise as a useful explanation method, and its proven performance supports the thesis statement.

Chapter 4

Utilizing constituents in building interpretable ensemble model

4.1 Introduction and related works

One of the approaches to achieving a high degree of transparency in the applied predictive models is to use an algorithm which is intelligible by design. Widely used examples of such models are rule-based models [113], DT, logistic regression or even Generalized Additive Models [114]. While, in some cases, the training procedure could be relatively easy to understand, the outcome model poses difficulties when it comes to understanding its structure or relationship between input and output.

Based on research presented in the previous chapter [112], the efficiency of having designated, specialised models for specific areas in decision space was noticed. In this chapter, a method utilising this observation in building a transparent model is proposed. It is obtained via optimizing consituents distribution around the decision space via GA.

As [115] point out, while there are XAI methods for explaining the behaviour of Evolutionary Computing (EC) processes, there is not much research when it comes utilising EC for explaining classification models. Guidotti et al. [116] use GA exploration to generate synthetic neighbourhoods for learning local interpretable predictors. Sharma et al. [117] use EC to generate counterfactual explanations – a widely adopted technique of providing samples in input space that are close to samples of a different classes. Evolutionary techniques have been used to produce self-interpretable agents [118] and discover interpretable plasticity rules [119]. It is much more common to encounter evolutionary approach in inducing glass-box models, such as trees [120]–[122] or decision rules [123]–[126].

When it comes to utilizing Genetic Programming (GP) in XAI domain, methods can be separated into 3 top-level categories [127]:

- Intrinsic interpretability – where GP is used to create inherently interpretable model
- Post-hoc interpretability – where GP is used as explainer for already trained black-box model
- Visualization – when GP is used for examining how given model behaves when performing.

The proposed method falls into the first category of intrinsic interpretability. Major subcategory in this area is so called *bloat control* [128]. As GP models tend to grow extensively, many redundant components can be observed, which results in unnecessarily

large models [129]. Bloat control aims to reduce output model size without significantly decreasing accuracy. It can be controlled by fixing the generated tree's depth [130], penalizing size of the model in the search procedure [131], applying specific (for example pruning) mutation or crossover operators [132], controlling model size via adapting the target distribution during GP process [133] or simplifying the model by replacing its parts (eg. subtrees) via simpler ones after the process [134]. Another way of making the output model more interpretable is making sure, that forms of structural complexity – other than its size – are taken into consideration in evaluation function. Measure proposed in [135] recursively aggregates complexity of the nodes underneath each internal node of induce tree. In the case of non-linear regression models, their functional complexity can be reduced by taking curvature into consideration [136]. Another technique of reducing model complexity relate to feature manipulations. Input features can be aggregated and combined to reduce problem dimensional, original features can be measured and selected [137] or completely new features can be constructed, depending on domain of the problem [138]. Genetic algorithms and evolutionary approaches are not new when it comes to building ensemble models. Oliviera et al. [139] proposed usage hierarchical multi-objective genetic algorithm in building ensemble via, first features selection, and then model selection. [140] used an evolutionary approach to prune RF and select the possible best subset of trees that would not impact the predictive abilities of the ensemble. [141] used evolutionary algorithms to enhance bagging by evolving contents of the bags to increase their diversity. Sylvester and Chawla [142] implement framework that assigns weights to classifiers in ensemble in order to maximize collaboration among classifiers.

In the following proposed approach, the evolutionary genetic algorithm is utilised to maximize the accuracy of the ensemble by finding the optimal distribution of competence areas of each model. In contrast to the aforementioned approaches, the proposed ensemble can be thought of as an interpretable model, as models do not overlap each other. Therefore, the integration step is simplified to the simple nearest neighbour assignment.

4.2 Search based framework for transparent non-overlapping ensemble models

The proposed algorithm is expanding upon the mechanism from the previous research [112] of using dedicated models for selected decision space areas. From now on, it will be referred to as Optimal Centroids because of its internal mechanics explained later.

The following are details of the proposed learning algorithm, whose pseudocode is shown in Alg. 5. A possible classifier ensemble is represented by a chromosome that is a real-valued vector describing centroid coordinates and depths of each cluster's tree. For example, the following is GA *individual* of three centroids in two dimensional space:

$$\text{indv} = \left[\overbrace{c_1^1, c_2^1}^{\text{centroid 1}} \ , \ \underbrace{c_1^2, c_2^2}_{\text{centroid 2}} \ , \ \overbrace{c_1^3, c_2^3}^{\text{centroid 3}} \right] \tag{4.1}$$

Initially, the set of centroids is randomly sampled by the user's GA preferences. In each generation, the centroids, due to real-numbered mutation and crossover operators, are "*floating*" in the feature space to find their optimal position. Additionally, to make the solutions domain continuous, tree depths are also represented as real-valued integers, and are discretized upon individual evaluation. The chromosomes are evaluated by the scoring

Algorithm 5 Procedure of model training

Input:
set of training samples $\mathcal{LS}$
centroids $\mathcal{C}$
Output:
model $\hat{\Psi}$

procedure TrainModel($\mathcal{LS}, \mathcal{C}$)
  $\Pi \leftarrow \emptyset$
  1nn $\leftarrow$ Train1NN($\mathcal{C}$)
  for all $c_i \in \mathcal{C}$ **do**
    $\mathcal{LS}_i \leftarrow$ SamplesInCluster(c_i, 1nn, $\mathcal{LS}$)
    $\Psi_i \leftarrow$ TrainDT($\mathcal{LS}_i$)
    $\Pi \leftarrow$ AddToEnsemble(Π, Ψ_i, c_i)
  end for
  return Π, 1nn
end procedure

method and the best individual is used to train the final model. If tree depth is equal to zero, the cluster given by such centroid is deactivated.

The function *SamplesInCluster* is responsible for selecting only those samples that fall into the competence area given by the first argument. *Train1NN* and *TrainDT* are respectively training NN model, used for assigning samples into correct constituent, and *decision tree* model. Finally, *AddToEnsemble* adds trained base model to final set of classifiers, along with its assigned competence area label. An exemplary visualization of the training steps for an example presented in (4.1) and depicted in Figure 4.1.

Although the scoring method could be easily manipulated and changed according to needs, ultimately, one is interested in the accuracy of the created predictive model. Therefore, fitness consists of averaged 5x2 cross-validated accuracy over training set [143]. Namely, during each fitness evaluation, cross-validation creates train and validation folds and validates t decision trees. Samples are assigned to each competence area – and finally, to each tree – by NN classifier [144]. The final model consists of up to t centroids (or *nearest-neighbor* classifier) and up to t trained decision trees.

This proposition offers excellent opportunities for future modifications, as the chosen cross-validation method and the underlying model can easily be tailored to the user's preferences. In some applications, it may be more appropriate to optimize other criteria, e.g., those dedicated to imbalanced data [145], or one may consider multi-criteria optimization methods that could offer the user the Pareto-front optimal solutions that would be selected for further exploitation. Also, the way the model is evaluated during the learning process may be adjusted to the user's preferences., i.e., the cross-validation parameters and the cross-validation itself.

4.3 Computational complexity analysis

This section will analyze the computational complexity of model querying and training procedures. Let N and d denote consecutively training dataset samples to count and feature space dimension. Furthermore, let t be competence areas count.

For experimental evaluation purposes, *Scikit-learn* implementation of decision tree classifier was used [43]. Its worst case complexity is estimated to be $\mathcal{O}(Nd \log N)$ for training. Let us assume that *kd-tree* spatial data structure is used as a base for the competence area classifier. Its construction complexity can be estimated as $\mathcal{O}(t \log t)$

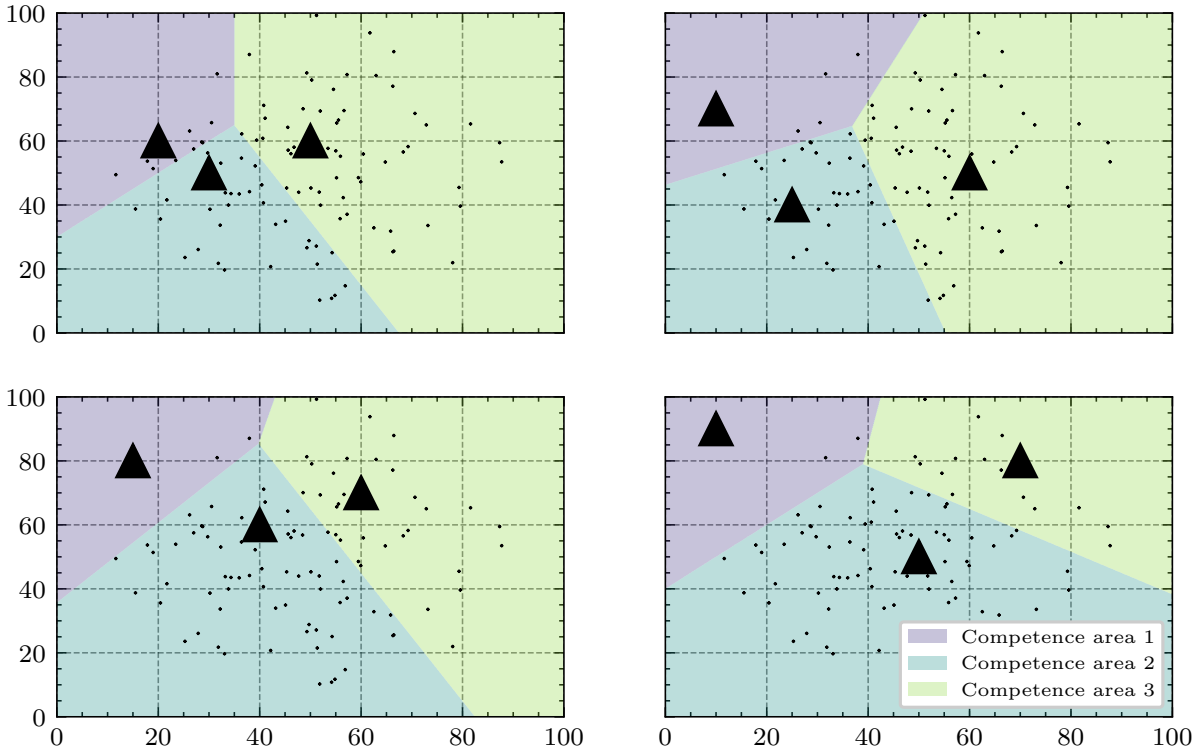

Figure 4.1: Visualisation of learning procedure using GA. Conceptually the algorithm is moving centroids (marked as black triangles) in space, evaluating them as competence area centres to find the most accurate distribution.

[146].

Then, we can estimate the computational complexity of single model training as:

$$\mathcal{O}(tNd \log N + t \log t) \tag{4.2}$$

Likewise, a trained model query consists of querying 1-NN and corresponding *decision tree* later. Tree's implementation query time is estimated as $\mathcal{O}(\log N)$ [43]. Additionally. *kd-tree* query complexity is given as $\mathcal{O}(t^{1-\frac{1}{d}})$ [146]. Therefore, we can estimate total query complexity for N samples as:

$$\mathcal{O}(t^{1-\frac{1}{d}} + \log N) \tag{4.3}$$

Those estimates aren't generally high, as compared to, for example, training a single decision tree, and could render the model usable in high data-intensive scenarios. However, the potential user should consider that the whole learning procedure includes a genetic algorithm, which parameters, especially the number of generations, population size and evaluation procedure, which may include cross-validation, multiply those estimates by a significant value.

4.4 Experimental evaluation

Originally the research questions included verifying only performance against widely used Decision Tree and Random Forest classification algorithms [147]. In the thesis, the domain was extended to include analysis of algorithm behaviour under certain dataset complexities. Additionally, analysis of the complexity of the model itself is a topic of interest. Overall, the following research questions were raised:

RQ1 What is the impact of number of subspaces on algorithm performance?

RQ2 How does the algorithm's accuracy, f1 score and geometric mean compare overall to other commonly used models?

RQ3 What is the influence of dataset complexity on model performance?

RQ4 How does the dataset complexity influence final model's complexity in comparison to other algorithms?

RQ5 How does the algorithm perform as explainer for Random Forest model?

4.4.1 Setup

In this section approach to experimental evaluation and environment details are described.

Choice of datasets A series of experiments were conducted across selected binary classification datasets. As the source for those Knowledge Extraction based on Evolutionary Learning repository was used [97]. Datasets were pre-processed beforehand. Labels were encoded into integers. Finally, to reasonably limit scope of experiments, 16 datasets were randomly chosen. Although there is no consensus about a minimal amount of datasets when comparing classification algorithms, such an amount is widely considered by the research community as standard and good practice. Details of used datasets are in Table 2.1. Each dataset was shuffled and split according to the 5x2 CV experimental protocol. Reasons behind such choice were described in chapter 2.

Complexity discretization In order to be able to raise conclusions about how a given algorithm performs under certain dataset properties, complexity metrics were computed for every training fold. Complexity periods were calculated as follows. First, every complexity metric was calculated for every used dataset training fold. Then, those complexities were discretized using a quantile-based approach into equal-sized buckets. Three discrete values were used: *low*, *medium*, *high*. Such quantization resulted in each experiment training instance having assigned multiple discrete values equal to the amount of calculated complexity metrics. To get a final estimation of how complex the data fold is, the mode of those discrete values was taken. In the case of draws, a higher complexity value was used. This resulted in every experiment having one complexity label assigned. Specific, after-discretization bin values may be found in Table 2.2.

Base algorithms For comparison, state-of-the-art RF with DT in CART implementation [109] were used. For comparison, three other interpretable models were used: *RuleFit*, *Greedy* and OneR. Their internals were described in previous chapters.

Measuring models complexity In order to objectively compare trained model complexities, they needed to be quantified into numerical values. For the sake of this experiment, the pattern of assigning numerical values to models present in Table 4.1 was used. In the case of *RuleFit* the linear integration step was excluded, and so was in the case of RF and the proposed algorithm, as there is no standard way to quantify their complexities.

Table 4.1: Internal models complexity computation details

model	complexity formula
Optimal Centroids	$\sum^{\text{trees}}$ # of leaves
RF	$\sum^{\text{trees}}$ # of leaves
DT	# of leaves
OneR	# of rules
RuleFit	# of rules
Greedy	# of rules

Implementation and reproducibility. The described method was implemented using *Python 3.8* programming language. *scikit-learn*[43] implementation of DT and RF base models was used along with *iModels*[110] implementation of *RuleFit*, *Greedy* and OneR. Slightly modified in terms of performance complexity metric implementation based on Problexity library [58] was used. Experiments were run using *Apple MacBook Pro M1 Max* with 32GB RAM. *PyMoo*[148] implementation of GA was used. Following trend of research replicability, the method's source code has been published in online repository[1].

Model parameters selection. The subject and base algorithms were pre-trained on a subset of datasets to choose parameters from. Four datasets for pre-training procedure were randomly selected. Grid Search method[111] was used for finding the best parameter values. Special care was taken for alpha parameters of both CART and *RuleFit* algorithms. As it follows particular pattern, exponential distribution was used for sampling those with scale of 0.1. Then, best parameters for each dataset were taken and average for continuous or mode for discrete parameters was taken as final set. DT parameters were propagated to RFs' base models. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers. The pre-training procedure was also applied to choosing the genetic algorithm's population size and number of generations. Results of the procedure can be found in Table 4.2.

In the following sections, results leading to answering research questions are presented.

4.4.2 What is the impact of number of subspaces on algorithm performance?

Apart from setting the parameters for the base classifier that is being trained in the constituents and GA parameters, the user of the algorithm is able to choose a number of subspaces being trained. Picking different number of subspaces was tested and the results may be found in Figure 4.2. Out of all tested values, which spanned from three to 30, results with a lesser amount of subspaces prevailed as the best. For the considered metrics, differences in means between the best and second-best parameter values were not bigger than one percent in the case of F1. What is also noticeable is that a certain tendency can be seen that the bigger the number of subspaces, the worse the algorithm's performance.

The decreasing performance is likely the result of the slight overfitting of models in every subspace. Picking more subspaces creates more specialized models in the ensemble.

[1] `https://github.com/bgulowaty/optimal-centroids`

Table 4.2: Result of pretraining procedure for base algorithms undertaken on *wisconsin*, *australian*, *seheart* and *haberman* datasets.

Algorithm	Parameter	Value	Distribution	Pretrain iterations
GA	n_gen	100		1000
	pop_size	25		
DT	ccp_alpha	0.0115	exponential(0.1)	10000
	criterion	gini		
	max_features	None		
RF	size	32	range(1,100)	1000
	ccp_alpha	0.0115		
	criterion	gini		
	max_features	None		
OneR	max_depth	27		1000
	criterion	gini		
Greedy	max_depth	3	range(1,100)	1000
	criterion	gini		
RuleFit	alpha	0.1983	exponential(0.1)	1000
	size	36	range(1,100)	
	tree_size	4	range(1,100)	

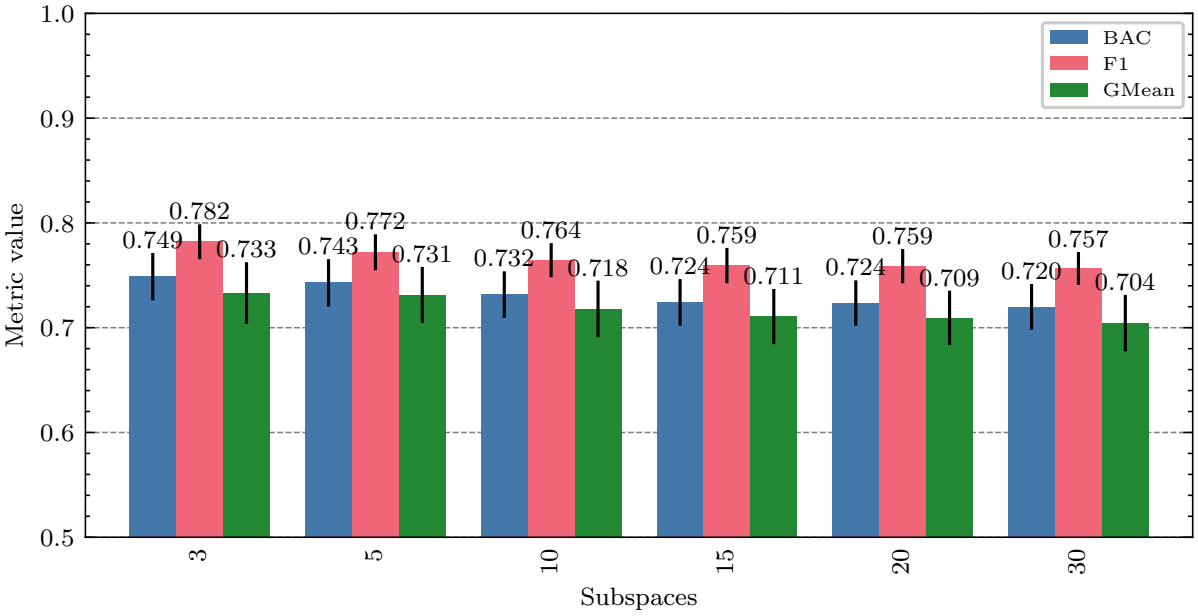

Figure 4.2: Comparison of performance metrics across different number of subspaces. The numbers above the bars indicate metric values.

It leads to some models being trained with smaller sample sizes or one specific for a given consistent pattern. As a result, overall generalizing ability decreases.

For further evaluation, the best-performing number of subspaces (3) is considered.

4.4.3 Learner performance in comparison to other models

In order to see how the proposed algorithm competes against other tested ones, let's take a look at Figure 4.3. While simple OneR and *Greedy* algorithms are usually outperformed in more than 50% of cases, followed up by *RuleFit* in roughly 40%, the situation differs when it comes to DT and RF. For the former, what is noticeable is that Decision Tree comparison results as a draw most often out of all tested algorithms. Additionally, it is constantly outperformed in over 32% of cases. Finally, RF is best outperformed when it comes to the GMean, where in over 38%, it is either worse or equal to the proposed method. The worst scenario happens for the F1 metric, where such phenomena occur only 25% times. It draws promising results given the much higher complexity of RF and state-of-the-art applications of DT.

4.4.4 Influence of datasets complexity on model performance

By taking a look at Figure 4.4, one can see how the model behaves with different complexities of datasets.

What is noticeable is that when the dataset displays low complexity, the algorithm is much more likely to tie other algorithms, which can happen even in 20% of cases when it comes to *RuleFit*. In medium complexities ties almost do not occur, and in high they occur only when it comes to DT, which is due to its fallback mechanism. It is also noticeable that algorithm is much more likely to outperform its competitors in higher complexities, especially in terms of GMean, but when it would be considered as explainer of RF, low complexities would be where it could be most likely applied due to high value of draws and wins combined.

The above inference gives general application suggestions when it comes to dataset complexity. Let's dig deeper to see what kind of characteristics and specific complexity metrics a given dataset needs to have to be suitable for the application. Figure 4.5 presents the variance of wins for a given metric and given complexity measure. It can be seen that wins are not distributed equally along all complexity metrics. For some, the wins occur more often than for others. For RF and DT, and for each of the metrics complexity measures with the most amount of wins – in other words, most often repeated as "biggest" in pie plots – were t4, c2, c1 and n2.

Figure 4.6 presents wins-ties-losses plots for most discriminative complexity metrics. The tendency from previous observation prevails – algorithm does best in cases of high complexities. Especially when the overlapping of features is high, as presented by F1 and F1V metrics. When distances between samples of the same class are relatively small in comparison to distances to samples between classes, as expressed by high N2 score, the algorithm also does very well. Finally, same thing happens when it comes to higher imbalances between labels. In all those situations, almost every other algorithm is outperformed in over 50% of cases. Where algorithm falls short are low complexities in general, where it can have almost 90% of losses as compared to Random Forest and low T4 measures. It indicates that the algorithm fails to draw better decision boundaries when the dataset displays fairly simple relationships, where, for example, the ratio of PCA dimensions to original dimensions is small, or features are not overlapping much.

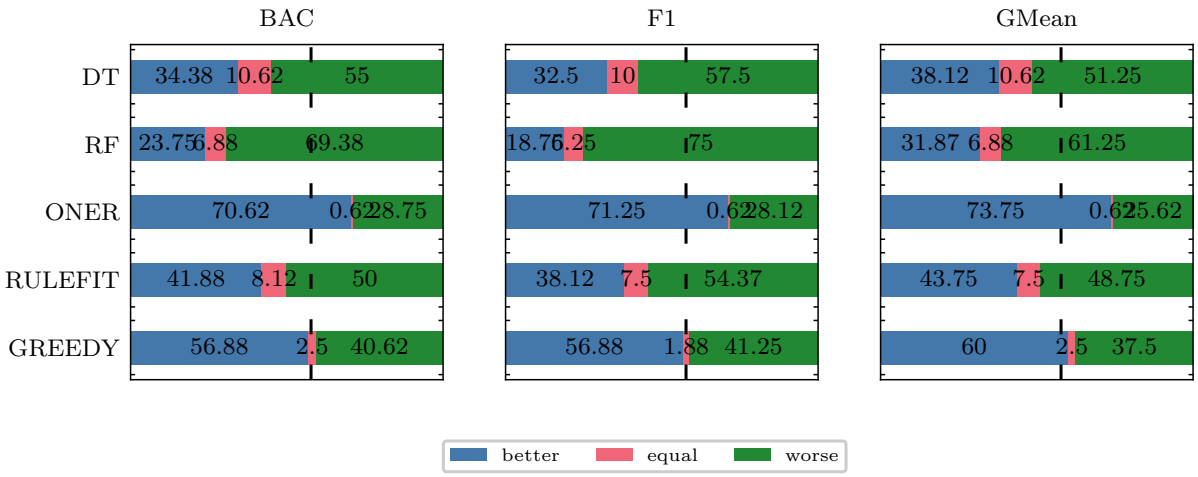

Figure 4.3: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics. Vertical dashed lines indicate sign-test values.

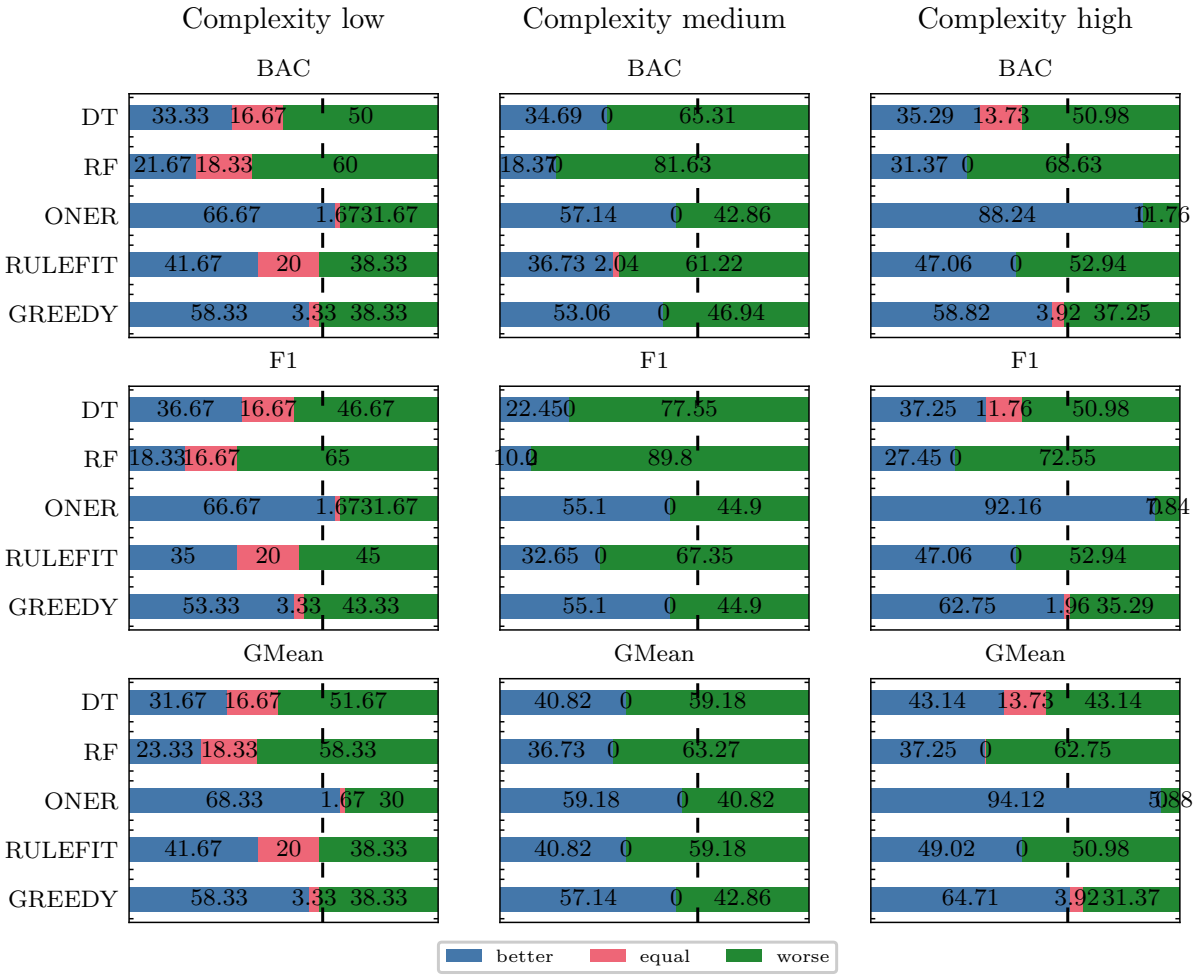

Figure 4.4: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in domain of different dataset complexities. Vertical dashed lines indicate sign-test values.

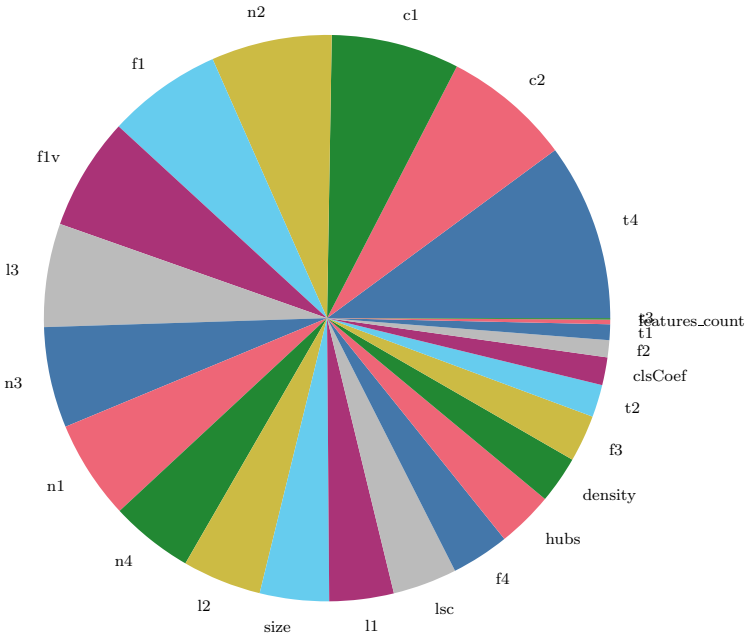

Figure 4.5: Weighted variance of wins against RF and DT for Optimal Centroids algorithm within discretized complexities for dataset complexity metrics.

4.4.5 Comparison of different model depths for different datasets

Figures 4.7 and 4.8 display model complexity evaluation in domain of datasets of different inherent complexities. Surprisingly, the proposed algorithm does not exceed *RuleFit* complexity and is competitive to DT itself. It displays an intuitive relationship where in higher dataset complexities, mean model complexity tends to be smaller than in medium ones. It is worth noting again that this complexity computation does not include the NN classification step, which might raise perception complexity for the potential operator of the model. Finally, by looking at Figure 4.9 two behaviors may be examined. Model complexity rises with the overlapping of features in training data, and it decreases when the dataset displays a high number of features per sample.

4.4.6 Algorithm as Random Forest explainer

As an extension of the original paper to accommodate the need to explainability research, two modifications of the proposed algorithm were developed:

- One where original algorithm takes RF as input and, using additional features of individual in GA, picks trees from the forest as base models (*with tree selection*)
- One where best-performing tree is fed into constituent as base model, but without the utilisation of additional GA features – simply by iterating over all trees in the forest and picking the best (*without tree selection*)

Figures 4.10 and 4.11 present performance results for both of those variations. As can be seen, there is no decisive winner between the two proposed versions. Variation with tree selection tends to outperform DT more often than its competitor. It also prevails when considered in terms of BAC and GMean. Version, where the best tree is simply

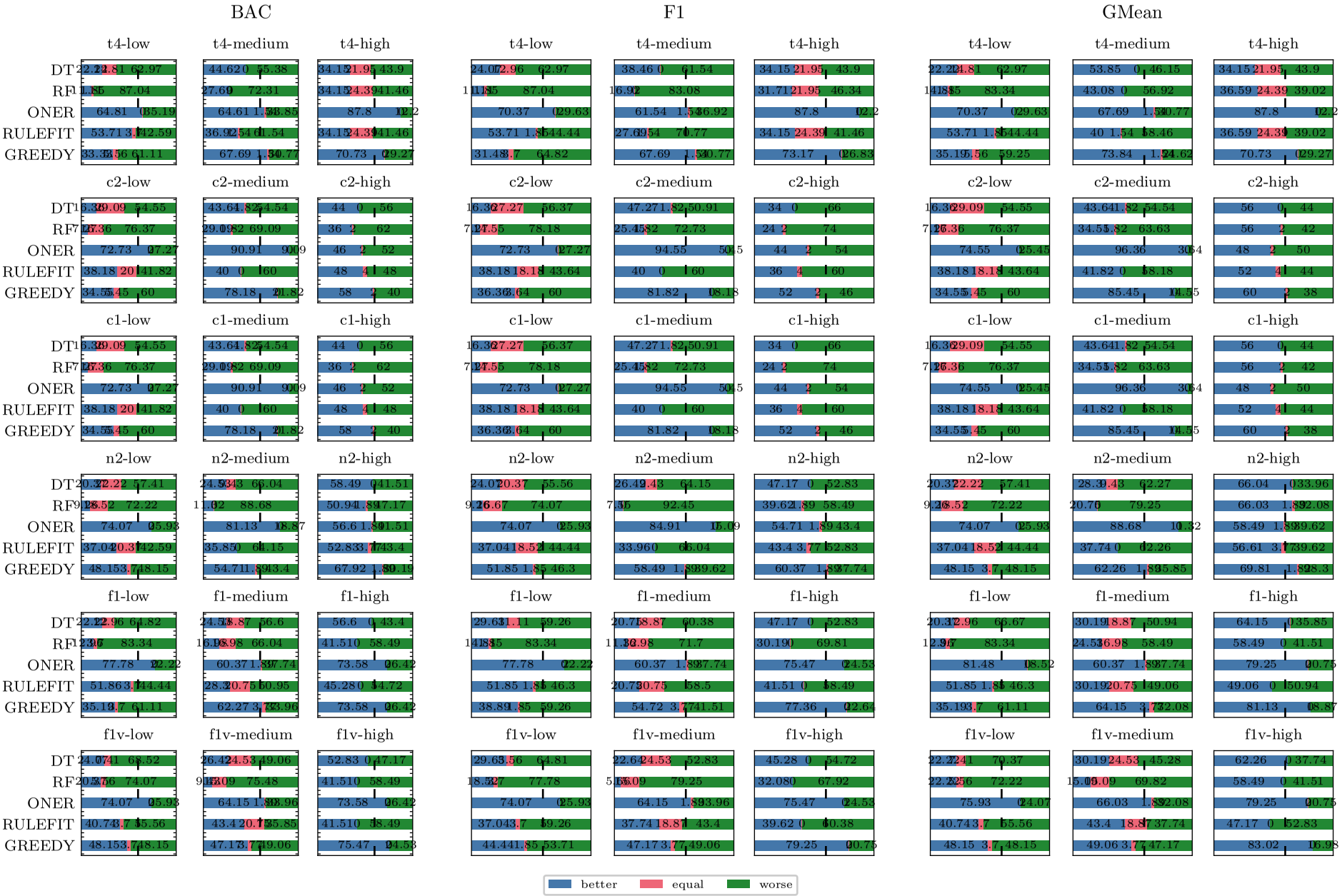

Figure 4.6: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in domain of dataset complexity metrics. Vertical dashed lines indicate sign-test values.

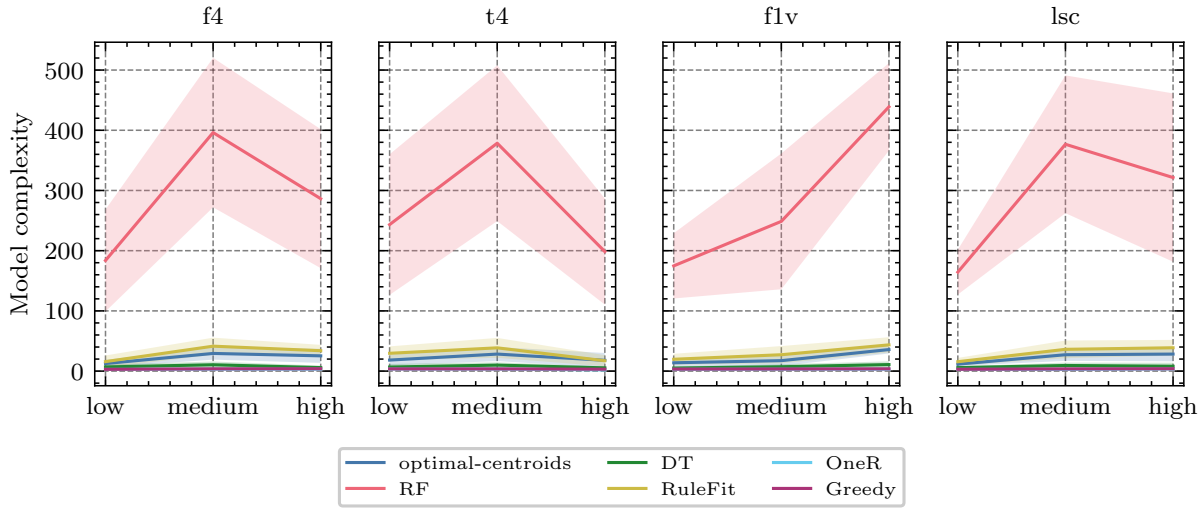

Figure 4.7: Changes in internal model complexities in domain of dataset complexity.

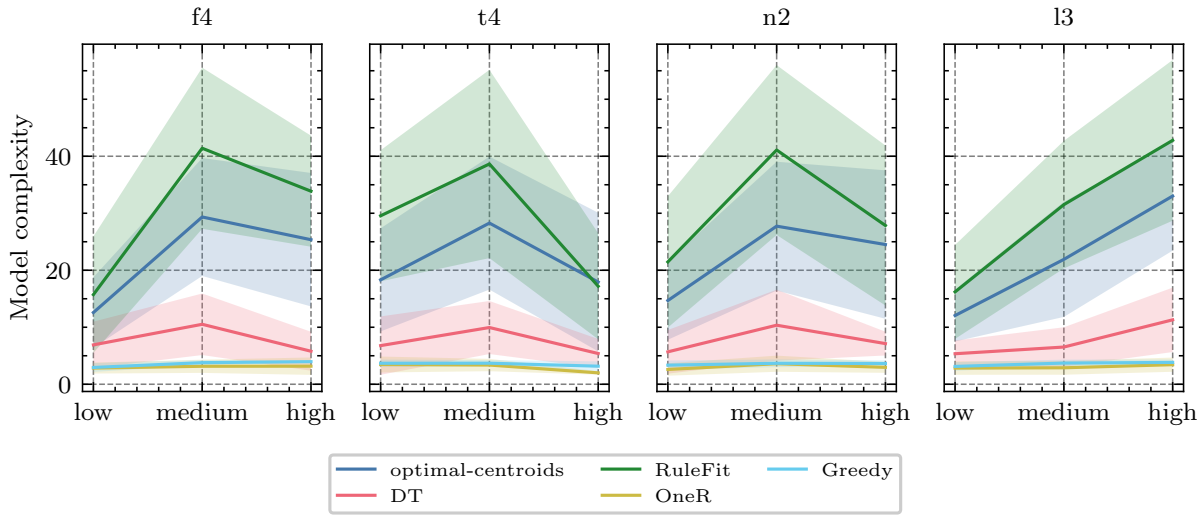

Figure 4.8: Changes in internal model complexities in domain of dataset complexity without RF.

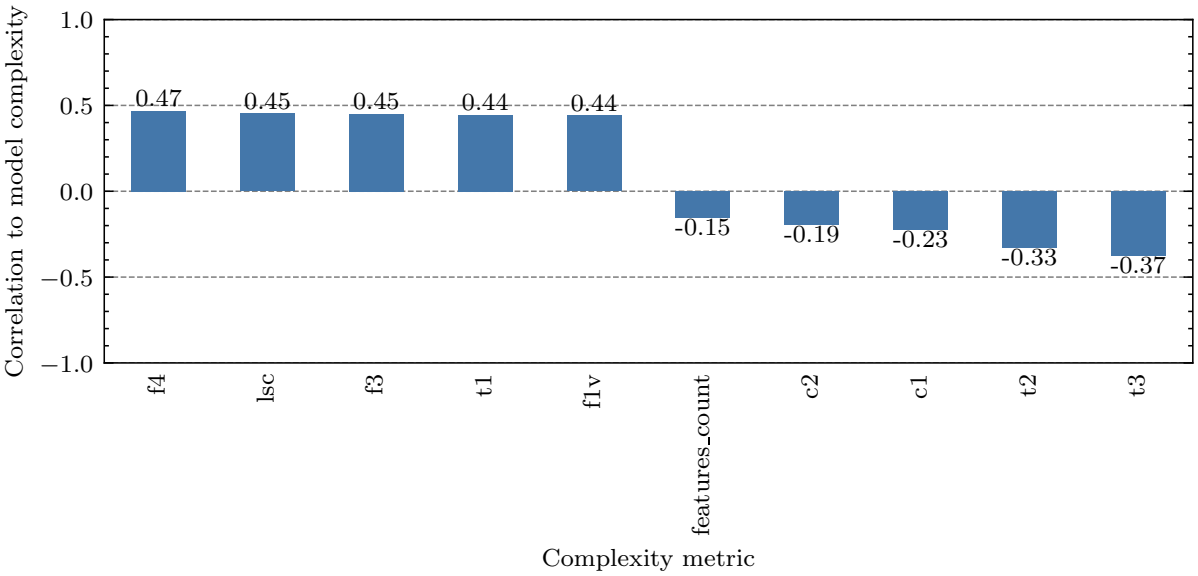

Figure 4.9: Correlation of internal model complexity to dataset complexities.

picked, on the other hand, outperforms the first one when it comes to the number of wins in F1 scoring.

4.5 Summary and lessons learned

In this chapter novel classification model creation algorithm was presented. It can be considered an ensemble model, where designated competence areas are classified by a specified model. The predictions are assigned to specified models by picking the nearest neighbour.

The algorithm was examined in a variety of different areas: by comparison to other interpretable models and RF, by validating its performance in different dataset complexities. On top of that, two variations of algorithms were presented, which serve as RF explanation methods.

The algorithm is parameterized by the amount of subspaces into which the decision space is split. It displays a decay of performance as the number of subspaces increases. As more subspaces are present, smaller are the decision areas for specific models, and therefore smaller is the amount of training samples which are being picked for training of base models. This may lead to decreased predictive capabilities as some patterns might be underrepresented, as they might span across multiple subspaces and therefore being fed to multiple base algorithms.

Overall performance-wise, the model was able to outperform the strongest out of competitors – RF – in over 30% of cases when it comes to GMean score, and as low as 18%, when it comes to BAC. Further examinations have shown that model performs overwhelmingly well when presented data with high complexity measures. Specifically, when dataset is characterized by high overlapping of features, small ratio of distances between samples inter- and intra- classes or high imbalance. In those cases, algorithm was able to stable outperform almost all models in over 50% of cases GMean wise. When presented with a dataset of low complexity - especially when the ratio of the PCA dimensions to the original dataset dimensions is small – model falls short in performance. Additionally, the state-of-the-art CART algorithm was outperformed in high and medium complexities

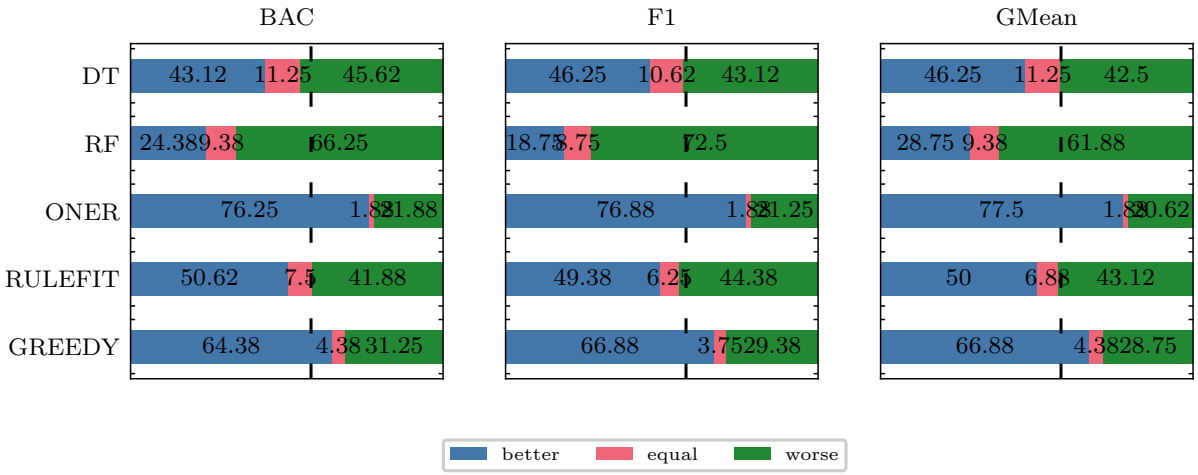

Figure 4.10: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics for algorithm explainer modification with tree selection. Vertical dashed lines indicate sign-test values.

Figure 4.11: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics for algorithm explainer modification without tree selection. Vertical dashed lines indicate sign-test values.

steadily in roughly 50% of cases.

Finally, algorithm's complexity was verified. It was shown that inherent model complexity, excluding the KNN step, goes on par with other models excluding RF. Complexity was usually lower than this of *RuleFit* algorithm and higher than this of DT.

Chapter 5

Application of the complexity measure in interpretable model training

5.1 Introduction and related works

In the previous chapters, introduced methods utilised the notion of splitting decision space into specialized parts covered by specific models. While the first method's outcome could, in fact, be interpreted as a set of rules, it was complex in terms of computation. The second method, while performing better out of the box, it used the nearest centroid to assign the samples to specific constituents, which may be interpreted as impeding explainability [114].

In this chapter, another novel method, based on observations made in previous developments, is introduced. By utilising data complexity metrics, one is able to quantify how complex a given data distribution is given some specific criteria. The overview of widely used measures and their characteristics have been reviewed in section (2.3). Measures of complexity are functions applied over labelled datasets used to estimate the difficulty of a given classification problem. Computing such value allows us to discriminate which classification method will perform better or worse [149] and allows a better understanding of the given dataset. Based on such metrics, improved classification methods are developed, for example, by integrating information about instance-level complexity to boost the learning process [150]. The learner for the interpretable model is introduced by utilising such metrics. The next section introduces its mechanics and is followed by experiments proving its viability in both learning new intelligible models, as well as explaining a black-box one.

5.2 Decision space splitting based on complexity

Dataset complexity metrics are widely used for so called meta-learning [50], [51]. It is process of selecting particular classifier based on characteristics of specific problem. It is closely related with [151], which automates various aspect of solving machine learning problem, such as feature engineering, aforementioned model search, model compression and hyper-parameter tuning.

Most of the metrics mentioned here [52] may be somehow standardized to provide a value between $[0, 1]$, where 1 means that it is complex, according to a given metric, problem. Based on that notion, one could calculate the metrics for a specific dataset and, based on its output, figure out that either a very complex or very simple model could suffice for solving the issue. Visualization of the behaviour of one of the linearity metrics

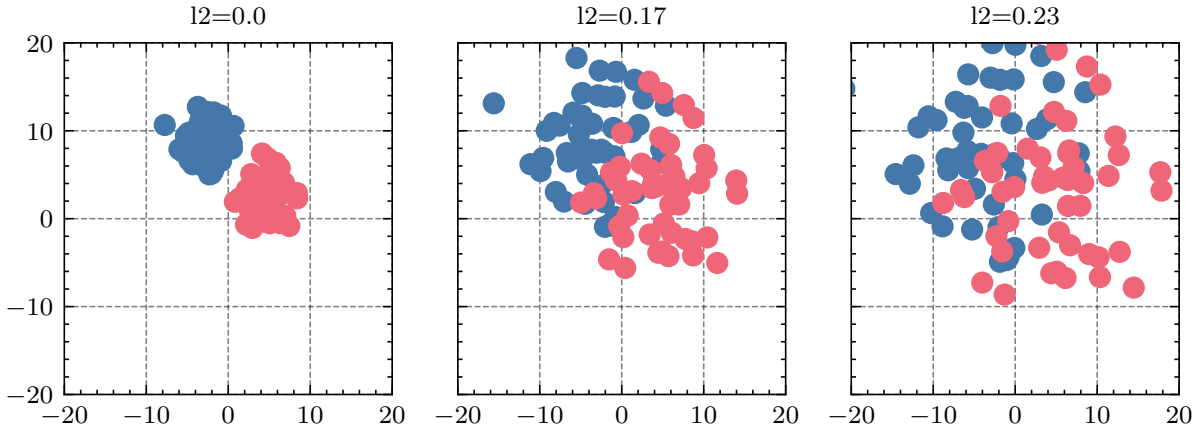

Figure 5.1: Visualisation of linearity-based complexity metric changes with dataset. Dataset was generated using `make_blobs` from scikit-learn with standard deviation 3, 5 and 7.

can be found in Figure (5.1). Therefore, one can see that finding a split that minimizes complexities in both resulting subspaces could be beneficial in terms of simplifying problems for potential learners. Having this assumption, an algorithm was created for finding an optimal split that would have learners operate on easier, in terms of given metric, classification task. Such an idea could be classified as an example of meta-learning and static classifier selection [51].

As an input parameters algorithm accepts complexity metric, dataset and base classifier, which will be used as a predictor after the splits are done. The algorithm works recursively in the sense that after splitting the decision space into two parts, it will proceed further, splitting the outcome parts until stop criteria are reached. Therefore, the algorithm accepts two additional parameters indicating whether the split will process further: a minimal number of samples in the resulting split and a minimal percentage of samples in each of the halves. This way, the operator can control the stop criteria of the algorithm, thus controlling overfitting and the level of generalization of subsequent models.

The procedure works as follows. After being presented with training dataset $\mathcal{LS}$, the algorithm considers every sample $x \in \mathcal{LS}$ and every one of d features $x = \left[x^1, x^2, ..., x^{(l)}, ..., x^{(d)}\right]^T$. In order to find the best split possible given the selected complexity metric, the algorithm iterates over features, and for every datapoint value for this feature, it considers it at the potential split. For the given the n-th sample and the i-th feature, it does so by calculating the complexities of data distributions remaining at both sides of the split. As a result, every single feature value is assigned two values:

$$\mathcal{LS}^{n,i}_{left} = \{x \in \mathcal{LS} | x^i < x^i_n\} \tag{5.1}$$

$$\mathcal{LS}^{n,i}_{right} = \{x \in \mathcal{LS} | x^i \geq x^i_n\} \tag{5.2}$$

Then, such values are averaged and saved as this split's estimate. So, overall value of split for n-th sample and i-th feature is defined as:

$$\text{complexity}(x^i_n) = \frac{\text{CM}(\mathcal{LS}^{n,i}_{left}) + \text{CM}(\mathcal{LS}^{n,li}_{right})}{2} \tag{5.3}$$

After iterating over all samples and features, the split that offers the lowest average complexity is picked. Such process is then recursively repeated for samples defined by

left and right split until stop criteria are reached. Such stop criteria can be defined by a minimal number of samples in a split or by the minimal percentage of samples in created splits. This can be understood as a variation of DT pruning, as having a bigger number will lead to less amount of splits in total, having a potentially less complex model. The whole learning procedure is described in Algorithm 6.

Algorithm 6 Procedure of training model

Input:
set of training samples $\mathcal{LS}$
complexity metric CM
minimal number of samples in split m_s
minimal percentage of samples in split p_s
base classifier Ψ
Output:
set of predicates

```
1: procedure TRAIN(LS)
2:     RS ← SPLIT(LS, ∅)
3:     Ψ̂ ← ∅
4:     for all r ∈ RS do
5:         LS_r ← GETSAMPLES(r, LS)
6:         if HASSINGLECLASS(LS_r) then
7:             Ψ̂ ← SINGLECLASSCLASSIFIER(LS_r)
8:         else
9:             Ψ̂ ← Ψ(LS_r)
10:        end if
11:    end for
12:    return Ψ̂
13: end procedure
```

5.3 Computational complexity analysis

In this section estimation of computational complexity of the algorithm is presented. Let $\mathcal{LS}$ denote the training dataset. As the number of splits, and therefore recursions, is dependent on *min_samples_split* and *min_percentage_split* parameters, and ultimately, every sample can have its own subspace, there is maximal N number of splits possible. Additionally, let's denote the number of features as d. Then, in each subspace, the base model will be trained. Let's denote its training complexity as $\mathcal{O}(\text{train})$. Additionally, the complexity metric used as splitting criteria is also attributed to some computational effort. Lets denote it by $\mathcal{O}(\text{complexityMetric})$. In each step, for each sample, there is a split calculated and the left and right sides of it are considered - thus, a factor of 2, which is reduced in final calculation. So, computing candidates of a single split for every feature has the complexity of

$$\mathcal{O}(dN\mathcal{O}(\text{complexityMetric})) \tag{5.4}$$

Then, picking the best value of split would be just finding the maximum array element, which would be $\mathcal{O}(dN)$. In the worst case, the algorithm would produce a single rule for a single training data sample, therefore iterating $N-1$. Given this scenario, no classifier training would occur, as those subspaces would be homogeneous. Assuming a scenario

Algorithm 7 Procedure of recursively splitting subspaces

Input:
current set of predicates $\mathcal{PS}$
set of training samples $\mathcal{LS}$
complexity metric CM
minimal number of samples in split m_s
minimal percentage of samples in split p_s
Output:
set of predicates

procedure SPLIT($\mathcal{LS}$, $\mathcal{PS}$)
  splitsWithScores $\leftarrow \emptyset$
  for all $x \in \mathcal{LS}$ **do**
    for all $x^i \in x$ **do**
      $\mathcal{LS}_{left}$, $\mathcal{LS}_{right} \leftarrow$ SPLIT(x^i)
      if not REACHEDENDCONDITIONS($\mathcal{LS}_{left}$, $\mathcal{LS}_{right}$, m_s, p_s) **then**
        $\text{CM}_{\text{left}} \leftarrow \text{CM}(\mathcal{LS}_{left})$
        $\text{CM}_{\text{right}} \leftarrow \text{CM}(\mathcal{LS}_{right})$
        splitsWithScores $\leftarrow (x^i, avg(\text{CM}_{\text{left}}, \text{CM}_{\text{right}}))$
      end if
    end for
  end for
  if size(splitsWithScores) = 0 **then**
    return $\mathcal{PS}$
  end if
  LS_{left}, LS_{right}, predicate $\leftarrow$ GETBEST(splitsWithScores)
  $\mathcal{PS} \leftarrow$ ADD($\mathcal{PS}$, predicate)
  return SPLIT($LS_{left}, \mathcal{PS}$) $\cup$ SPLIT($LS_{right}, \mathcal{PS}$)
end procedure

where the split occurs in a way where in every subspace there are two samples of different classes, training can occur $\frac{N}{2}$ times. Therefore, having an assumption that training single base model is more complex than iterating over dataset, total worst-case complexity is:

$$\mathcal{O}(dN^2\mathcal{O}(\text{complexityMetric}) + \frac{N\mathcal{O}(\text{train}_{\text{base}})}{2}) \tag{5.5}$$

where $\mathcal{O}(\text{train}_{\text{base}})$ is the complexity of building base classifier. As for prediction, having the assumption that predicting using a base model is more complex than reading constant class assigned to a given subspace, one would need to iterate over all rules and check their predicates. Assuming that model produced r rules, where most complex rule has p predicates and, that base classifier prediction complexity is $\mathcal{O}(\text{predict})$, the prediction complexity for single sample is:

$$\mathcal{O}(rp + \mathcal{O}(\text{predict})) \tag{5.6}$$

5.4 Experimental evaluation

To analyse the proposed method in terms of applicability for classification and explanation of complex models, the following research questions were raised:

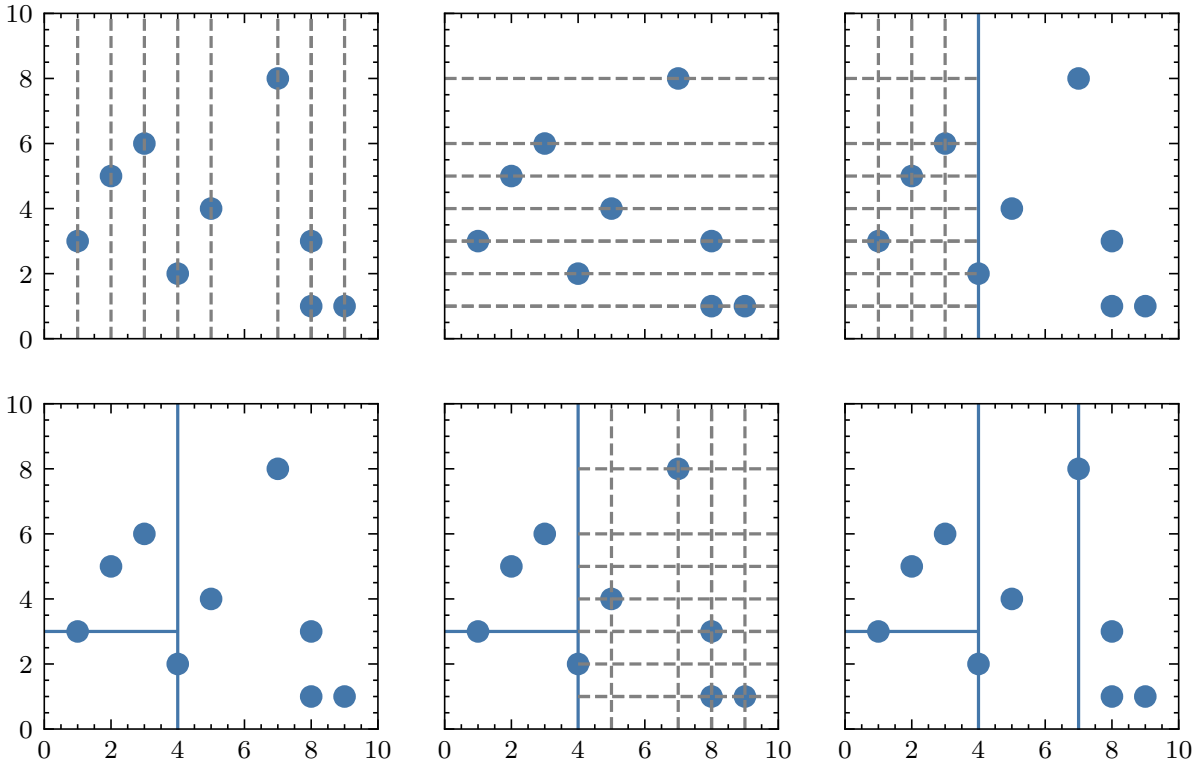

Figure 5.2: Visualisation of learning procedure using quad-split.

RQ1 What is the impact of the selected complexity measure used for subspace splitting and base algorithm parameters on model performance?

RQ2 How does algorithm's BAC, F1 and GMean compare overall to other selected models?

RQ3 What is the influence of dataset complexity on model performance?

RQ4 How does the dataset complexity influence the final model's complexity in comparison to other algorithms?

RQ5 How does the model perform when used as an explainer for the Random Forest classifier?

5.4.1 Setup

In this section approach to experimental evaluation and environment details are described.

Choice of datasets A series of experiments were conducted across selected binary classification datasets. As the source for those Knowledge Extraction based on Evolutionary Learning repository was used [97]. Datasets were pre-processed beforehand. Labels were encoded into integers. Finally, to reasonably limit scope of experiments, 16 datasets were randomly chosen. Although there is no consensus about a minimal amount of datasets when comparing classification algorithms, such an amount is widely considered by the research community as standard and good practice. Details of used datasets are in Table 2.1. Each dataset was shuffled and split according to the 5x2 CV experimental protocol. Reasons behind such choice were described in chapter 2.

Complexity discretization In order to be able to raise conclusions about how a given algorithm performs under certain dataset properties, complexity metrics were computed for every training fold. Complexity periods were calculated as follows. First, every complexity metric was calculated for every used dataset training fold. Then, those complexities were discretized using a quantile-based approach into equal-sized buckets. Three discrete values were used: *low*, *medium*, *high*. Such quantization resulted in each experiment training instance having assigned multiple discrete values equal to the amount of calculated complexity metrics. To get a final estimation of how complex the data fold is, the mode of those discrete values was taken. In the case of draws, a higher complexity value was used. This resulted in every experiment having one complexity label assigned. Specific, after-discretization bin values may be found in Table 2.2.

Base algorithms For comparison, state-of-the-art RF with DT in CART implementation [109] were used. For comparison, three other interpretable models were used: *RuleFit*, *Greedy* and OneR. Their internals were described in previous chapters.

Measuring models complexity In order to objectively compare trained model complexities, they needed to be quantified into numerical values. For the sake of this experiment, the pattern of assigning numerical values to models present in Table 5.1 was used. In the case of *RuleFit* the linear integration step was excluded, and so was in the case of RF, as there is no standard way to quantify their complexities.

Table 5.1: Internal models complexity computation details

model	complexity formula
Quad Split	$\sum^{\text{trees}}$ +# of rules
RF	$\sum^{\text{trees}}$ # of leaves
DT	# of leaves
OneR	# of rules
RuleFit	# of rules
Greedy	# of rules

Implementation and reproducibility. The described method was implemented using *Python 3.8* programming language. *scikit-learn*[43] implementation of DT and RF base models was used along with *iModels*[110] implementation of *RuleFit*, *Greedy* and OneR. Slightly modified in terms of performance complexity metric implementation based on Problexity library [58] was used. Experiments were run using *Apple MacBook Pro M1 Max* with 32GB RAM. Following the trend of research replicability, the method's source code has been published in the online repository[1].

Model parameters selection. The subject and base algorithms were pre-trained on a subset of datasets to choose parameters from. Four datasets for pre-training procedure were randomly selected. Grid Search method[111] was used for finding the best parameter values. Special care was taken for alpha parameters of both CART and *RuleFit* algorithms. As it follows particular pattern, exponential distribution was used for sampling those with scale of 0.1. Then, best parameters for each dataset were taken and average for

[1] `https://github.com/bgulowaty/quad-split`

continuous or mode for discrete parameters was taken as final set. DT parameters were propagated to RFs' base models. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers. The pre-training procedure was also applied to choosing the genetic algorithm's population size and number of generations. Results of the procedure can be found in Table 5.2. What is significant is that beta-binominal distribution was used for sampling the "min_samples" parameter and uniform distribution for split percentage. The selection of complexity metric user for splitting subspaces was performed as follows. To consider the algorithm applicable, it is important to consider its internal computational complexity and speed of execution. Therefore, a subset of randomly chosen datasets was chosen, and then each dataset was split according to the 5x2 CV procedure, giving 10 folds in total. Then, on each fold, each complexity metric was calculated, and its time execution time was measured. Machine specs on which it was performed were as follows: Executions were not parallelized in any way, meaning each metric was computed sequentially in Python using one core and thread. Given results were averaged, giving each metric execution time. Then, 6 fastest metrics from each group were selected for experiment and evaluation purposes, being: F2, T4, C1, N3, L2 and DENSITY.

In the following sections, results leading to answering research questions are presented.

5.4.2 What is the impact of selected complexity measures used for subspace splitting and base algorithm parameters on model performance?

Figure 5.3 presents classification metric measures in domain of complexity metric parameter used for subspacing. Mean test metric values were aggregated over all experiments for given splitting criteria. As mentioned before, potential complexity metric candidates were chosen based on their speed and complexity. It may be seen that there is no major difference when it comes to parametrizing models with different splitting criteria. All metric values, as well as their variances, are distributed the same way. Given this fact, for subsequent experiment evaluation only one base complexity measure is selected, which is density.

Figure 5.4 presents the aforementioned equivalent for the base classification algorithm. Three basic models were used, which are DT, 3-NN and SVC. It is shown that among the chosen, the best performing was the DT. It is, therefore, chosen as a parameter for evaluation in further experiments.

5.4.3 How does algorithm's BAC, F1 score and GMean compare overall to other selected models?

To validate how the proposed model compares to others, let's take a look at Figure 5.5, which presents the number of wins, ties and losses in comparison to other algorithms. The subspace splitting approach mostly resembles the decision tree classification idea. When it comes to comparison with the CART algorithm, model can be found to be better in almost 21% of cases BAC wise. In terms of F1 and GMean, DT was outperformed consequently 20% and 27% of times.

Most complex and strongest in terms of generalizing abilities of the opponents, RF, was only outperformed in terms of BAC in 12.5%, but 18.75% in terms of GMean. Other

Table 5.2: Pre-train results for *quad-split* method.

Algorithm	Parameter	Value	Distribution	Iterations
Quad Split	min_samples	28	betabinom(65, 2.578, 5.106)	1000
	min_split_percentage	0.183	uniform(0.01, 0.49)	
DT	ccp_alpha	0.0115	exponential(0.1)	10000
	criterion	gini		
	max_features	None		
RF	size	32	range(1,100)	1000
	ccp_alpha	0.0115		
	criterion	gini		
	max_features	None		
OneR	max_depth	27		1000
	criterion	gini		
Greedy	max_depth	3	range(1,100)	1000
	criterion	gini		
RuleFit	alpha	0.1983	exponential(0.1)	1000
	size	36	range(1,100)	
	tree_size	4	range(1,100)	

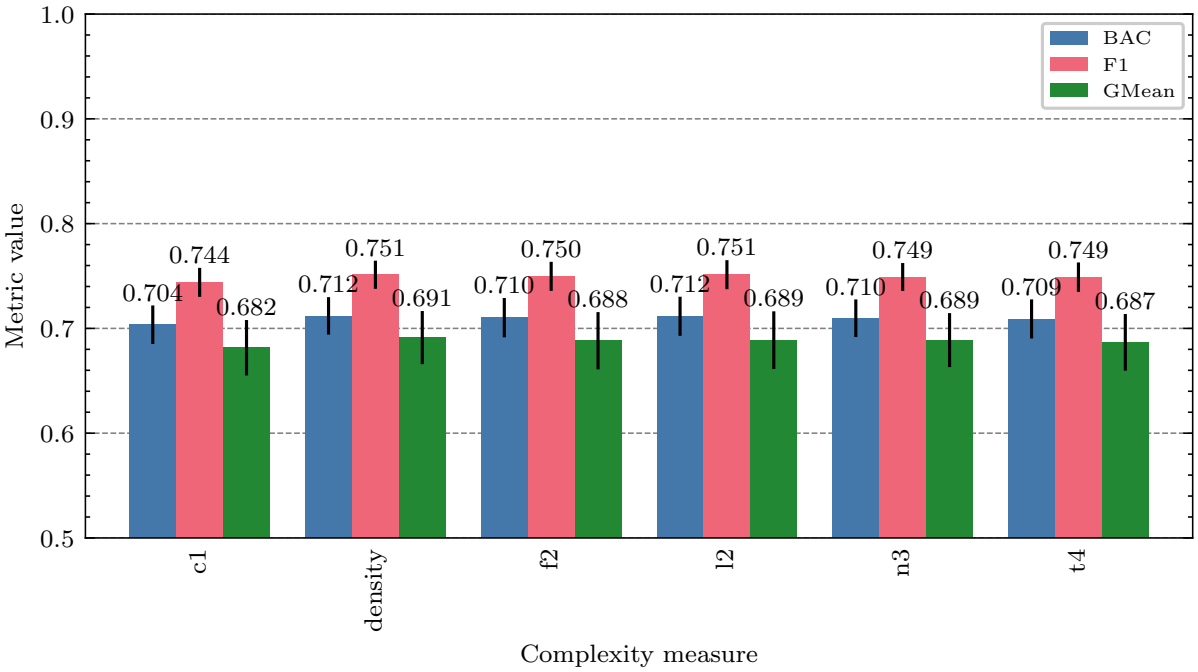

Figure 5.3: Comparison of different complexity metrics used as splitting parameter and their performance. The numbers above the bars indicate metric values.

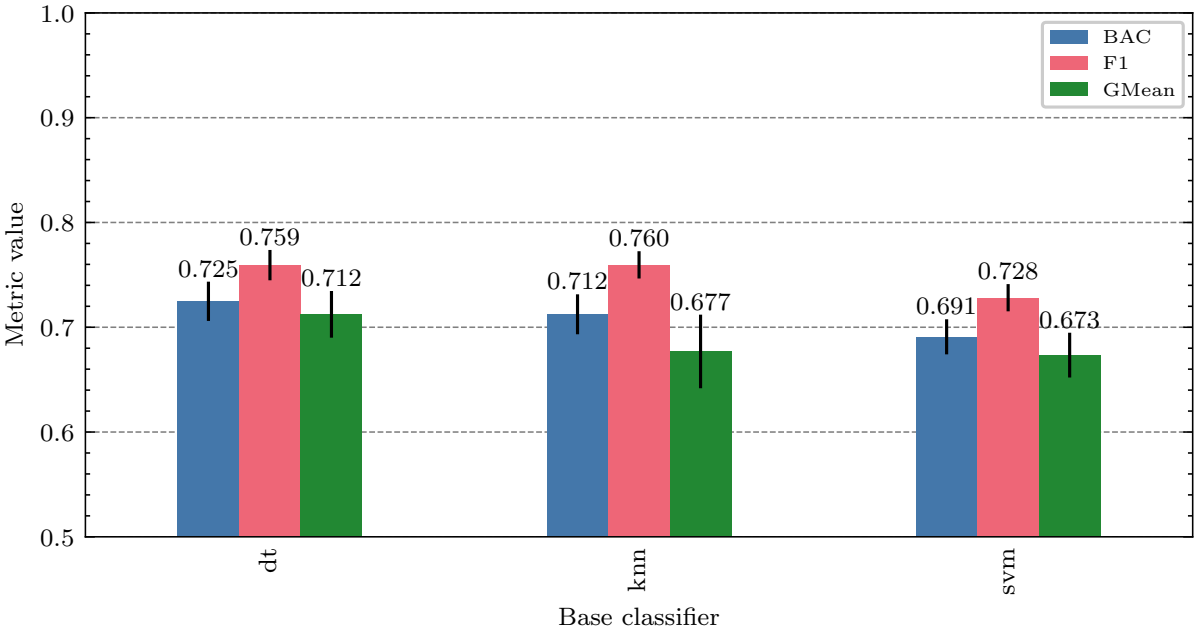

Figure 5.4: Comparison of different used base models and their performance. The numbers above the bars indicate metric values.

interpretable models, such as OneR and *Greedy* were either completely outperformed – as in case with the latter – or almost equalized when it comes to BAC.

5.4.4 What is the influence of dataset complexity on model performance?

By taking a look at Figure 5.6, one may see that dataset complexity plays an important role in Quad Split's application area. While RF was hardly outperformed in the general view, it is visible that it is only outperformed when dataset complexity is low. In this case, the proposed model is better in 11% of cases and equalizes RF accuracy in 11% of cases. The model showed rather a stable amount of wins against OneR. In general, it may be concluded that it is best to apply the model in either big or low-complexity datasets.

To validate which complexities were most predictive when it comes to the number of the model's wins against other algorithms, let's take a look at Figure 5.7. It presents the distribution of wins variance against RF and RF algorithms. Based on that, one may conclude that there are some complexity metrics which indicate whether quad-split will be performing better than other algorithms. To analyse further behaviour in different categories of those complexities, the top eight with the highest variance were selected, which are: T3, T2, T1, N3, T4, L3, F2, and L1.

Figure 5.8 shows different metrics performance for different categories dataset-wise complexities. It can be noticed that the proposed method performs exceedingly well with high complexities. In three cases of high complexity, RF is outperformed in more than 30% of the cases, while DT is outperformed in more than half the cases the same amount of times when it comes to GMean. Promising results are also shown in medium values of T3 and N3 complexities. Method falls short when the dataset is characterized by low to moderate values of feature-based metrics, as well as linearity metrics.

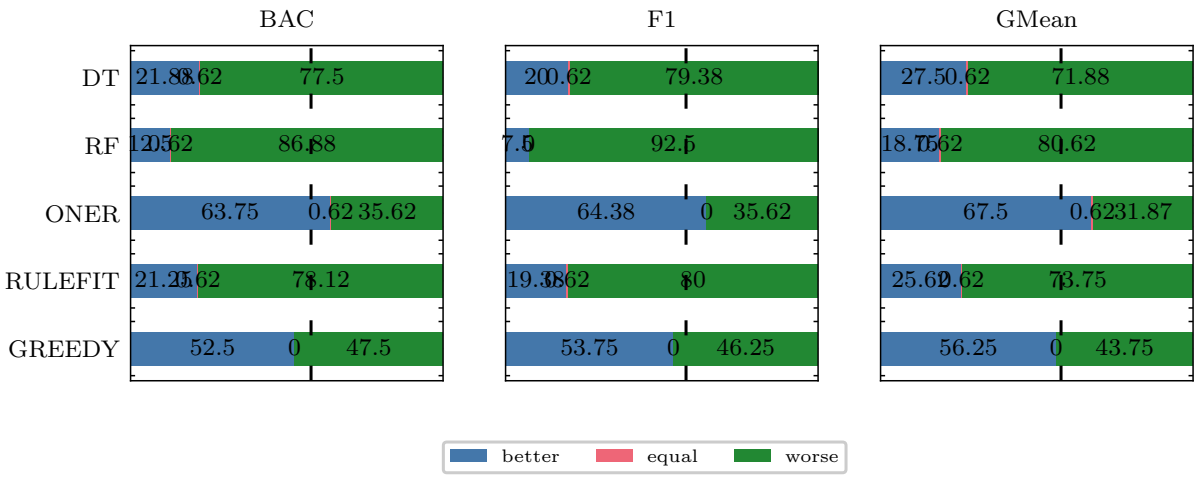

Figure 5.5: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics. Vertical dashed lines indicate sign-test values.

5.4.5 How does model perform when used as explainer for Random Forest classifier?

In order to adapt the algorithm to be able to act as an explainer of the RF model, two approaches are proposed. From now, they will be referenced to as V1 and V2.

- V1 - the algorithm is splitting the decision space as originally designed, but instead of training the base classifier, it iterates through RF's trees and picks the best in terms of accuracy for the given space.
- V2 - the algorithm is trained as usual, using DT as the base model. It uses RF's predictions as training labels.

Results for the first approach may be seen in Figure 5.9, while results for the version with just training data utilisation are in Figure 5.10. It can be seen that the tree-picking method outperforms the second version in almost every aspect. In high and low complexities, BAC and GMean wise first version beats RF twice as much, and DT roughly 15% more often. The only lacking area is the medium complexity datasets' F1 score, where the second approach slightly outperforms.

5.4.6 How does the dataset complexity influence the final model's complexity in comparison to other algorithms?

To see how the model's complexity changes in the domain of dataset complexity, the four most influential dataset complexities were calculated. The calculation assumed ranking each complexity by how much all model complexities change within it – absolute change values were summed. This allowed narrowing examination to F4, LSC, dataset size and T2. The change plots for those metrics are present in Figures 5.11 and 5.12. RF, excluding even its integration step, is the most complex. It is followed by quad-split, which usually has a tendency to follow common sense's expected behaviour, where model complexity rises along with dataset complexity. This is not true for T2 metric, where – for its high values – both *RuleFit* and Quad Split complexities seem to decline. Out of a more complex model, DT displays one of the lower complexities. Additionally, *Greedy* and OneR algorithms display the lowest internal complexity.

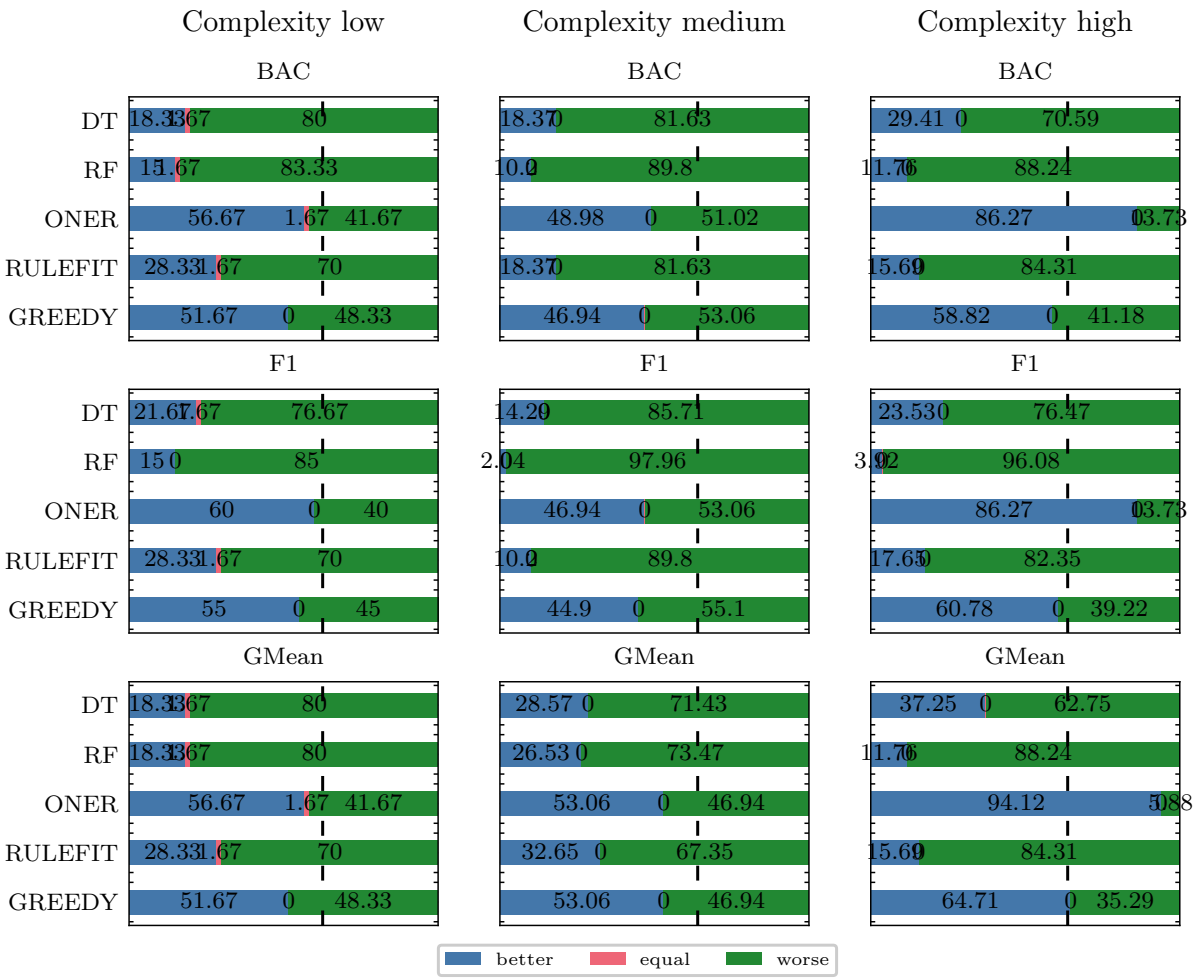

Figure 5.6: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in domain of different dataset complexities. Vertical dashed lines indicate sign-test values.

To examine what influences the complexity of Quad Split model, let's take a look at Figures 5.14 and 5.13. Five of the highest and lowest correlations were calculated for dataset complexity measures for a number of rules in the model and the proportion of complex classifiers to dummy ones. Simple classifiers are those subspaces/rules where, in the end, samples were homogeneous. Therefore, there was no need for training of the base classifier. Instead – class label was assigned to such rule. The proportion was calculated as:

$$\frac{\#\text{ of complex models}}{\#\text{ of complex models} + \text{no of simple models}} = \frac{\#\text{ of complex models}}{\#\text{ of rules}} \tag{5.7}$$

.

The model tends to have more complex classifiers as neighbourhood and feature-based metrics have higher values. What is also noticeable is that when the number of rules decreases, so does the number of complex models, which is counter-intuitive – one might suspect that when space was split fewer times, it would require more complex classifiers to be present in those. Additionally, the smaller the dataset, the more dummy classifiers will there be.

When it comes to a number of rules, the most influential predictor of those is the size of the dataset – it is natural that the bigger the dataset is, it might require more rules to be covered. Additionally, high values of complexity measures indicate more rules and

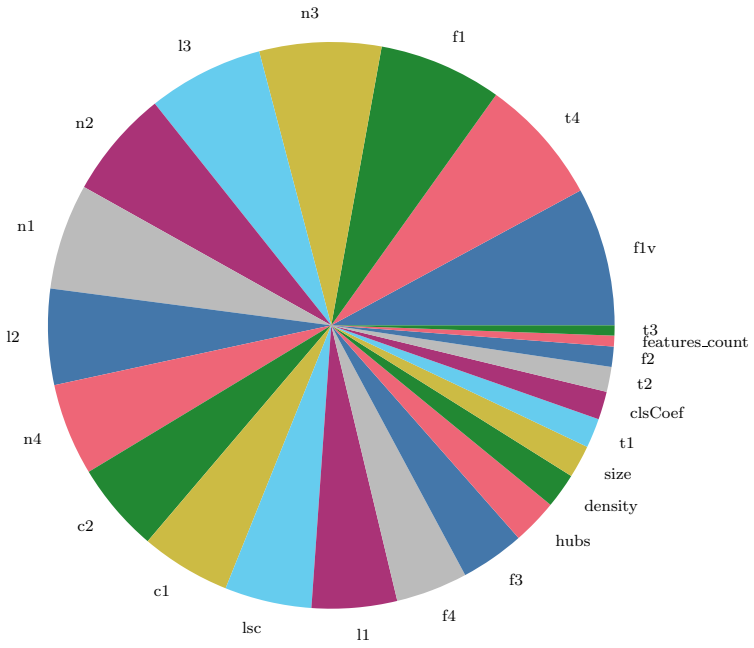

Figure 5.7: Weighted variance of wins against RF and DT for Quad Split algorithm within discretized complexities for dataset complexity metrics.

lower dimensionality datasets tend to have fewer rules.

5.5 Summary and lessons learned

In this chapter, a novel algorithm based on recursive decision space splitting was presented. Three variations of the method were examined: one purely for classification and two others acting as explanator of RF classifier. The output model has the structure of a rules list where each rule points to a trained, application-specific transparent model.

The model is parameterized by a complexity metric that is used to build a rule list via splitting subspaces. A set of fastest-performing complexity measures was evaluated as potential model parameters. Experiments have shown that there are small differences when it comes to picking splitting metrics. Average BAC, F1 score and GMean were roughly similar, with a slight edge for density metric. The algorithm allows using any function that takes a set of training data instances as splitting criteria, which allows fine-tuning it according to application criteria.

Another parameter that can be modified is the base classifier, which is trained with samples designated by rules. It allows fine-tuning it for tasks where high interpretability and explainability are needed. During experiments, the performance of three base classifiers was evaluated. Out of 3-NN, SVC and DT algorithms, DT performed best. Therefore, it was used as a base for further evaluation.

Experiments were designed and conducted to validate the performance of the proposed algorithm in comparison to other commonly used interpretable algorithms as well as RF. Different dataset complexity circumstances were considered. The proposed algorithm finds its application areas in scenarios where:

- data in which the overall distance between examples of different classes is smaller

Figure 5.8: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics in domain of dataset complexities. Vertical dashed lines indicate sign-test values.

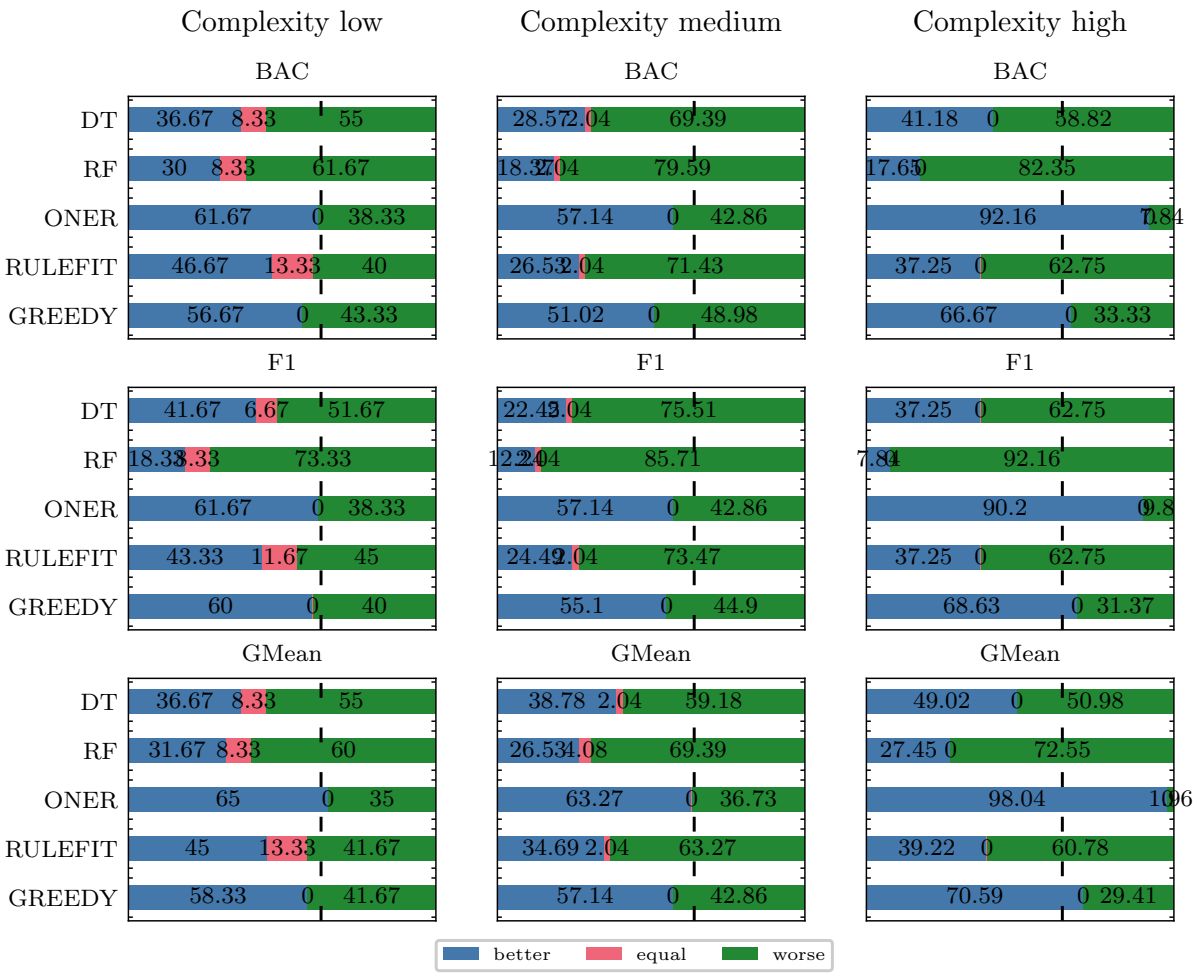

Figure 5.9: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics for explaining algorithm modification based on choosing trees. Vertical dashed lines indicate sign-test values.

than the overall distance between examples from the same class as indicated by the high amount of wins for high N2 complexity measures. In such cases RF and DT were outperformed in 47% and 52% of cases in terms of geometric mean measure. Fact, that it those algorithms are excelled in terms of balanced accuracy and geometric mean indicates, that the proposed algorithms is better in learning decision boundaries for underrepresented problems, as caused by big imbalance ratio

- data in which many features display overlapping, as indicated amount of wins in high F1 and F1V datasets

- in datasets where a moderate amount of original features are needed to describe data variability indicated by T4

- data in which there is a moderate amount of inter-class distance of samples, as indicated by performance in N3 metric

Those create opportunities for potential ML application designers to consider the proposed algorithm as a viable, interpretable alternative to RF and even to well-established DT.

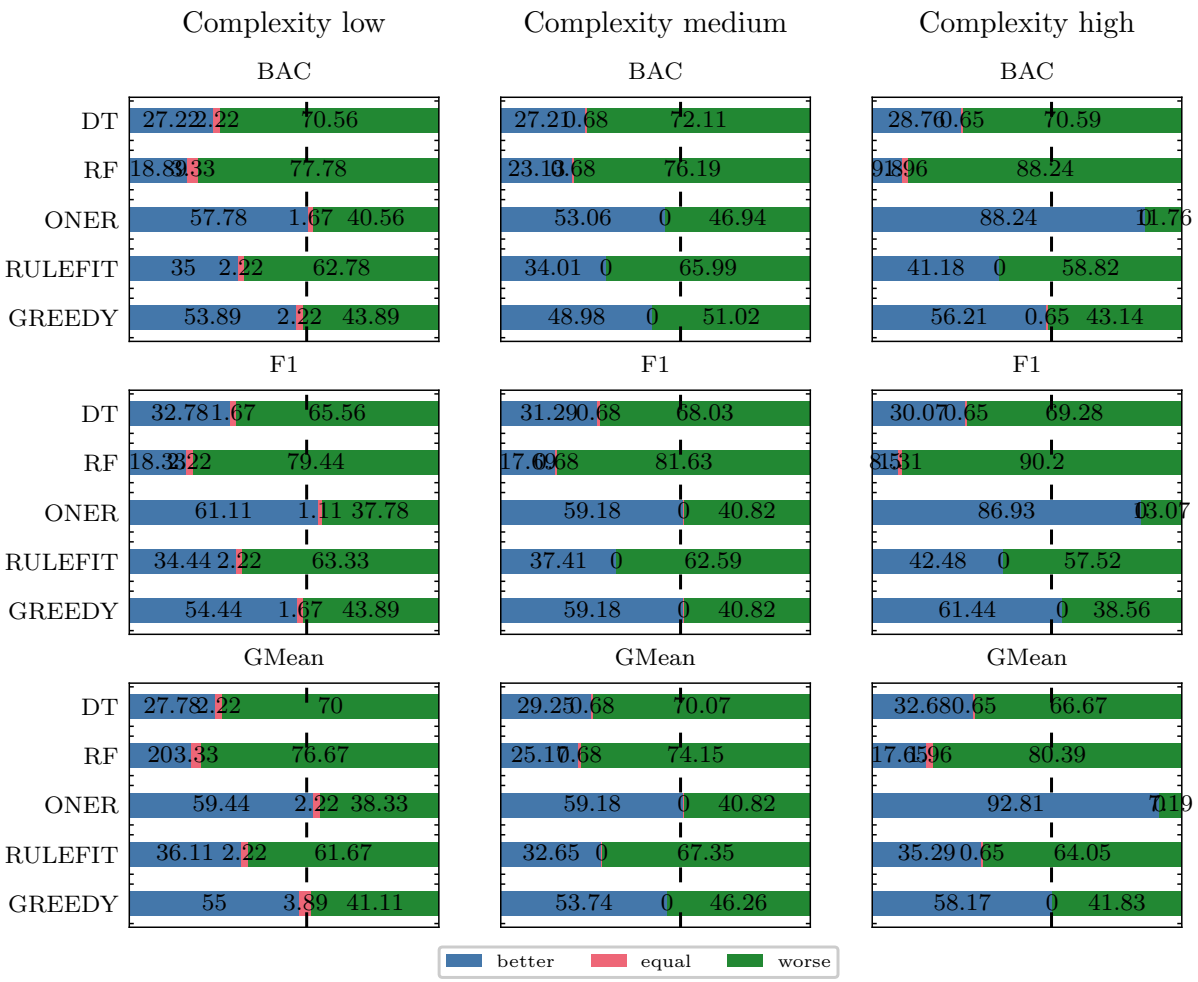

Figure 5.10: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics for explaining algorithm modification based on RF predictions. Vertical dashed lines indicate sign-test values.

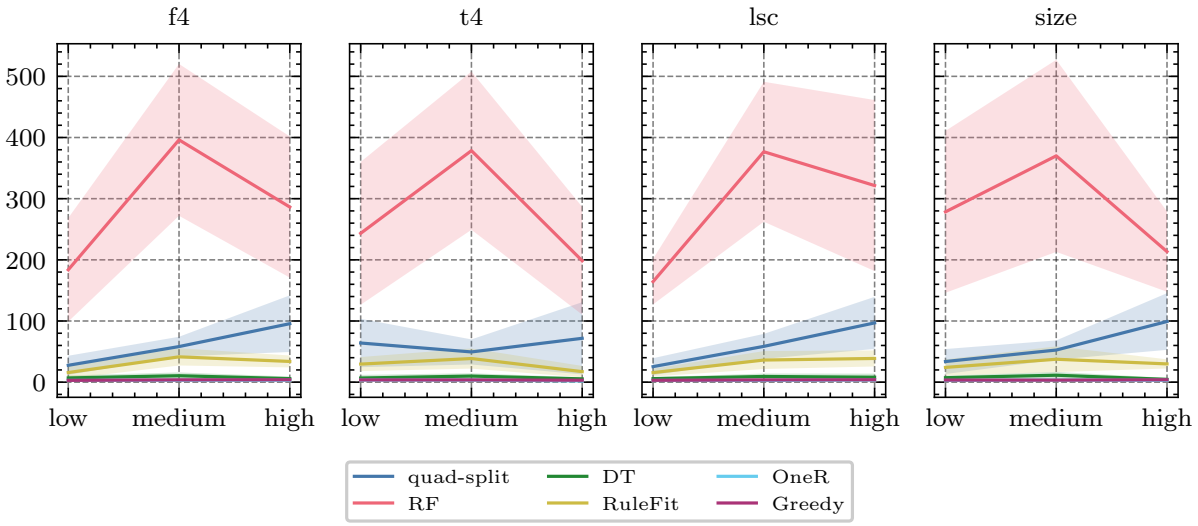

Figure 5.11: Change of internal model complexities in domain of dataset complexities for quad-split.

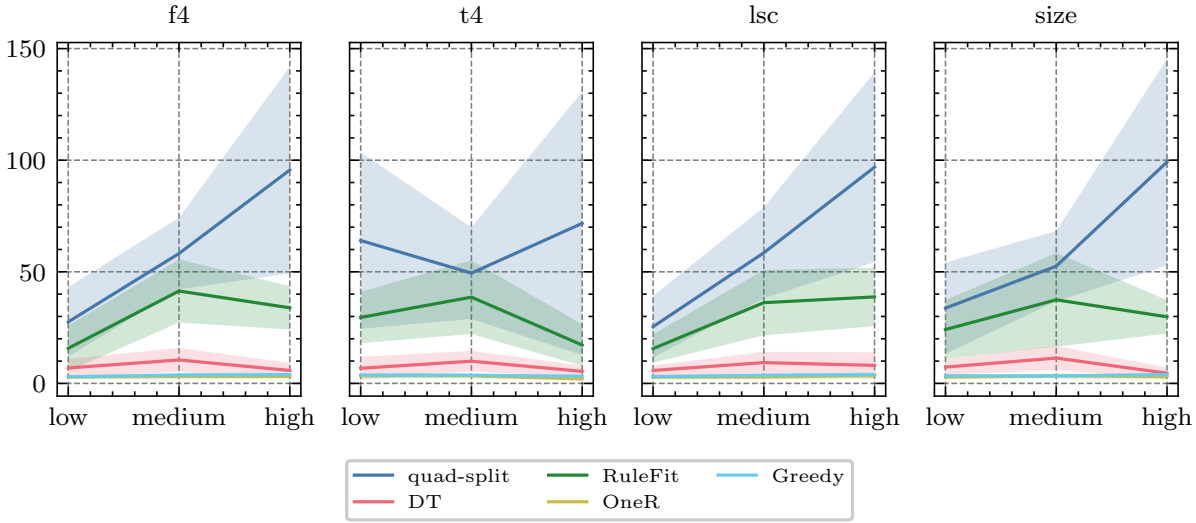

Figure 5.12: Change of internal model complexities in domain of dataset complexities for quad-split without RF.

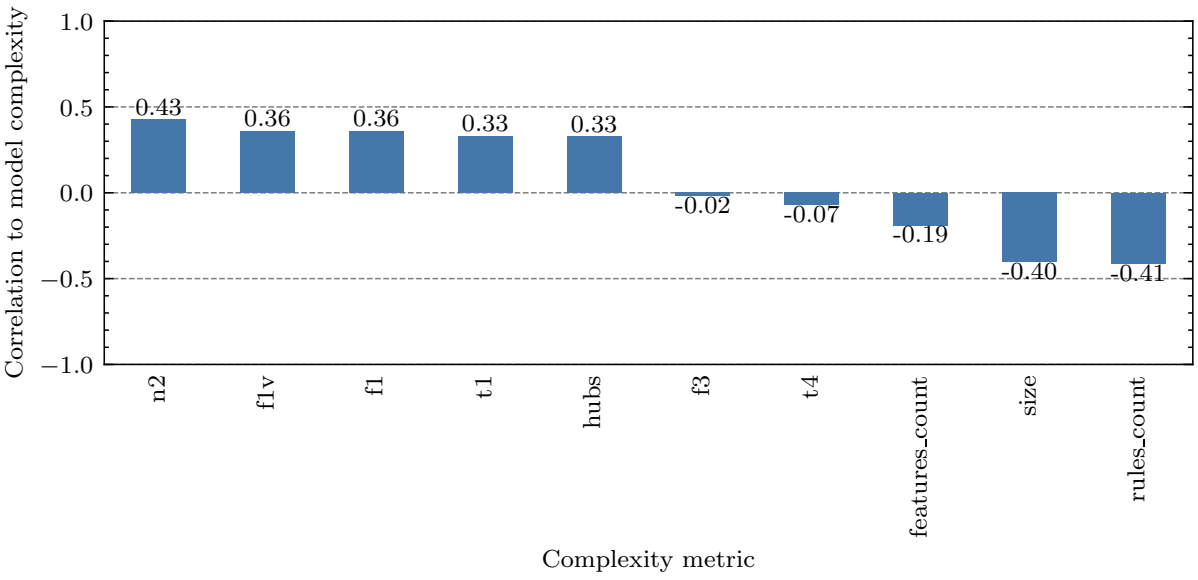

Figure 5.13: Correlation of complex to dummy classifiers proportion to dataset complexities.

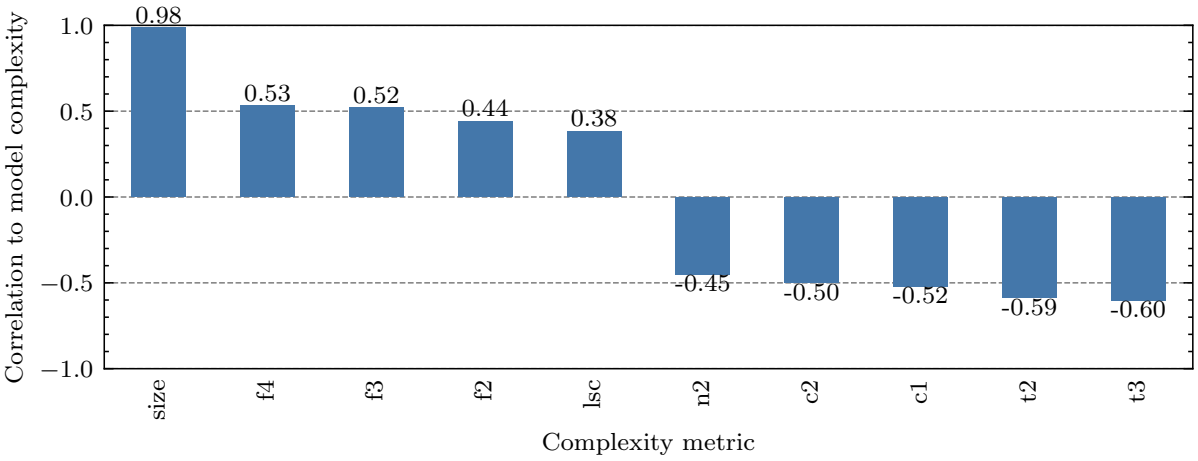

Figure 5.14: Correlation of number of rules to dataset complexities.

Additionally, computational complexity analysis shows that the algorithm is suitable for high-demand applications as it has relatively low training and prediction complexity. Its prediction complexity is based mostly on base classifier complexity, which provides great flexibility.

Last but not least, the model was tested as an explainer for RF algorithm. Two versions were present, where one would use RF's trees as classifiers, and the other one would just be trained at the forest's output. The first one significantly outperformed the latter, which indicates that extracting knowledge from the forest is a viable source of improving classification. Additionally, the forest was not only significantly simplified but was outperformed in numerous cases, most significant being those indicated by geometric mean for low – over 25%.

The proposed model complexity was also examined. It was shown that, although it is higher than all other explainable algorithms, it's still significantly lower than Random Forest while having the ability to outperform all tested models in some cases. Model behaved intuitively when it comes to changing its complexity in most of the tested cases but one. In a situation where the dataset displays a high ratio of features per point, model complexity decreases, which pinpoints possible areas of explainability application. It was shown that the number of rules decreases as dimensionality metrics rise, and it increases in the case of features overlapping. When it comes to model complexity displayed by a number of complex classifiers, it tends to decrease in favour of simple areas when the dataset has a big size, or the feature count is higher. On the other hand, there are more complex classifiers trained when features and decision areas are overlapping.

Overall, the algorithm shows a variety of promising features in different areas, making it not only competitive when it comes to standard classification but also as an explainer of RF.

Chapter 6

Comparison of proposed Explainable AI algorithms

6.1 Introduction

In the previous chapters, three XAI algorithms were introduced. While the first one – NOTE – was inherently designed for explaining RF. The other two – Optimal Centroids and Quad Split were designed as a way of building transparent models. For each of the latter two explainable variations were introduced in order to check feasibility of applying them as surrogate model explainer and to compare the results to NOTE.

- For Optimal Centroids:
  - Algorithm variation with tree selection, that would enhance GA's individual to include parameters for picking DT models out of all RF's trees and applying them for every constituent
  - Variation without tree selection which, for every constituent, would iterate over all RF's trees and pick the best one (as per accuracy)
- For Quad Split:
  - Algorithm version which, for every rule, assigns best (as per accuracy) tree from RF
  - Version which uses original algorithm and base models, but uses output of RF as training labels

Out of the proposed variations of Optimal Centroids method, one with tree selection was discarded as it has shown slightly better performance, which has been proven in previous chapters. The same goes for Quad Split and its version, which is trained on the output of RF predictions. As a result, in the following sections, the experimental evaluation of the three methods in explaining the same RF is presented.

6.2 Experimental evaluation

Three proposed methods were evaluated in-depth when it comes to their performance and complexity in previous chapters. In this one focus is put on evaluating them as each other. Following research questions till be evaluated:

RQ1 Which of the proposed methods performs best as RF explainer in terms of standard performance metrics?

RQ2 Are there significant differences when explaining RFs of different sizes?

RQ3 Which of the explaining approaches creates most complex internal model?

6.2.1 Setup

In this section approach to experimental evaluation and environment details are described.

To provide common ground for testing all three methods, following assumptions were made:

- All algorithms would be tested as explainers, and against, same RF of equal sizes - 3, 5, 7
- Every possible randomness (coming for example from inherent RF bagging randomness or GA individuals sampling) would be mitigated by using same pseudo-random number generators with same seed

Choice of datasets A series of experiments were conducted across selected binary classification datasets. As the source for those Knowledge Extraction based on Evolutionary Learning repository was used [97]. Datasets were pre-processed beforehand. Labels were encoded into integers. Finally, to reasonably limit scope of experiments, 16 datasets were randomly chosen. Although there is no consensus about a minimal amount of datasets when comparing classification algorithms, such an amount is widely considered by the research community as standard and good practice. Details of used datasets are in Table 2.1. Each dataset was shuffled and split according to the 5x2 CV experimental protocol. Reasons behind such choice were described in chapter 2.

Measuring models complexity In order to objectively compare trained model complexities, they needed to be quantified into numerical values. For the sake of this experiment, the pattern of assigning numerical values to models present in Table 6.1 was used. In the case of Optimal Centroids the 1-NN step was excluded as there is no unified way to measure its complexity.

Table 6.1: Internal models complexity computation details

model	complexity formula
Quad Split	$\sum^{\text{trees}}$ +# of rules
Optimal Centroids	$\sum^{\text{trees}}$ # of leaves
NOTE	# of leaves + # of rules

Base algorithms For comparison, state-of-the-art RF with DT in CART implementation [109] were used.

Measuring models complexity In order to objectively compare trained model complexities, they needed to be quantified into numerical values. For sake of this experiment, the following pattern of assigning numerical values to models were used:

- **Quad Split** - $\sum^{\text{trees}}$ +# of rules
- **Optimal centroids** - $\sum^{\text{trees}}$ # of leaves
- **NOTE** - # of leaves + # of rules
- **RF** - $\sum^{\text{trees}}$ # of leaves

Implementation and reproducibility. The described methods were implemented using *Python 3.8* programming language. *scikit-learn*[43] implementation of DT and RF base models was used. Slightly modified in terms of performance complexity metric implementation based on Problexity library [58] was used. Details of implementation for specific algorithms can be found in previous chapters. Experiments were run using *Apple MacBook Pro M1 Max* with 32GB RAM.

Model parameters selection. The subject and base algorithms were pretrained on subset of datasets to choose parameters from. Four datasets for pretraining procedure were randomly selected. Grid Search procedure was used for optimizing along with for parameters. Special care was taken for alpha parameters of both CART and RuleFit algorithms. As it follows particular pattern, exponential distribution was used for sampling those with scale of 0.1. Then, best parameters for each dataset were taken and average for continuous or mode for discrete parameters was taken as final set. DT parameters were propagated to RFs' base models. For consistency, the proposed algorithm also used pre-trained decision tree parameters for the base classifiers.

In the following sections, results leading to answering research questions are presented.

6.2.2 Which of the proposed methods performs best as RF explainer in terms of standard performance metrics?

Figure 6.2 presents performance metrics in domain of different size of explained forest ensemble. As in previous Chapters, three metrics were used - balanced accuracy, f1 score

Table 6.2: Pre-train results for the compared methods.

Algorithm	Parameter	Value	Distribution	Iterations
GA	n_gen	100		1000
	pop_size	25		
Quad Split	min_samples	28	betabinom(65, 2.578, 5.106)	1000
	min_split_percentage	0.183	uniform(0.01, 0.49)	
DT	ccp_alpha	0.0115	exponential(0.1)	10000
	criterion	gini		
	max_features	None		
RF	size	32	range(1,100)	1000
	ccp_alpha	0.0115		
	criterion	gini		
	max_features	None		

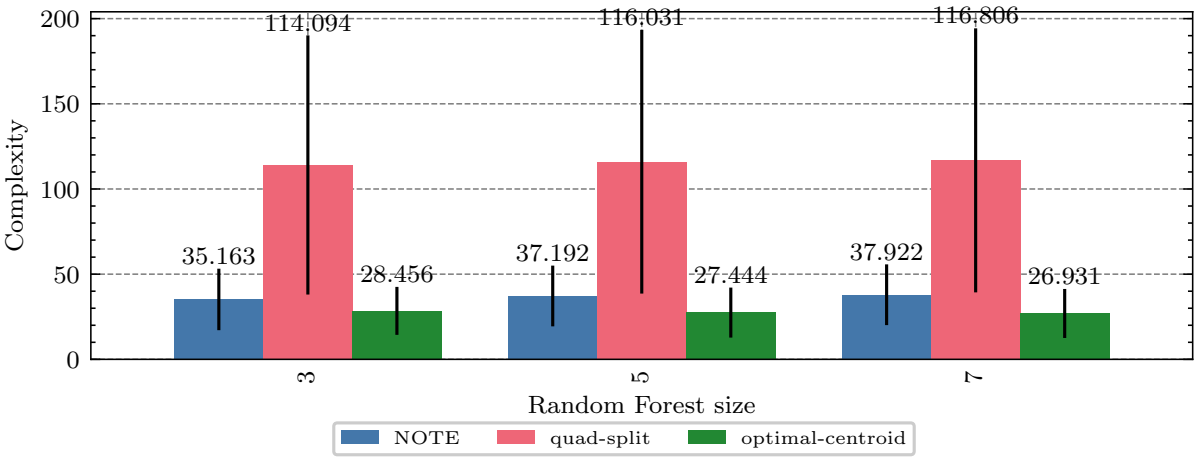

Figure 6.1: Averaged model complexities for different RF sizes.

and geometric mean. It can be observed that NOTE outperforms two other algorithms in every scenario and every metric. When it comes to figuring out winner between the other two – in most cases, apart from F1 score, quad split has edge over optimal centroids.

Above mentioned observations find its clarifications in amount of wins, draws and losses against explained Random Forest, as depicted in Figure 6.3. In this case however, amount of wins as compared between optimal-centroid and quad-split points out the former as a clear winner. It is able to outperform Random Forest in 32% and 28% of cases when it comes to F1 score and forests of size 3 and 5, where quad-split falls short by even 7%.

6.2.3 Are there significant differences when explaining RFs of different sizes?

When it comes to change in domain of RF size, the algorithms display steady pattern of decreasing performance as the size increases. The most significant drop occurs with the best performing algorithm – NOTE – where the percentage of wins for BAC would drop by 10% as the forest would grow from 3 to 7. Optimal Centroids would note the lowest drop, by less than 7%. What is interesting is while NOTE, as depicted by Figure 6.2, average metric values increase for Quad Split and Optimal Centroids, but keep stable with RF size for NOTE. As can be also observed and which is expected, the forest performs better with increase with its size – the amount of wins over methods increases, while NOTE metric values did not.

6.2.4 Which of the explaining approaches creates the most complex internal model?

When it comes to overall final model complexity, as displayed in Figure 6.2, Quad Split algorithm, for every size of explained forest, produces most complex models – even by factor of three. Lowest complexity models are produced consequently by optimal-centroids. Although it is worth remembering, that complexity calculation for it does not include existing within KNN model used for constituent selection. What is also noticeable, is that Quad Split has the biggest variance, meaning that the created models complexity was much less predictable and would not correlate with model performance.

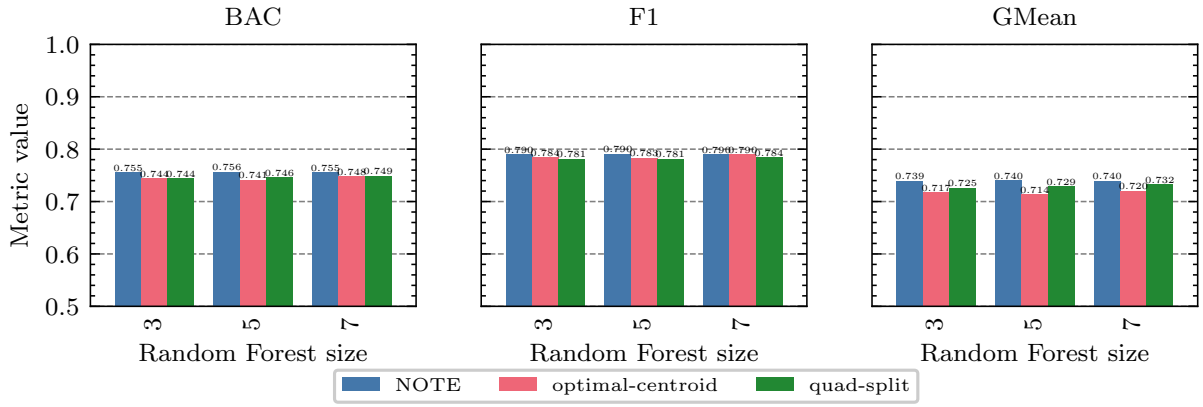

Figure 6.2: Comparison of RF size influence on models performance. The numbers above the bars indicate metric values.

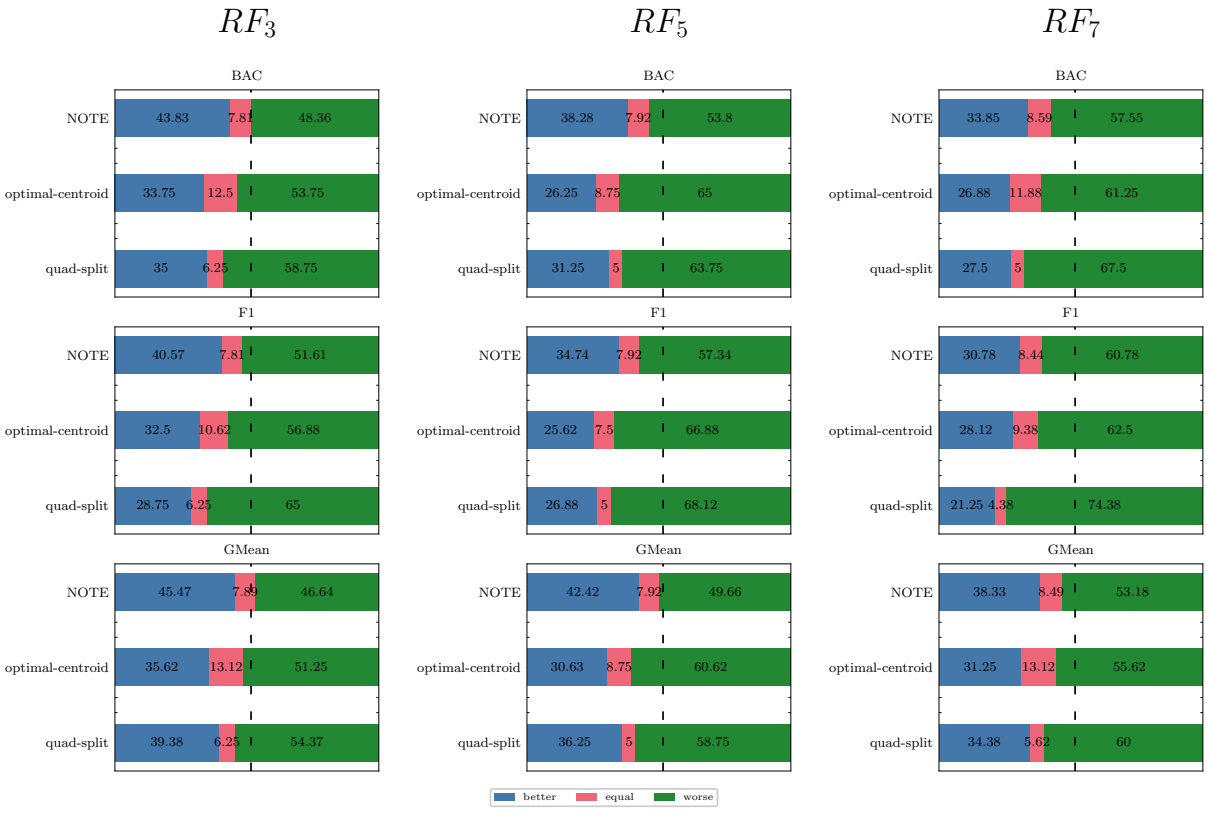

Figure 6.3: Overall percentage of wins, ties and losses counted over all datasets for BAC, F1 and GMean metrics for different RF sizes. Vertical dashed lines indicate sign-test values.

6.3 Summary and lessons learned

Three methods for explaining Random Forest were examined in this chapter. One inherently designed for explaining Random Forest – NOTE – with variations of transparent model algorithms presented previously.

What's noteworthy is that, in comparison, NOTE – the method that was natively designed for explaining forest ensemble, was able to outperform it most of the time out of all tested approaches. At the same time, stale of NOTE model performance was noticed as RF size and generalizing abilities increased. It would indicate that the algorithm would be unable to extract all knowledge achieved by the forest or was bound by the ability to find the best clique. As the drop of metrics Quad Split and Optimal Centroids was less than this of NOTE, it can be theorized that at some arbitrary forest size, the models would grow closer in generalizing abilities or even swap in rankings. The last observation is that the proposed model complexity does not necessarily transfer to predictive abilities, as was shown by Quad Split. Its complexity was multiple bigger than this of other models – even though it was not able to outperform NOTE. Finally, the model utilizing the actual structure of RF performed the best, which can act as an indicator in further development and design of XAI models.

Chapter 7

Conclusion and future research

This thesis touched upon three critical AI areas: supervised classification, explainability and interpretability and data complexity measures. This thesis aimed to elaborate on and support or dismiss the hypothesis stating that:

For a given classification task, it is possible to build such a transparent or explainable model, whose quality is not worse than the similarly applied black-box model.

Four main objectives were formulated to acknowledge or drop the claim. As a result of research conducted and described in past chapters, those targets were completed, namely:

- **Development of novel ensemble "glass-box" model extraction method** and **Development of novel transparent, "white-box" models** – Three novel algorithms were proposed:
  - **NOTE** - which extracts interpretable model from RF by using graph modelling of rules and scoring cliques
  - **Optimal Centroids** - which utilises a genetic algorithm in building an intepretable model that splits decision area into constituents by nearest neighbour approach and then assigns each constituents interpretable base model
  - **Quad Split** - that splits recursively decision space, in DT manner, by using complexity measures, and then assigns each fold (described by rule) interpretable base model
- **Utilisation of data complexity metrics in developing novel transparent classification method** – metrics were used as inherent mechanism in Quad Split algorithm
- **Experimental evaluation of proposed algorithms using wide array of datasets attributed with different complexities** – methods were evaluated against each other as well as commonly used classification algorithms, which was subject of this thesis
- **Creation of programming library** – implementation of every method – NOTE[1], Optimal Centroids[2] and Quad Split[3] – were released as open-source *Python* code bundle along with additional utilitity library[4]

[1] `https://github.com/bgulowaty/non-overlapping-rules-ensemble`
[2] `https://github.com/bgulowaty/optimal-centroids`
[3] `https://github.com/bgulowaty/quad-split`
[4] `https://github.com/bgulowaty/ml-utils`

While the first proposed method inherently works as an explainer of RF, the other two can be used as independent, transparent classification models. Nevertheless, two variations of each serving as RF explainer were proposed and examined.

Every algorithm's predictive abilities were verified multifold:

- by testing its significant configuration parameters
- by verifying its behavior in a quantitative manner over all experiments and comparing it to other models
- by verifying its performance in domain of different dataset complexities, which shed light on potential areas of applications
- by examining model complexities and comparing it to others numerically

The stated hypothesis was found to be true during an examination of the thesis. As shown in previous chapters, experimental results supported the above-mentioned. In the process of evaluating every proposed algorithm, it was shown that there were numerous examples in which the built model would outperform RF, as well as other interpretable models. Although the phenomena were not constant, it was seen that there are certain properties of the dataset that favour applying proposed transparent models over complex ones without losing quality. During research following additional observations were made:

- **Evaluated classification models behave immensely differently when applied in datasets with different complexity properties**. During examination of proposed algorithms, their performance was evaluated for different datasets, which had assigned different complexity labels. It was shown that the same model can behave vastly differently, in terms of competing with another model, when applied to datasets with high and low values of some complexity metric. This indicates different application areas for specific models, and – what was observed – renders more complex models less usable in some scenarios.
- **Intrinsic model complexities do not always follow dataset complexities**. By analysing quantified internal model complexities (as expressed, for example, by a number of rules), it was shown that even though some complexity metrics were describing the dataset as complex, it did not reflect in trained model complexity. It was true not only for specific metrics but overall dataset complexity as quantified by mean of all complexities.
- **Different complexity metrics are better predictors of model performance**. As shown during the evaluation of each method, some complexity measures described bigger potential when predicting wins and losses. This indicates that some of them may be applied as potential meta-learners or predictor of model performance
- **Transparent model that used knowledge extracted from complex model behaved better than inherently transparent ones**. As per the comparison presented in the previous chapter, it was found that NOTE algorithm had the best performance. This leads to the conclusion that, at least in the tested sizes of RF, developing an explaining algorithm that utilises the internal structure of the explained model may bring better value than using just the black-box predictions

7.1 Future work

The methods and observations subjected in this work may serve as a base for further research. Various aspects can be pursued, method-specific as well as general ones, such as:

- Developing more fine-grained scoring metrics for NOTE, which would allow extracting subspaces raising results closer to theoretical maximum
- Extending optimal-centroid methods into using constituen selector, that is more interpretable than *k-nearest neighbours* algorithm
- Fine-tuning optimal-centroid's genetic algorithm parameters and verify, how often it is being stuck in local optima
- Verify all of the above as explainers for other models, such as neural networks or other ensemble models
- Develop more fine-grained methods of assigning base classifiers into constituents of each methods, such as using heterogenous approach
- Use observations made in this thesis to train meta-learning model, that would assess and pick a specific model based on dataset complexity

7.2 Publications

During process of working on this thesis, the author has published following articles:

- B. Gulowaty and P. Ksieniewicz, "Smote algorithm variations in balancing data streams", in *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, H. Yin, D. Camacho, P. Tino, *et al.*, Eds., Cham: Springer International Publishing, 2019, pp. 305–312, ISBN: 978-3-030-33617-2
  CORE C, **MNiSW**: 20
- B. Gulowaty and M. Woźniak, "Extracting interpretable decision tree ensemble from random forest", in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533601
  CORE A, **MNiSW**: 140
- B. Gulowaty and M. Woźniak, "Search-based framework for transparent non-overlapping ensemble models", in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–6. DOI: 10.1109/IJCNN55064.2022.9892360
  CORE A, **MNiSW**: 140

Bibliography

[1] D. Reinsel, J. Gantz, and J. Rydning, "The Digitization of the World", en, 2020.

[2] W. Bedingfield, *Generative ai is playing a surprising role in israel-hamas disinformation*, Oct. 2023. [Online]. Available: https://www.wired.com/story/israel-hamas-war-generative-artificial-intelligence-disinformation/.

[3] M. Choras, K. Demestichas, A. Gielczyk, *et al.*, *Advanced Machine Learning Techniques for Fake News (Online Disinformation) Detection: A Systematic Mapping Study*, en, arXiv:2101.01142 [cs], Dec. 2020. [Online]. Available: http://arxiv.org/abs/2101.01142 (visited on 09/07/2024).

[4] M. T. Hošman, "Richard Baldwin: The Globotics Upheaval: Globalisation, Robotics, and the Future of Wor.k", *Czech Journal of International Relations*, vol. 55, no. 2, pp. 65–69, Jun. 2020, ISSN: 2570-9429, 0323-1844. DOI: 10.32422/mv.1695. [Online]. Available: https://cjir.iir.cz/index.php/cjir/article/view/91 (visited on 09/07/2024).

[5] Sedat Sonko, Adebunmi Okechukwu Adewusi, Ogugua Chimezie Obi, Shedrack Onwusinkwue, and Akoh Atadoga, "A critical review towards artificial general intelligence: Challenges, ethical considerations, and the path forward", *World Journal of Advanced Research and Reviews*, vol. 21, no. 3, pp. 1262–1268, Mar. 2024, ISSN: 25819615. DOI: 10.30574/wjarr.2024.21.3.0817. [Online]. Available: https://wjarr.com/content/critical-review-towards-artificial-general-intelligence-challenges-ethical-considerations (visited on 09/07/2024).

[6] R. Fjelland, "Why general artificial intelligence will not be realized", en, *Humanities and Social Sciences Communications*, vol. 7, no. 1, p. 10, Jun. 2020, ISSN: 2662-9992. DOI: 10.1057/s41599-020-0494-4. [Online]. Available: https://www.nature.com/articles/s41599-020-0494-4 (visited on 09/07/2024).

[7] V. C. Müller, "Ethics of Artificial Intelligence and Robotics", in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta and U. Nodelman, Eds., Fall 2023, Metaphysics Research Lab, Stanford University, 2023.

[8] A. Das and P. Rad, *Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey*, en, arXiv:2006.11371 [cs], Jun. 2020. [Online]. Available: http://arxiv.org/abs/2006.11371 (visited on 03/17/2024).

[9] B. C. Stahl and D. Wright, "Ethics and Privacy in AI and Big Data: Implementing Responsible Research and Innovation", en, *IEEE Security & Privacy*, vol. 16, no. 3, pp. 26–33, May 2018, ISSN: 1540-7993, 1558-4046. DOI: 10.1109/MSP.2018.2701164. [Online]. Available: https://ieeexplore.ieee.org/document/8395078/ (visited on 03/17/2024).

[10] R. Challen, J. Denny, M. Pitt, L. Gompels, T. Edwards, and K. Tsaneva-Atanasova, "Artificial intelligence, bias and clinical safety", en, *BMJ Quality & Safety*, vol. 28, no. 3, pp. 231–237, Mar. 2019, ISSN: 2044-5415, 2044-5423. DOI: 10.1136/bmjqs-2018-008370. [Online]. Available: https://qualitysafety.bmj.com/lookup/doi/10.1136/bmjqs-2018-008370 (visited on 03/17/2024).

[11] A. Mikołajczyk, M. Grochowski, and A. Kwasigroch, "Towards Explainable Classifiers Using the Counterfactual Approach - Global Explanations for Discovering Bias in Data", en, *Journal of Artificial Intelligence and Soft Computing Research*, vol. 11, no. 1, pp. 51–67, Jan. 2021, ISSN: 2083-2567. DOI: 10.2478/jaiscr-2021-0004. [Online]. Available: https://www.sciendo.com/article/10.2478/jaiscr-2021-0004 (visited on 03/17/2024).

[12] K. Kirkpatrick, "It's not the algorithm, it's the data", en, *Communications of the ACM*, vol. 60, no. 2, pp. 21–23, Jan. 2017, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3022181. [Online]. Available: https://dl.acm.org/doi/10.1145/3022181 (visited on 03/17/2024).

[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and Harnessing Adversarial Examples*, en, arXiv:1412.6572 [cs, stat], Mar. 2015. [Online]. Available: http://arxiv.org/abs/1412.6572 (visited on 03/17/2024).

[14] B. Goodman and S. Flaxman, "European Union regulations on algorithmic decision-making and a "right to explanation"", en, *AI Magazine*, vol. 38, no. 3, pp. 50–57, Sep. 2017, arXiv:1606.08813 [cs, stat], ISSN: 0738-4602, 2371-9621. DOI: 10.1609/aimag.v38i3.2741. [Online]. Available: http://arxiv.org/abs/1606.08813 (visited on 03/17/2024).

[15] P. Biecek and T. Burzykowski, *Explanatory model analysis: Explore, explain and examine predictive models*. Chapman and Hall/CRC, 2021.

[16] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", en, *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020, ISSN: 15662535. DOI: 10.1016/j.inffus.2019.12.012. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253519308103 (visited on 09/14/2020).

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier", KDD '16, pp. 1135–1144, 2016. DOI: 10.1145/2939672.2939778. [Online]. Available: https://doi.org/10.1145/2939672.2939778.

[18] S. Lundberg and S.-I. Lee, *A Unified Approach to Interpreting Model Predictions*, arXiv:1705.07874 [cs, stat], Nov. 2017. [Online]. Available: http://arxiv.org/abs/1705.07874 (visited on 09/17/2024).

[19] O. Sagi and L. Rokach, "Explainable decision forest: Transforming a decision forest into an interpretable tree", en, *Information Fusion*, vol. 61, pp. 124–138, Sep. 2020, ISSN: 15662535. DOI: 10.1016/j.inffus.2020.03.013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253519307869 (visited on 11/02/2020).

[20] K. Simonyan, A. Vedaldi, and A. Zisserman, *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*, en, arXiv:1312.6034 [cs], Apr. 2014. [Online]. Available: http://arxiv.org/abs/1312.6034 (visited on 03/17/2024).

[21] A. Adadi and M. Berrada, "Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI)", en, *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2870052. [Online]. Available: https://ieeexplore.ieee.org/document/8466590/ (visited on 05/17/2020).

[22] R. Dwivedi, D. Dave, H. Naik, *et al.*, "Explainable AI (XAI): Core Ideas, Techniques, and Solutions", en, *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–33, Sep. 2023, ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3561048. [Online]. Available: https://dl.acm.org/doi/10.1145/3561048 (visited on 09/12/2024).

[23] W. Saeed and C. Omlin, "Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities", en, *Knowledge-Based Systems*, vol. 263, p. 110 273, Mar. 2023, ISSN: 09507051. DOI: 10.1016/j.knosys.2023.110273. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950705123000230 (visited on 09/12/2024).

[24] J. Górriz, I. Álvarez-Illán, A. Álvarez-Marquina, *et al.*, "Computational approaches to Explainable Artificial Intelligence: Advances in theory, applications and trends", en, *Information Fusion*, vol. 100, p. 101 945, Dec. 2023, ISSN: 15662535. DOI: 10.1016/j.inffus.2023.101945. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1566253523002610 (visited on 10/06/2024).

[25] C. Rudin, *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead*, May 2019. DOI: 10.1038/s42256-019-0048-x.

[26] S. Thiebes, S. Lins, and A. Sunyaev, "Trustworthy artificial intelligence", en, *Electronic Markets*, vol. 31, no. 2, pp. 447–464, Jun. 2021, ISSN: 1019-6781, 1422-8890. DOI: 10.1007/s12525-020-00441-4. [Online]. Available: https://link.springer.com/10.1007/s12525-020-00441-4 (visited on 09/16/2024).

[27] B. Goodman and S. Flaxman, "European Union regulations on algorithmic decision-making and a "right to explanation"", 2016, Publisher: arXiv Version Number: 3. DOI: 10.48550/ARXIV.1606.08813. [Online]. Available: https://arxiv.org/abs/1606.08813 (visited on 09/16/2024).

[28] T. J. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction* (Springer series in statistics), eng, 2nd ed. New York: Springer, 2009, ISBN: 978-0-387-84858-7.

[29] O. Sagi and L. Rokach, "Ensemble learning: A survey", en, *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, e1249, Jul. 2018, ISSN: 1942-4787, 1942-4795. DOI: 10.1002/widm.1249. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1249 (visited on 03/17/2024).

[30] R. Polikar, "Ensemble based systems in decision making", en, *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006, ISSN: 1531-636X. DOI: 10.1109/MCAS.2006.1688199. [Online]. Available: http://ieeexplore.ieee.org/document/1688199/ (visited on 04/11/2024).

[31] Y. Song, "Ensemble reinforcement learning: A survey", en, *Applied Soft Computing*, 2023.

[32] J. O. Mendes-Moreira, L.-I. Tec, C. Soares, A. M. R. Jorge, L.-I. Tec, and J. F. D. Sousa, "Ensemble approaches for regression: A survey", en, *ACM Computing Surveys*, vol. 45, no. 1,

[33] B. Krawczyk, "Ensemble learning for data stream analysis: A survey", en, *Information Fusion*, 2017.

[34] L. Breiman, "Bagging predictors", *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, ISSN: 0885-6125. DOI: 10.1023/A:1018054314350. [Online]. Available: https://doi.org/10.1023/A:1018054314350.

[35] Y. Cao, Q.-G. Miao, J.-C. Liu, and L. Gao, "Advance and Prospects of AdaBoost Algorithm", en, *Acta Automatica Sinica*, vol. 39, no. 6, pp. 745–758, Jun. 2013, ISSN: 18741029. DOI: 10.1016/S1874-1029(13)60052-X. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S187410291360052X (visited on 09/07/2024).

[36] B. Pavlyshenko, "Using stacking approaches for machine learning models", pp. 255–258, Aug. 2018. DOI: 10.1109/DSMP.2018.8478522.

[37] M. Hossin and S. M.N, "A review on evaluation metrics for data classification evaluations", *International Journal of Data Mining & Knowledge Management Process*, vol. 5, pp. 01–11, Mar. 2015. DOI: 10.5121/ijdkp.2015.5201.

[38] Z. Chen, L. D. Van Khoa, E. N. Teoh, A. Nazir, E. K. Karuppiah, and K. S. Lam, "Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: A review", en, *Knowledge and Information Systems*, vol. 57, no. 2, pp. 245–285, Nov. 2018, ISSN: 0219-1377, 0219-3116. DOI: 10.1007/s10115-017-1144-z. [Online]. Available: http://link.springer.com/10.1007/s10115-017-1144-z (visited on 07/30/2024).

[39] D. Brzezinski, J. Stefanowski, R. Susmaga, and I. Szczech, "On the Dynamics of Classification Measures for Imbalanced and Streaming Data", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2868–2878, Aug. 2020, Publisher: Institute of Electrical and Electronics Engineers (IEEE), ISSN: 2162-237X, 2162-2388. DOI: 10.1109/tnnls.2019.2899061. [Online]. Available: https://ieeexplore.ieee.org/document/8668688/ (visited on 09/17/2024).

[40] J. R. Quinlan, "Induction of decision trees", en, *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1986, ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00116251. [Online]. Available: http://link.springer.com/10.1007/BF00116251 (visited on 02/18/2024).

[41] B. Charbuty and A. Abdulazeez, "Classification Based on Decision Tree Algorithm for Machine Learning", en, *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, ISSN: 2708-0757. DOI: 10.38094/jastt20165. [Online]. Available: https://jastt.org/index.php/jasttpath/article/view/65 (visited on 09/08/2024).

[42] G. Biau and E. Scornet, "A random forest guided tour", en, *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016, ISSN: 1133-0686, 1863-8260. DOI: 10.1007/s11749-016-0481-7. [Online]. Available: http://link.springer.com/10.1007/s11749-016-0481-7 (visited on 10/03/2021).

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[44] K. Dembczyński, W. Kotłowski, and R. Słowiński, "Maximum likelihood rule ensembles", en, in *Proceedings of the 25th international conference on Machine learning - ICML '08*, Helsinki, Finland: ACM Press, 2008, pp. 224–231, ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390185. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1390156.1390185 (visited on 09/08/2024).

[45] J. H. Friedman and B. E. Popescu, "Predictive learning via rule ensembles", en, *The Annals of Applied Statistics*, vol. 2, no. 3, Sep. 2008, arXiv:0811.1679 [stat], ISSN: 1932-6157. DOI: 10.1214/07-AOAS148. [Online]. Available: http://arxiv.org/abs/0811.1679 (visited on 09/08/2024).

[46] M. Li and P. M. Vitányi, "Kolmogorov Complexity and its Applications", en, in *Algorithms and Complexity*, Elsevier, 1990, pp. 187–254, ISBN: 978-0-444-88071-0. DOI: 10.1016/B978-0-444-88071-0.50009-6. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780444880710500096 (visited on 04/14/2024).

[47] M. Gell-Mann, "What Is Complexity?", in *Complexity and Industrial Clusters*, A. Q. Curzio and M. Fortis, Eds., Heidelberg: Physica-Verlag HD, 2002, pp. 13–24, ISBN: 978-3-642-50007-7.

[48] J. Maciejowski, "Model discrimination using an algorithmic information criterion", en, *Automatica*, vol. 15, no. 5, pp. 579–593, Sep. 1979, ISSN: 00051098. DOI: 10.1016/0005-1098(79)90006-2. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0005109879900062 (visited on 04/14/2024).

[49] Tin Kam Ho and M. Basu, "Complexity measures of supervised classification problems", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 289–300, Mar. 2002, ISSN: 01628828. DOI: 10.1109/34.990132. [Online]. Available: http://ieeexplore.ieee.org/document/990132/ (visited on 04/14/2024).

[50] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning", *Artificial Intelligence Review*, vol. 18, Sep. 2001. DOI: 10.1023/A:1019956318069.

[51] I. Khan, X. Zhang, M. Rehman, and R. Ali, "A Literature Survey and Empirical Study of Meta-Learning for Classifier Selection", *IEEE Access*, vol. 8, pp. 10 262–10 281, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2964726. [Online]. Available: https://ieeexplore.ieee.org/document/8951014/ (visited on 04/15/2024).

[52] A. Lorena, L. P. Garcia, J. Lehmann, M. de Souto, and T. Ho, "How complex is your classification problem?: A survey on measuring classification complexity", *ACM Computing Surveys*, vol. 52, pp. 1–34, Sep. 2019. DOI: 10.1145/3347711.

[53] L. Bottou and C.-J. Lin, "Support Vector Machine Solvers", en, in *Large-Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds., The MIT Press, Aug. 2007, pp. 1–28, ISBN: 978-0-262-25579-0. DOI: 10.7551/mitpress/7496.003.0003. [Online]. Available: https://direct.mit.edu/books/book/3172/chapter/88087/Support-Vector-Machine-Solvers (visited on 06/06/2024).

[54] J. C. Gower, "A General Coefficient of Similarity and Some of Its Properties", *Biometrics*, vol. 27, no. 4, p. 857, Dec. 1971, ISSN: 0006341X. DOI: 10.2307/2528823. [Online]. Available: https://www.jstor.org/stable/2528823?origin=crossref (visited on 06/10/2024).

[55] A. Kershenbaum and R. Van Slyke, "Computing minimum spanning trees efficiently", en, in *Proceedings of the ACM annual conference on - ACM'72*, vol. 1, Boston, Massachusetts, United States: ACM Press, 1972, p. 518. DOI: 10.1145/800193.569966. [Online]. Available: http://portal.acm.org/citation.cfm?doid=800193.569966 (visited on 06/10/2024).

[56] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes", en, *Pattern Recognition*, vol. 44, no. 8, pp. 1761–1776, Aug. 2011, ISSN: 00313203. DOI: 10.1016/j.patcog.2011.01.017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0031320311000458 (visited on 09/28/2024).

[57] E. Leyva, A. Gonzalez, and R. Perez, "A Set of Complexity Measures Designed for Applying Meta-Learning to Instance Selection", *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 2, pp. 354–367, Feb. 2015, ISSN: 1041-4347. DOI: 10.1109/TKDE.2014.2327034. [Online]. Available: http://ieeexplore.ieee.org/document/6823733/ (visited on 06/10/2024).

[58] J. Komorniczak and P. Ksieniewicz, *Problexity – an open-source Python library for binary classification problem complexity assessment*, Version Number: 1, 2022. DOI: 10.48550/ARXIV.2207.06709. [Online]. Available: https://arxiv.org/abs/2207.06709 (visited on 06/15/2024).

[59] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (PCA)", en, *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, Mar. 1993, ISSN: 00983004. DOI: 10.1016/0098-3004(93)90090-R. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/009830049390090R (visited on 06/15/2024).

[60] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation", en, *Information Sciences*, vol. 513, pp. 429–441, Mar. 2020, ISSN: 00200255. DOI: 10.1016/j.ins.2019.11.004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025519310497 (visited on 09/12/2024).

[61] A. K. Tanwani and M. Farooq, "Classification Potential vs. Classification Accuracy: A Comprehensive Study of Evolutionary Algorithms with Biomedical Datasets", in *Learning Classifier Systems*, J. Bacardit, W. Browne, J. Drugowitsch, E. Bernadó-Mansilla, and M. V. Butz, Eds., vol. 6471, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 127–144, ISBN: 978-3-642-17507-7 978-3-642-17508-4. DOI: 10.1007/978-3-642-17508-4_9. [Online]. Available: http://link.springer.com/10.1007/978-3-642-17508-4_9 (visited on 06/15/2024).

[62] R. Marcinkevičs and J. E. Vogt, *Interpretability and Explainability: A Machine Learning Zoo Mini-tour*, en, arXiv:2012.01805 [cs], Mar. 2023. [Online]. Available: http://arxiv.org/abs/2012.01805 (visited on 01/31/2024).

[63] F. Doshi-Velez and B. Kim, "Towards A Rigorous Science of Interpretable Machine Learning", en, *arXiv:1702.08608 [cs, stat]*, Mar. 2017, arXiv: 1702.08608. [Online]. Available: http://arxiv.org/abs/1702.08608 (visited on 05/17/2020).

[64] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.", *Queue*, vol. 16, no. 3, pp. 31–57, Jun. 2018, ISSN: 1542-7730. DOI: 10.1145/3236386.3241340. [Online]. Available: https://doi.org/10.1145/3236386.3241340.

[65] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models", en, *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, Apr. 2011, ISSN: 01679236. DOI: 10.1016/j.dss.2010.12.003. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167923610002368 (visited on 02/03/2024).

[66] A. F. Markus, J. A. Kors, and P. R. Rijnbeek, "The role of explainability in creating trustworthy artificial intelligence for health care: A comprehensive survey of the terminology, design choices, and evaluation strategies", en, *Journal of Biomedical Informatics*, vol. 113, p. 103 655, Jan. 2021, ISSN: 15320464. DOI: 10.1016/j.jbi.2020.103655. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1532046420302835 (visited on 02/04/2024).

[67] L. S. Carrier, "On What We Know We Don't Know. Explanation, Theory, Linguistics, and How Questions Shape Them", en, *Philosophical Books*, vol. 35, no. 1, pp. 38–39, Jan. 1994, ISSN: 0031-8051, 1468-0149. DOI: 10.1111/j.1468-0149.1994.tb02395.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1111/j.1468-0149.1994.tb02395.x (visited on 02/18/2024).

[68] M. Lent, W. Fisher, and M. Mancuso, "An explainable artificial intelligence system for small-unit tactical behavior.", Jan. 2004, pp. 900–907.

[69] D. Thompson, Ed., *The Oxford dictionary of current English* (Oxford paperbacks), eng, 2. ed., Oxford University Press paperback. Oxford: Oxford University Press, 1993, ISBN: 978-0-19-283127-9.

[70] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning", en, *arXiv:1806.00069 [cs, stat]*, Feb. 2019, arXiv: 1806.00069. [Online]. Available: http://arxiv.org/abs/1806.00069 (visited on 05/16/2020).

[71] R. S. Michalski, "A Theory and Methodology of Inductive Learning", en, in *Machine Learning*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 83–134, ISBN: 978-3-662-12407-9 978-3-662-12405-5. DOI: 10.1007/978-3-662-12405-5_4. [Online]. Available: http://link.springer.com/10.1007/978-3-662-12405-5_4 (visited on 09/20/2024).

[72] Y. Zhang and X. Chen, "Explainable Recommendation: A Survey and New Perspectives", en, *FNT in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020, ISSN: 1554-0669, 1554-0677. DOI: 10.1561/1500000066. [Online]. Available: http://www.nowpublishers.com/article/Details/INR-066 (visited on 09/14/2020).

[73] E. Tjoa and C. Guan, "A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI", en, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 4793–4813, Nov. 2021, ISSN: 2162-237X, 2162-2388. DOI: 10.1109/TNNLS.2020.3027314. [Online]. Available: https://ieeexplore.ieee.org/document/9233366/ (visited on 03/17/2024).

[74] M. Szczepański, M. Pawlicki, R. Kozik, and M. Choraś, "New explainability method for BERT-based model in fake news detection", en, *Scientific Reports*, vol. 11, no. 1, p. 23 705, Dec. 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-021-03100-6. [Online]. Available: https://www.nature.com/articles/s41598-021-03100-6 (visited on 10/06/2024).

[75] A. Gramegna and P. Giudici, "SHAP and LIME: An Evaluation of Discriminative Power in Credit Risk", *Frontiers in Artificial Intelligence*, vol. 4, p. 752 558, Sep. 2021, ISSN: 2624-8212. DOI: 10.3389/frai.2021.752558. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2021.752558/full (visited on 10/06/2024).

[76] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, *Striving for Simplicity: The All Convolutional Net*, Version Number: 3, 2014. DOI: 10.48550/ARXIV.1412.6806. [Online]. Available: https://arxiv.org/abs/1412.6806 (visited on 09/12/2024).

[77] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", 2016, Publisher: arXiv Version Number: 4. DOI: 10.48550/ARXIV.1610.02391. [Online]. Available: https://arxiv.org/abs/1610.02391 (visited on 09/12/2024).

[78] M. D. Zeiler and R. Fergus, *Visualizing and Understanding Convolutional Networks*, arXiv:1311.2901 [cs], Nov. 2013. [Online]. Available: http://arxiv.org/abs/1311.2901 (visited on 09/12/2024).

[79] J. H. Friedman, "Greedy function approximation: A gradient boosting machine.", *The Annals of Statistics*, vol. 29, no. 5, Oct. 2001, ISSN: 0090-5364. DOI: 10.1214/aos/1013203451. [Online]. Available: https://projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full (visited on 09/12/2024).

[80] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, *Intriguing properties of neural networks*, Version Number: 4, 2013. DOI: 10.48550/ARXIV.1312.6199. [Online]. Available: https://arxiv.org/abs/1312.6199 (visited on 09/12/2024).

[81] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable AI: A Review of Machine Learning Interpretability Methods", en, *Entropy*, vol. 23, no. 1, p. 18, Dec. 2020, ISSN: 1099-4300. DOI: 10.3390/e23010018. [Online]. Available: https://www.mdpi.com/1099-4300/23/1/18 (visited on 01/31/2024).

[82] R. R. Hoffman, S. T. Mueller, G. Klein, and J. Litman, "Metrics for explainable ai: Challenges and prospects", 2019. arXiv: 1812.04608 [cs.AI]. [Online]. Available: https://arxiv.org/abs/1812.04608.

[83] A. Rosenfeld, "Better metrics for evaluating explainable artificial intelligence", in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '21, Virtual Event, United Kingdom: International Foundation for Autonomous Agents and Multiagent Systems, 2021, pp. 45–50, ISBN: 9781450383073.

[84] F. Sovrano and F. Vitali, "An objective metric for Explainable AI: How and why to estimate the degree of explainability", en, *Knowledge-Based Systems*, vol. 278, p. 110 866, Oct. 2023, ISSN: 09507051. DOI: 10.1016/j.knosys.2023.110866. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950705123006160 (visited on 09/11/2024).

[85] F. Sovrano, S. Sapienza, M. Palmirani, and F. Vitali, "A Survey on Methods and Metrics for the Assessment of Explainability under the Proposed AI Act", 2021, Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2110.11168. [Online]. Available: https://arxiv.org/abs/2110.11168 (visited on 09/11/2024).

[86] G. Vilone, L. Rizzo, and L. Longo, "A comparative analysis of rule-based, model-agnostic methods for explainable artificial intelligence", Dec. 2020.

[87] M. R. Islam, M. U. Ahmed, S. Barua, and S. Begum, "A Systematic Review of Explainable Artificial Intelligence in Terms of Different Application Domains and Tasks", en, *Applied Sciences*, vol. 12, no. 3, p. 1353, Jan. 2022, ISSN: 2076-3417. DOI: 10.3390/app12031353. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1353 (visited on 09/20/2024).

[88] M. Pawlicki, A. Pawlicka, F. Uccello, *et al.*, "Evaluating the necessity of the multiple metrics for assessing explainable AI: A critical examination", en, *Neurocomputing*, vol. 602, p. 128 282, Oct. 2024, ISSN: 09252312. DOI: 10.1016/j.neucom.2024.128282. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231224010531 (visited on 09/20/2024).

[89] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, en. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, ISBN: 978-3-642-08233-7 978-3-662-03315-9. DOI: 10.1007/978-3-662-03315-9. [Online]. Available: http://link.springer.com/10.1007/978-3-662-03315-9 (visited on 09/20/2024).

[90] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future", en, *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, ISSN: 1380-7501, 1573-7721. DOI: 10.1007/s11042-020-10139-6. [Online]. Available: http://link.springer.com/10.1007/s11042-020-10139-6 (visited on 09/20/2024).

[91] D. Berrar, "Cross-Validation", en, in *Encyclopedia of Bioinformatics and Computational Biology*, Elsevier, 2019, pp. 542–545, ISBN: 978-0-12-811432-2. DOI: 10.1016/B978-0-12-809633-8.20349-X. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B978012809633820349X (visited on 09/20/2024).

[92] T. G. Dietterich, "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms", en, *Neural Computation*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976698300017197. [Online]. Available: https://direct.mit.edu/neco/article/10/7/1895-1923/6224 (visited on 06/03/2024).

[93] E. Alpaydm, "Combined 5 × 2 cv F Test for Comparing Supervised Classification Learning Algorithms", en, *Neural Computation*, vol. 11, no. 8, pp. 1885–1892, Nov. 1999, ISSN: 0899-7667, 1530-888X. DOI: 10.1162/089976699300016007. [Online]. Available: https://direct.mit.edu/neco/article/11/8/1885-1892/6310 (visited on 06/03/2024).

[94] T.-T. Wong and P.-Y. Yeh, "Reliable Accuracy Estimates from k -Fold Cross Validation", en, *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1586–1594, Aug. 2020, ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2019.2912815. [Online]. Available: https://ieeexplore.ieee.org/document/8698831/ (visited on 06/03/2024).

[95] J. Demšar, "Statistical comparisons of classifiers over multiple data sets", *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006, ISSN: 1532-4435.

[96] D. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, eng, Fifth edition. Boca Raton: Chapman & Hall/CRC, 2020, OCLC: 1162175922, ISBN: 978-1-00-008327-9.

[97] J. Alcalá-Fdez, L. Sánchez, S. García, *et al.*, "KEEL: A software tool to assess evolutionary algorithms for data mining problems", en, *Soft Computing*, vol. 13, no. 3, pp. 307–318, Feb. 2009, ISSN: 1432-7643, 1433-7479. DOI: 10.1007/s00500-008-0323-y. [Online]. Available: http://link.springer.com/10.1007/s00500-008-0323-y (visited on 06/03/2024).

[98] H. Deng, "Interpreting tree ensembles with inTrees", en, *Int J Data Sci Anal*, vol. 7, no. 4, pp. 277–287, Jun. 2019, ISSN: 2364-415X, 2364-4168. DOI: 10.1007/s41060-018-0144-8. [Online]. Available: http://link.springer.com/10.1007/s41060-018-0144-8 (visited on 01/24/2021).

[99] Y. Zhou and G. Hooker, *Interpreting Models via Single Tree Approximation*, en, arXiv:1610.09036 [stat], Oct. 2016. [Online]. Available: http://arxiv.org/abs/1610.09036 (visited on 03/17/2024).

[100] G. Vandewiele, O. Janssens, F. Ongenae, F. De Turck, and S. Van Hoecke, *GENESIM: Genetic extraction of a single, interpretable model*, en, arXiv:1611.05722 [cs, stat], Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.05722 (visited on 10/17/2022).

[101] G. Tolomei, F. Silvestri, A. Haines, and M. Lalmas, "Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking", en, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, arXiv:1706.06691 [stat], Aug. 2017, pp. 465–474. DOI: 10.1145/3097983.3098039. [Online]. Available: http://arxiv.org/abs/1706.06691 (visited on 03/17/2024).

[102] O. Bastani, C. Kim, and H. Bastani, *Interpreting Blackbox Models via Model Extraction*, en, arXiv:1705.08504 [cs], Jan. 2019. [Online]. Available: http://arxiv.org/abs/1705.08504 (visited on 03/17/2024).

[103] K. Kuratowski and A. Mostowski, *Set Theory: With an Introduction to Descriptive Set Theory* (Studies in logic and the foundations of mathematics). North-Holland, 1976. [Online]. Available: https://books.google.pl/books?id=cdcDMQAACAAJ.

[104] U. Adamy, M. Hoffmann, J. Solymosi, and M. Stojaković, "Coloring octrees", en, *Theoretical Computer Science*, vol. 363, no. 1, pp. 11–17, Oct. 2006, ISSN: 03043975. DOI: 10.1016/j.tcs.2006.06.021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0304397506003562 (visited on 05/03/2024).

[105] M. Brent, "Instance-based learning: Nearest neighbour with generalisation", Master thesis, University of Waikato, Hamilton, New Zealand, Mar. 1995. [Online]. Available: https://www.researchgate.net/publication/33051309_Instance-based_learning_Nearest_neighbour_with_generalisation.

[106] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection", *Statistics Surveys*, vol. 4, no. none, Jan. 2010, ISSN: 1935-7516. DOI: 10.1214/09-SS054. [Online]. Available: https://projecteuclid.org/journals/statistics-surveys/volume-4/issue-none/A-survey-of-cross-validation-procedures-for-model-selection/10.1214/09-SS054.full (visited on 09/13/2024).

[107] J. Fürnkranz, D. Gamberger, and N. Lavrač, *Foundations of Rule Learning* (Cognitive Technologies), en. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN: 978-3-540-75196-0 978-3-540-75197-7. DOI: 10.1007/978-3-540-75197-7. [Online]. Available: https://link.springer.com/10.1007/978-3-540-75197-7 (visited on 09/13/2024).

[108] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring Network Structure, Dynamics, and Function using NetworkX", Pasadena, California, Jun. 2008, pp. 11–15. DOI: 10.25080/TCWV9851. [Online]. Available: https://doi.curvenote.com/10.25080/TCWV9851 (visited on 09/13/2024).

[109] R. Lewis, "An introduction to classification and regression tree (cart) analysis", Jan. 2000.

[110] C. Singh, K. Nasseri, Y. S. Tan, T. Tang, and B. Yu, "Imodels: A python package for fitting interpretable models", *Journal of Open Source Software*, vol. 6, no. 61, p. 3192, 2021. DOI: 10.21105/joss.03192. [Online]. Available: https://doi.org/10.21105/joss.03192.

[111] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization", *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 281–305, Feb. 2012, ISSN: 1532-4435.

[112] B. Gulowaty and M. Wozniak, "Extracting Interpretable Decision Tree Ensemble from Random Forest", in *2021 International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China: IEEE, Jul. 2021, pp. 1–8, ISBN: 978-1-66543-900-8. DOI: 10.1109/IJCNN52387.2021.9533601. [Online]. Available: https://ieeexplore.ieee.org/document/9533601/ (visited on 10/03/2021).

[113] P. Langley and H. A. Simon, "Applications of machine learning and rule induction", en, *Communications of the ACM*, vol. 38, no. 11, pp. 54–64, Nov. 1995, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/219717.219768. [Online]. Available: https://dl.acm.org/doi/10.1145/219717.219768 (visited on 05/06/2024).

[114] C. Molnar, *Interpretable machine learning: a guide for making black box models explainable*, eng. Victoria, British Columbia: Leanpub, 2020, ISBN: 978-0-244-76852-2.

[115] J. Bacardit, A. E. I. Brownlee, S. Cagnoni, G. Iacca, J. McCall, and D. Walker, "The intersection of evolutionary computation and explainable AI", en, in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Boston Massachusetts: ACM, Jul. 2022, pp. 1757–1762, ISBN: 978-1-4503-9268-6. DOI: 10.1145/3520304.3533974. [Online]. Available: https://dl.acm.org/doi/10.1145/3520304.3533974 (visited on 09/12/2024).

[116] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, *Local Rule-Based Explanations of Black Box Decision Systems*, Version Number: 1, 2018. DOI: 10.48550/ARXIV.1805.10820. [Online]. Available: https://arxiv.org/abs/1805.10820 (visited on 09/12/2024).

[117] S. Sharma, J. Henderson, and J. Ghosh, "CERTIFAI: A Common Framework to Provide Explanations and Analyse the Fairness and Robustness of Black-box Models", en, in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, New York NY USA: ACM, Feb. 2020, pp. 166–172, ISBN: 978-1-4503-7110-0. DOI: 10.1145/3375627.3375812. [Online]. Available: https://dl.acm.org/doi/10.1145/3375627.3375812 (visited on 09/12/2024).

[118] Y. Tang, D. Nguyen, and D. Ha, "Neuroevolution of Self-Interpretable Agents", 2020, Publisher: arXiv Version Number: 2. DOI: 10.48550/ARXIV.2003.08165. [Online]. Available: https://arxiv.org/abs/2003.08165 (visited on 09/12/2024).

[119] J. Jordan, M. Schmidt, W. Senn, and M. A. Petrovici, "Evolving interpretable plasticity for spiking networks", en, *eLife*, vol. 10, e66273, Oct. 2021, ISSN: 2050-084X. DOI: 10.7554/eLife.66273. [Online]. Available: https://elifesciences.org/articles/66273 (visited on 09/12/2024).

[120] E. Canti-Paz and C. Kamath, "Inducing oblique decision trees with evolutionary algorithms", en, *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 54–68, Feb. 2003, ISSN: 1089-778X. DOI: 10.1109/TEVC.2002.806857. [Online]. Available: http://ieeexplore.ieee.org/document/1179908/ (visited on 09/12/2024).

[121] M. Kretowski, "An Evolutionary Algorithm for Oblique Decision Tree Induction", in *Artificial Intelligence and Soft Computing - ICAISC 2004*, D. Hutchison, T. Kanade, J. Kittler, *et al.*, Eds., vol. 3070, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 432–437, ISBN: 978-3-540-22123-4 978-3-540-24844-6. DOI: 10.1007/978-3-540-24844-6_63. [Online]. Available: http://link.springer.com/10.1007/978-3-540-24844-6_63 (visited on 09/12/2024).

[122] A. Shali, M. R. Kangavari, and B. Bina, "Using genetic programming for the induction of oblique decision trees", in *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, Cincinnati, OH, USA: IEEE, Dec. 2007, pp. 38–43, ISBN: 978-0-7695-3069-7. DOI: 10.1109/ICMLA.2007.66. [Online]. Available: http://ieeexplore.ieee.org/document/4457205/ (visited on 09/12/2024).

[123] J. Bacardit, E. K. Burke, and N. Krasnogor, "Improving the scalability of rule-based evolutionary learning", en, *Memetic Computing*, vol. 1, no. 1, pp. 55–67, Mar. 2009, ISSN: 1865-9284, 1865-9292. DOI: 10.1007/s12293-008-0005-4. [Online]. Available: http://link.springer.com/10.1007/s12293-008-0005-4 (visited on 09/12/2024).

[124] J. Juan Liu and J. Tin-Yau Kwok, "An extended genetic rule induction algorithm", in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, vol. 1, La Jolla, CA, USA: IEEE, 2000, pp. 458–463, ISBN: 978-0-7803-6375-5. DOI: 10.1109/CEC.2000.870332. [Online]. Available: http://ieeexplore.ieee.org/document/870332/ (visited on 09/12/2024).

[125] R. Mazouni and A. Rahmoun, "AGGE: A Novel Method to Automatically Generate Rule Induction Classifiers Using Grammatical Evolution", en, in *Intelligent Distributed Computing VIII*, D. Camacho, L. Braubach, S. Venticinque, and C. Badica, Eds., vol. 570, Series Title: Studies in Computational Intelligence, Cham: Springer International Publishing, 2015, pp. 279–288, ISBN: 978-3-319-10421-8 978-3-319-10422-5. DOI: 10.1007/978-3-319-10422-5_30. [Online]. Available: https://link.springer.com/10.1007/978-3-319-10422-5_30 (visited on 09/12/2024).

[126] R. C. Barros, M. P. Basgalupp, A. C. P. L. F. De Carvalho, and A. A. Freitas, "A Survey of Evolutionary Algorithms for Decision-Tree Induction", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 3, pp. 291–312, May 2012, ISSN: 1094-6977, 1558-2442. DOI: 10.1109/TSMCC.2011.2157494. [Online]. Available: http://ieeexplore.ieee.org/document/5928432/ (visited on 09/12/2024).

[127] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable Artificial Intelligence by Genetic Programming: A Survey", *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 3, pp. 621–641, Jun. 2023, ISSN: 1089-778X, 1089-778X, 1941-0026. DOI: 10.1109/TEVC.2022.3225509. [Online]. Available: https://ieeexplore.ieee.org/document/9965435/ (visited on 09/12/2024).

[128] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming", en, *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, Sep. 2006, ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco.2006.14.3.309. [Online]. Available: https://direct.mit.edu/evco/article/14/3/309-344/1242 (visited on 09/12/2024).

[129] S. Silva and E. Costa, "Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories", en, *Genetic Programming and Evolvable Machines*, vol. 10, no. 2, pp. 141–179, Jun. 2009, ISSN: 1389-2576, 1573-7632. DOI: 10.1007/s10710-008-9075-9. [Online]. Available: http://link.springer.com/10.1007/s10710-008-9075-9 (visited on 09/12/2024).

[130] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, "Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning", en, in *Proceedings of the Genetic and Evolutionary Computation Conference*, Berlin Germany: ACM, Jul. 2017, pp. 1041–1048, ISBN: 978-1-4503-4920-8. DOI: 10.1145/3071178.3071287. [Online]. Available: https://dl.acm.org/doi/10.1145/3071178.3071287 (visited on 09/12/2024).

[131] T. Soule and J. A. Foster, "Effects of Code Growth and Parsimony Pressure on Populations in Genetic Programming", en, *Evolutionary Computation*, vol. 6, no. 4, pp. 293–309, Dec. 1998, ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco.1998.6.4.293. [Online]. Available: https://direct.mit.edu/evco/article/6/4/293-309/832 (visited on 09/12/2024).

[132] E. Alfaro-Cid, J. J. Merelo, F. F. De Vega, A. I. Esparcia-Alcázar, and K. Sharman, "Bloat Control Operators and Diversity in Genetic Programming: A Comparative Study", en, *Evolutionary Computation*, vol. 18, no. 2, pp. 305–332, Jun. 2010, ISSN: 1063-6560, 1530-9304. DOI: 10.1162/evco.2010.18.2.18206. [Online]. Available: https://direct.mit.edu/evco/article/18/2/305-332/1341 (visited on 09/12/2024).

[133] S. Silva and S. Dignum, "Extending Operator Equalisation: Fitness Based Self Adaptive Length Distribution for Bloat Free GP", in *Genetic Programming*, L. Vanneschi, S. Gustafson, A. Moraglio, I. De Falco, and M. Ebner, Eds., vol. 5481, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 159–170, ISBN: 978-3-642-01180-1 978-3-642-01181-8. DOI: 10.1007/978-3-642-01181-8_14. [Online]. Available: http://link.springer.com/10.1007/978-3-642-01181-8_14 (visited on 09/12/2024).

[134] M. Naoki, B. McKay, N. Xuan, E. Daryl, and S. Takeuchi, "A New Method for Simplifying Algebraic Expressions in Genetic Programming Called Equivalent Decision Simplification", in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, S. Omatu, M. P. Rocha, J. Bravo, *et al.*, Eds., vol. 5518, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 171–178, ISBN: 978-3-642-02480-1 978-3-642-02481-8. DOI: 10.1007/978-3-642-02481-8_24. [Online]. Available: http://link.springer.com/10.1007/978-3-642-02481-8_24 (visited on 09/12/2024).

[135] M. Kommenda, G. Kronberger, M. Affenzeller, S. M. Winkler, and B. Burlacu, "Evolving Simple Symbolic Regression Models by Multi-Objective Genetic Programming", in *Genetic Programming Theory and Practice XIII*, R. Riolo, W. Worzel, M. Kotanchek, and A. Kordon, Eds., Series Title: Genetic and Evolutionary Computation, Cham: Springer International Publishing, 2016, pp. 1–19, ISBN: 978-3-319-34221-4 978-3-319-34223-8. DOI: 10.1007/978-3-319-34223-8_1. [Online]. Available: http://link.springer.com/10.1007/978-3-319-34223-8_1 (visited on 09/12/2024).

[136] L. Vanneschi, M. Castelli, and S. Silva, "Measuring bloat, overfitting and functional complexity in genetic programming", en, in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, Portland Oregon USA: ACM, Jul. 2010, pp. 877–884, ISBN: 978-1-4503-0072-8. DOI: 10.1145/1830483.1830643. [Online]. Available: https://dl.acm.org/doi/10.1145/1830483.1830643 (visited on 09/12/2024).

[137] G. S. I. Aldeia and F. O. De França, "Measuring feature importance of symbolic regression models using partial effects", en, in *Proceedings of the Genetic and Evolutionary Computation Conference*, Lille France: ACM, Jun. 2021, pp. 750–758, ISBN: 978-1-4503-8350-9. DOI: 10.1145/3449639.3459302. [Online]. Available: https://dl.acm.org/doi/10.1145/3449639.3459302 (visited on 09/12/2024).

[138] S. Ahmed, M. Zhang, L. Peng, and B. Xue, "Multiple feature construction for effective biomarker identification and classification using genetic programming", en, in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Vancouver BC Canada: ACM, Jul. 2014, pp. 249–256, ISBN: 978-1-4503-2662-9. DOI: 10.1145/2576768.2598292. [Online]. Available: https://dl.acm.org/doi/10.1145/2576768.2598292 (visited on 09/12/2024).

[139] L. S. Oliveira, M. Morita, R. Sabourin, and F. Bortolozzi, "Multi-objective Genetic Algorithms to Create Ensemble of Classifiers", in *Evolutionary Multi-Criterion Optimization*, D. Hutchison, T. Kanade, J. Kittler, *et al.*, Eds., vol. 3410, Series Title: Lecture Notes in Computer Science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 592–606, ISBN: 978-3-540-24983-2 978-3-540-31880-4. DOI: 10.1007/978-3-540-31880-4_41. [Online]. Available: http://link.springer.com/10.1007/978-3-540-31880-4_41 (visited on 09/12/2024).

[140] M. N. Adnan and M. Z. Islam, "Optimizing the number of trees in a decision forest to discover a subforest with high ensemble accuracy using a genetic algorithm", en, *Knowledge-Based Systems*, vol. 110, pp. 86–97, Oct. 2016, ISSN: 09507051. DOI: 10.1016/j.knosys.2016.07.016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950705116302301 (visited on 09/12/2024).

[141] G. Ngo, R. Beard, and R. Chandra, "Evolutionary bagging for ensemble learning", en, *Neurocomputing*, vol. 510, pp. 1–14, Oct. 2022, ISSN: 09252312. DOI: 10.1016/j.neucom.2022.08.055. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0925231222010414 (visited on 09/12/2024).

[142] J. Sylvester and N. Chawla, "Evolutionary Ensemble Creation and Thinning", in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, BC, Canada: IEEE, 2006, pp. 5148–5155, ISBN: 978-0-7803-9490-2. DOI: 10.1109/IJCNN.2006.247245. [Online]. Available: http://ieeexplore.ieee.org/document/1716816/ (visited on 09/12/2024).

[143] E. Alpaydın, *Introduction to machine learning* (Adaptive computation and machine learning), eng, Fourth edition. Cambridge, Massachusetts London: The MIT Press, 2020, ISBN: 978-0-262-04379-3.

[144] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification", in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India: IEEE, May 2019, pp. 1255–1260, ISBN: 978-1-5386-8113-8. DOI: 10.1109/ICCS45141.2019.9065747. [Online]. Available: https://ieeexplore.ieee.org/document/9065747/ (visited on 09/13/2024).

[145] K. Stapor, P. Ksieniewicz, S. García, and M. Woźniak, "How to design the fair experimental classifier evaluation", en, *Applied Soft Computing*, vol. 104, p. 107 219, Jun. 2021, ISSN: 15684946. DOI: 10.1016/j.asoc.2021.107219. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1568494621001423 (visited on 09/13/2024).

[146] H. M. Kakde, *Range searching using kd tree.* 2005.

[147] B. Gulowaty and M. Woźniak, "Search-based framework for transparent non-overlapping ensemble models", in *2022 International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–6. DOI: 10.1109/IJCNN55064.2022.9892360.

[148] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python", *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[149] J. M. Sotoca, R. A. Mollineda, and J. S. Sanchez, "A meta-learning framework for pattern classification by means of data complexity measures", *INTELIGENCIA ARTIFICIAL*, vol. 10, no. 29, p. 481, Dec. 2006, ISSN: 1988-3064, 1137-3601. DOI: 10.4114/ia.v10i29.875. [Online]. Available: http://journal.iberamia.org/index.php/ia/article/view/481 (visited on 09/13/2024).

[150] M. R. Smith, T. Martinez, and C. Giraud-Carrier, "An instance level analysis of data complexity", en, *Machine Learning*, vol. 95, no. 2, pp. 225–256, May 2014, ISSN: 0885-6125, 1573-0565. DOI: 10.1007/s10994-013-5422-z. [Online]. Available: http://link.springer.com/10.1007/s10994-013-5422-z (visited on 04/14/2024).

[151] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art", en, *Knowledge-Based Systems*, vol. 212, p. 106 622, Jan. 2021, ISSN: 09507051. DOI: 10.1016/j.knosys.2020.106622. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950705120307516 (visited on 05/13/2024).

[152] B. Gulowaty and P. Ksieniewicz, "Smote algorithm variations in balancing data streams", in *Intelligent Data Engineering and Automated Learning – IDEAL 2019*, H. Yin, D. Camacho, P. Tino, A. J. Tallón-Ballesteros, R. Menezes, and R. Allmendinger, Eds., Cham: Springer International Publishing, 2019, pp. 305–312, ISBN: 978-3-030-33617-2.

[153] B. Gulowaty and M. Woźniak, "Extracting interpretable decision tree ensemble from random forest", in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533601.