Assignment of bachelor's thesis

Title:	Comparing interpretable models with post-hoc explainable black box models
Student:	Mikuláš Kočí
Supervisor:	Mgr. Vojtěch Rybář
Study program:	Informatics
Branch / specialization:	Artificial Intelligence 2021
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2024/2025

Electronically approved by Ing. Magda Friedjungová, Ph.D. on 14 February 2024 in Prague.

Instructions

Some authors argue that only interpretable models should be used for high-stake decisions as they can achieve similar levels of accuracy as black-box models [1], and their post-hoc explanations could be misguiding [2]. This thesis aims to test these claims on selected datasets.

1) Conduct a comprehensive survey of interpretable models and their performance metrics compared to black-box models.
2) Evaluate at least three interpretable models and one high-performing black-box model with post-hoc explanation, i.e. gradient boosting on regression trees with SHAP values post-hoc explanation, across at least three different tabular datasets.
3) Compare the performance of these models in terms of relevant metrics. Analyze the insights provided by interpretability and explainability methods.
4) Compile a detailed report that synthesizes the findings from the literature review, the empirical evaluations, and the comparative analysis.

Reference
[1] Rudin, Cynthia. "Stop explaining black-box machine learning models for high stakes decisions and use interpretable models instead." Nature machine intelligence 1.5 (2019): 206-215.
[2] Rudin, Cynthia, et al. "Interpretable machine learning: Fundamental principles and 10 grand challenges." Statistic Surveys 16 (2022): 1-85.

Bachelor's thesis

COMPARING INTERPRETABLE MODELS WITH POST-HOC EXPLAINABLE BLACK BOX MODELS

Mikuláš Kočí

Faculty of Information Technology
Department of Applied Mathematics
Supervisor: Mgr. Vojtěch Rybář
May 16, 2024

Citation of this thesis: Kočí Mikuláš. *Comparing interpretable models with post-hoc explainable black box models.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Contents

List of Figures

List of Tables

I am extremely grateful to my supervisor, Mgr. Vojtěch Rybář, for all his support with this thesis.

Declaration

In Prague on May 16, 2024

Abstract

The main objective of this thesis is to provide further proof that interpretable models can achieve similar, if not better performance than black-box models. In our experiment, we found that there was not one dataset, where the black-box model performed significantly better than its interpretable counterparts, the highest difference we saw in terms of the F1 score was 0.02. This advantage is definitely not significant enough to outweigh the advantages brought about by an inherently interpretable model. This is especially true for high-stakes decisions, where we would be forced to use some explainability method, which could fail to reveal bias of the black-box model, potentially leading to poor performance in the real world.

Keywords machine learning, interpretability, explainability, algorithm comparison, decision tree, scoring system, generalized additive model

Abstrakt

Hlavním cílem této práce bylo poskytnout další důkazy, že interpretovatelných modely jsou schopné dosáhnout podobných, ne-li lepšího výkonu, než černé skříňky. V našem experimentu jsme zjistili, že pro ani jeden dataset nedosahovala černá skříňka výrazně lepších výsledků, než její interpretovatelné protějšky, kde nejvyšší zaznamený rozdíl z hlediska F1 skóre byl 0.02. Tato výhoda však rozhodně není natolik významná, aby preválila výhody, které přináší ze své podstaty interpretovatelný model. To platí zejména pro rozhodování s vysokou mírou riyika, kde bychom museli využít nějakou metodu vysvětlitelnosti, která by však nemusela odhalit zaujatost modelu, což by mohlo vést ke špatnému výkonu v reálném světě.

Klíčová slova strojové učení, interpretabilita, vysvětlitelnost, porovnání algoritmů, rozhodovací strom, skórovací systém, zobecněný aditivní model

List of abbreviations

AI	Artificial Intelligence
CORELS	Certifiable Optimal RulE ListS
EBM	Explainable Boosting Machine
FN	False Negatives
FP	False Positives
GAM	Generalised Additive Models
GOSDT	Generalized and Scalable Optimal Sparse Decision Trees
LIME	Local Interpretable Model-agnostic Explanations
ML	Machine Learning
PDP	Partial Dependence Plot
SHAP	SHapley Additive exPlanations
SLIM	Supersparse Linear Integer Models
TN	True Negatives
TP	True Positives

Introduction

In recent years, we have seen an extreme increase in the popularity of machine learning. It is now widely used in fields where it helps to make important decisions. The most prominent fields are medicine, law, and finance.

Let us give an example from medicine so that we can better understand the demand for explanations. Suppose that you are a doctor and you come to the conclusion that your patient is not ill. Once you enter the medical information of your patient into a computer programme designed to help your decisions, it suggests that he has a serious disease. This makes you check the data once again only to come to the conclusion that he is healthy. If the programme does not explain why it believes the patient is ill, how can you trust it to help your decisions? Without providing an explanation as to why it gave such an answer, this programme is not very helpful.

If you have read the title of this thesis, you might be asking yourself what the word model is referring to. Think of it as a computer programme that will give a prediction based on an input. Now that we know why explanations are necessary, let us explore how we acquire these explanations. Some of these models are much easier to comprehend than others. This group of models is called interpretable models. Their beauty lies in their simplicity. We can easily understand how and why it made such predictions. The other group of models is called black-box models. Black-box models are exactly the opposite. They are quite complex, so there is no easy way to understand their behaviour. So why do we even have black-box models? Because they are more complex, in some cases they can provide better predictions than interpretable models.

Everyone desires to have the model with the most accurate predictions, so naturally we would like to use black-box models to not miss out on any potential improvements to predictions. As was mentioned previously in some fields, explaining the decision of our model is necessary, so methods were developed to provide some sort of explanation. There are many approaches to this problem, but we have no guarantee that the model behaves as the explanation suggests.

Although explanations of black-boxes might be misleading, they are still used even in fields where they shouldn't be due to the impact of the decision. In many cases where black-box models are currently being used, interpretable models would achieve similar predictive power, while providing a much better understanding of the model's prediction.

This thesis will be divided into two main parts. The first part will provide the necessary theoretical background on interpretable machine learning and the most prominent groups of interpretable models. It will also cover the black-box models used later in the thesis and provide findings comparing performance between the two groups of models. In the second part, we will conduct an experiment comparing the performance of selected interpretable models to a black-box model. In addition to the performance comparison, we will also take into account the insights provided by the interpretability and explainability methods used. The results will then be compared to the findings from the theoretical part of the thesis.

Chapter 1

Theoretical part

1.1 Interpretability in machine learning

In the field of interpretable machine learning (ML), there is no consensus on the definition of interpretability, as it is believed to be impossible to define it using strictly mathematical terms[1]. But to provide some insight into what the term might mean in the context of machine learning, we can define it as “the ability to explain or to present in understandable terms to a human”[2]. It is best to define the interpretability of a ML model relative to the problem domain[3, 4].

1.2 Interpretable machine learning models

Interpretability of a ML model can be a key factor in preventing potentially disastrous outcomes[5]. It allows us to communicate the exact behaviour of the model with experts in the field, who can help us examine if it bases its predictions on the correct principles. We can also explore whether the model is fair in relation to gender or ethnicity[5].

Another major advantage of interpretable models is that we can make an informed decision on whether to trust these models or not[6, 7]. When using ML models to aid our decision-making process, we need to have an understanding of the behaviour of the model. Without this knowledge, we would either have to trust it completely or choose to ignore its predictions. If we were to trust this model, we would effectively automate our decision-making process, which could ultimately lead to many unnecessary errors[5].

Now that we have established why these models are useful, let us present the most important groups of interpretable models.

1.2.1 Rule lists

Perhaps the simplest machine learning model to understand are rule lists. Rule lists are models consisting of multiple “if-then” rules, giving us the reasoning behind each prediction[8]. We provide an example of such a list in figure 1.1. While this format can obscure which features might be the most important for the model, it allows the analysis of each rule as an individual unit of knowledge[3].

To ensure the interpretability of the rule lists, sparsity is essential. A rule list is much easier to comprehend when it is sparse, which means that it contains a few simpler rules, as opposed to numerous complex ones. Although we inherently try to avoid overly simplified explanations of complicated relationships[9], sparsity of models can be crucial in many real-world applications, where it is necessary to memorise the model[5].

if $(age = 18 - 20)$ **and** $(sex = male)$ **then predict** *yes*
else if $(age = 21 - 23)$ **and** $(priors = 2 - 3)$ **then predict** *yes*
else if $(priors > 3)$ **then predict** *yes*
else predict *no*

Figure 1.1 Rule list predicting two-year recidivism. Taken from [8].

Sparsity can also be of great help in analysing counterfactuals[5]. Counterfactuals explore how changes in values of individual features affect the outcome, which can help to us identify problems in the model. Another often overlooked area to focus on when analysing the produced rules would be the cases where the observation satisfies a rule but its real class is different from the prediction, which can lead to the discovery of new knowledge about the domain[3].

Rule lists are a logical model. Logical models are composed of logical statements that contain “if-then” rules along with the connectives “and” and “or” in their clauses. Logical models such as rule lists and decision trees, which we will cover next, excel with categorical data that may have intricate relationships between them[5].

Generating a concise list of rules that performs well is a difficult task. Most recent approaches to generate rule lists leverage Bayesian analysis, which uses probability statements to speed up the optimisation process, but these methods fail to provide the best performing solution.

The approach which was used to create the rule list in figure 1.1, is called Certifiably Optimal RulE ListS (CORELS), which delivers the optimal solution[8].

1.2.1.1 CORELS

Optimality of the rule list provided by this branch-and-bound algorithm can help us examine how close other, perhaps simpler, models are to the optimal solution. The experiments in the paper introducing this algorithm lead us to believe that very little accuracy is sacrificed by using this model over other commonly used models that are not interpretable, more commonly referred to as black-box models[8].

1.2.2 Decision trees

For many years, decision trees have been a fundamental component of interpretable machine learning. They are also a logical model and can be easily visualised in the form of a graph, an example of which can be seen in figure 1.2. Every internal node in this graph symbolises a test of an attribute value, whose result dictates the branch to be followed during prediction[10]. When making a prediction, we start at the root node and follow branches of the tree based on the results of the tests in each internal node until we reach a leaf node where we find the value we will predict.

This graphical representation can help us better understand which attributes are the most important when making a prediction, with the attributes closest to the root node being the most important[3]. It is important to note that the tree usually does not contain all attributes, which can help us narrow down the attributes that might be helpful. Another way to represent a decision tree is through a list of rules. There will be a rule for each leaf in this tree, where each rule will describe the path to one of the leaves. Representing this way can improve its comprehensibility, as we can analyse each rule by itself[10].

The construction of high-performing decision trees has been a point of interest for many researchers over the past few decades. Generating an optimal decision tree is an NP-complete problem[12]. Perhaps the most common methods first create a tree using a heuristic function that determines the best way to split the data at this node and then prune the constructed tree afterwards[13]. One problem with this approach is the fact that we generate a tree that is not optimal, which can leave much to be desired in terms of performance and interpretability[5].

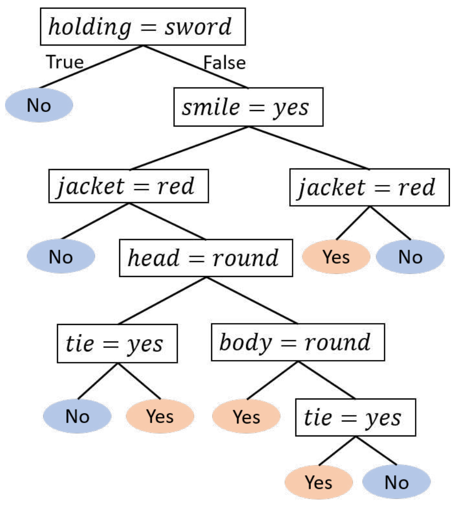

Figure 1.2 Decision tree, trained on the Monk 2 dataset[11]. Taken from [5].

Optimality of the model is also important with respect to the consequences of its application in the real world, where it could negatively impact someone's life[8].

If we make certain assumptions, we can reduce the difficulty of finding the optimal tree, such as the attributes being independent of each other[14]. However, these assumptions are not applicable to many real datasets. Another approach to finding an optimal decision tree can be to use branch-and-bound techniques along with standard mathematical programming solvers, which is computationally quite hard, largely due to continuous variables[13]. We can speed up the process using "bucketisation", which groups ranges of values together and replaces them with the number of the range into which they fall. With this approach, we lose optimality[13] since we do not know how to best group the values for optimal performance.

The method we will describe in more detail is Generalized and Scalable Optimal Sparse Decision Trees (GOSDT)[13], which was used to produce the tree in figure 1.2.

1.2.2.1 GOSDT

GOSDT[1] provides an improved adaptation of CORELS to decision trees. Lin et al.[13] provide an improvement in performance in datasets where the number of data points in one class is much higher than in the other. This can be done because we can choose the loss function that we would like to optimise. Loss functions evaluate how well the algorithm models the provided data. The function GOSDT aims to minimise is the sum of the loss function with a penalised number of leaves scaled by a regularisation parameter C.

$$\min_{f \in \text{ set of trees}} \frac{1}{n} \sum_i \text{Loss}(f, x_i) + C \cdot \text{Number of leaves } (f), \tag{1.1}$$

Another significant improvement is the implementation of the "similar support" bounds. These bounds are used when we have two similar features in the dataset. Using the bounds derived from the first feature for a tree split can be applied to derive bounds for the same tree if the second feature is used in place of the first. This allows for quicker assessment of trees that are very alike which massively decreases computational difficulty.

Further research from McTavish et al.[15] provides a dramatic reduction in training time, by using a black-box model, which helps to cut the size of the search space, while preserving the

[1]Pronounced as ghost

space where optimal or near-optimal solutions might be. When comparing the performance of GOSDT with the reference black-box, in some of their experiments, they found that GOSDT performs even better than the black-box, with respect to accuracy.

1.2.3 Linear models

Another cornerstone of interpretable ML models are linear models. These models assume a linear relationship between the independent variables and the target variable. Unlike the two models mentioned above, linear models do not natively handle interactions between features [5].

1.2.3.1 Linear regression

The first linear model that we would like to introduce is linear regression. Linear regression can be expressed as

$$y = \sum_{i=1}^{p} \beta_i x_i + \beta_0 + \varepsilon \tag{1.2}$$

where β_i are called regression coefficients, x_i are values of the independent variables, for the data point we are trying to classify. p denotes the number of independent variables in our dataset, β_0 is the intercept (sometimes called bias), and ε is a random variable, which represents the variability not accounted for by the independent variables[16]. The intercept represents the value, which we would predict if all the features were equal to zero. The predictions of the linear regression model are defined as

$$\hat{y} = \sum_{i=1}^{p} \beta_i x_i + \beta_0. \tag{1.3}$$

We can also model more complex relationships, if we use basis functions, which are nonlinear functions used to transform the input of a selected feature. We would represent this modification in the form

$$y = \sum_{i=1}^{p} \beta_i \phi_i(x_i) + \beta_0 + \epsilon \tag{1.4}$$

where ϕ_i represents the basis function for the feature x_i. This model is considered a linear model since it is still just a linear combination of features.

The regression coefficients play an important role in the interpretation of linear regression, as their magnitude represents the importance of the corresponding feature. To enable direct interpretability of the weights of the features as the importance of a feature, we first need to standardise the data[1]. This is necessary as the values of one feature could range in the thousands, whereas another could be in the millions. If the first feature had a weight of 1000 and the other of 1, they would effectively have the same impact on the outcome, although they have very different regression coefficients.

1.2.3.2 Logistic regression

Although the name of this model suggests that it is designed for a regression task, this is not the case. This is an adaptation of linear regression for classification.

This approach uses the sigmoid function to transform the predicted values of a linear regression model into the probability that the data point is from the positive class[1]. The sigmoid function is defined as:

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{1.5}$$

The probability of the data point being from the positive class is a great way to express how confident the model is with this particular prediction. This data is also great from the perspective of the user of such a model, who instead of just getting the predicted label can also receive this probability, which can help him decide whether he is going to trust this model's prediction or not.

The logistic regression model can then be defined as the sigmoid function of the predictions we would obtain with the linear regression model. The predicted probability that a data point is from the positive class can be expressed as:

$$P(y=1) = \frac{1}{1+\exp(-(\sum_{i=1}^{p} \beta_i x_i + \beta_0))} \tag{1.6}$$

with the β_i once again expressing the regression coefficients and x_i expressing individual values of variables for the data point whose class we are trying to predict.

Regression coefficients can once again be used to interpret how a change in a feature affects the output. In this case, we cannot base the interpretation directly on the coefficient values. When we increase the value of the feature x_i by one, we would effectively add its regression coefficient β_i to the value we pass to the exponential function within the prediction equation for this model. We can also put this in context of the ratio of the probability that the data point belongs to the positive class and the probability that the data point is from the negative class, where the ratio would change by a multiple of $\exp(\beta_i)$[1].

1.2.4 Scoring systems

Scoring systems are linear classification models, which are very simple to use for predictions even without the help of a computer, as they only require the user to know addition and subtraction for a few small integers[17]. This can be quite useful in some fields, such as medicine.

We provide an example scoring system in table 1.1.

Predict obstructive sleep apnea if score > 1		
Condition	**Points**	**Sum**
age $\geq$ 60	4 points	+ ...
hypertension	4 points	+ ...
bmi $\geq$ 30	2 points	+ ...
bmi $\geq$ 40	2 points	+ ...
female	-6 points	+ ...
Add points from 1 to 6		**Score** = ...

Table 1.1 Scoring system used for sleep apnea screening. Taken from [18].

Scoring systems tend to have points as integer values, mainly for improved interpretability[5]. This constraint makes the problem much more difficult to solve than if we allowed real-valued coefficients.

Many of the scoring systems currently used have been constructed strictly thanks to domain experts, who agreed upon a scoring system that should be used for this problem. The first attempts to generate a scoring system without domain knowledge involved learning a logistic regression model and then rounding its regression coefficients[19]. However, rounding of these values might result in a noticeable decrease in model performance.

There are now four main approaches that attempt to provide high-performing scoring systems. The first approach uses more complicated rounding methods. Another method that can be used is to approximate the original problem with a problem that is much easier to solve. The third method is constructed to interact with domain experts to tweak the produced model to their needs. The last method is to solve this problem exactly without any sacrifices. One model that falls into the last category is the Supersparse Linear Integer Model (SLIM)[19].

1.2.4.1 SLIM

SLIM is a model that is able to directly optimise both sparsity and accuracy. SLIM formulates the problem of building a high-performing scoring system using integer programming, which is later solved using an existing solver designed for this task.

Although this model is very easy to understand, the underlying problem we are solving is quite hard, so the time it takes to construct the final model is significantly longer than for many other available models. For this reason, SLIM is not suitable for large-scale datasets and can struggle with continuous variables[5].

1.2.5 General additive models

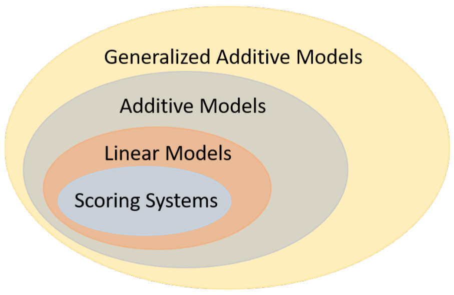

Figure 1.3 Hierarchical relationships between GAMs and some other previously mentioned interpretable models. Taken from [5].

General Additive Models (GAM) are a models that can be defined as:

$$g(E[y]) = \beta_0 + \sum_{i=1}^{p} f_i(x_i) \tag{1.7}$$

where $g()$ is a link function, which provide the relationship between the output on the right hand side and the mean of the target variable $E[y]$. If we aim to solve a regression task, we can use the identity function as the link function, while a logistic function should be used for classification. The functions f_i are component functions that do not have to be linear. β_0 and x_i carry the same meaning as in the linear regression model.

This model is probably the most complex yet interpretable model available. We can interpret this model by visualising each component function, an example of which we can see in figure 1.4.

This allows us to see not only the importance of this feature, but we can also examine how a certain range of values for an attribute might affect the decision more than another range. The graph is a little less interesting for categorical attributes, where we will only see step functions.

GAMs like other linear models do not capture interactions between features, which could hold back their performance on many datasets. Thankfully, this issue has been resolved with the GA^{2}M[20].

1.2.5.1 GA^{2}M

This model introduces automatic pairwise interaction detection between attributes. The suitable interactions can be found very quickly thanks to a new algorithm that the authors introduced alongside the model itself. These interaction terms can be studied using heatmaps to preserve interpretability.

There is also a publicly accessible implementation[21] of this model. In the paper introducing the library that provides an implementation of this model[21], the authors found that their model outperforms many of the popular black-box models for all but one of the datasets they tested.

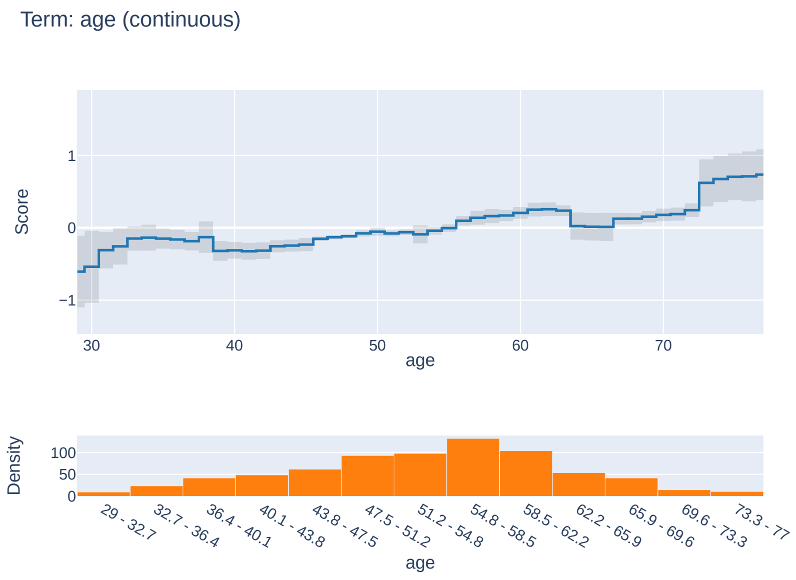

Figure 1.4 Example component function for a GAM.

1.3 Black-box models

We recognise two types of black-box models. The first more common type is a model that is too complex for us to comprehend. The other type is a proprietary model[22]. Black-box models are much more complex than interpretable ones, which we might associate with a better ability to learn from the data and provide better predictions. This assumption can cause us to not even consider using interpretable models, when they could achieve similar performance.

Although these models might be appealing, in many cases, their performance is comparable to their interpretable counterparts[22]. In addition to that, we have almost no insight into the inner workings of the model, so even though it gives the correct answer, it might be due to incorrect reasoning. This can cause the model to perform horribly in the real world, although it worked well when in development[23, 24, 25].

1.3.1 Gradient boosted decision trees

Gradient boosted decision trees are among the most popular black-box models[26].

It creates an ensemble of very simple trees, one after the other, where the next tree tries to focus on the data points which the previous tree misclassified. When constructing this ensemble, we assign each tree a weight, which typically decreases with each tree.

If we want to get predictions from this ensemble for a classification task, we have to compare the sum of weights for the trees predicting the positive class with the sum of weights for trees predicting the negative class.

1.3.1.1 CatBoost

CatBoost is a model that improves the standard gradient boosting algorithm and provides a better way to handle categorical features than other implementations of gradient boosting on decision trees. The improved gradient boosting technique is called ordered boosting, which prevents target leakage, which can cause very high performance on training data but poor performance on data unseen by the model. This should improve the generalisation ability of the trained model.

1.4 Explainable machine learning

Simply put, explainable ML is when we provide explanations of the black-box model's behaviour.

Post-hoc explanations were developed to try and combat the biggest disadvantage of black-box models, our inability to understand how the model predicts outcomes. However, there are all sorts of problems with these explanations. One of the main problems is that the explanations cannot perfectly represent the model, since there would be no need for the original model because it would be interpretable itself[22].

Some research[27, 28] suggests that there is an inherent trade-off between interpretability and accuracy of models, while other authors find that there is no such thing[13, 21]. We will put this to the test in the practical part of this thesis.

There are many approaches to explaining black-box models, but there are two main schools of thought when aiming to explain a black-box. The first approach, which we will refer to as global methods, aims to give a general understanding of the model's average behaviour. The second approach aims to explain each prediction individually.

1.4.1 Global methods

Starting with the global methods, there are many methods that use a graph to visualise the effects of one feature on the prediction made by the model. We will provide a closer look at one of them as well as another type of approach to global explanations.

1.4.1.1 Partial dependence plot

The Partial Dependence Plot (PDP) illustrates the isolated impact of an independent variable on the target variable[26]. It can help us examine the relationship between this feature and the target variable and see if it is, for example, linear or more complex. An example of this graph can be seen in figure 1.5.

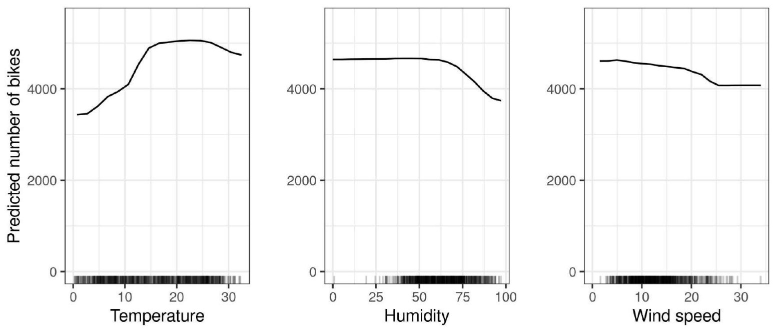

Figure 1.5 Example of a PDP. Taken from [1].

One problem with PDPs is that they assume that the feature that was used in the plot is not correlated with any other feature, which can produce data points that would otherwise never exist[1]. However, the accumulated local effects plot solves this issue by using the conditional distribution.

1.4.1.2 Permutation feature importance

Permutation feature importance is a completely different approach to global explanations. The concept of this method is that we measure the decrease in performance of our model if we train the model on data without knowing the real value of this feature for each data point[29].

This is accomplished by permuting the values of this attribute between the data points, which should break any relationship that might exist between the feature and the target variable. We then repeat this process for all attributes.

One possible downside to this problem is that the outcome of this method can change dramatically with different permutations[1]. We can combat this by repeating it over multiple permutations.

An advantage of this method is the ease of interpretation. If we compute it as a ratio of the error for a model that can not use this feature to a model that can, this metric can be easily compared across different models.

In the end, it is also important to ask ourselves if this method provides a sufficient explanation of the black-box model. This might be useful to provide a general understanding, but if we wanted to trust this model to make important decisions, we would definitely need more information about its behaviour.

1.4.2 Local methods

Local methods focus on explaining only one prediction at a time. We will provide an overview of two of the most commonly used methods.

1.4.2.1 Local interpretable model-agnostic explanations

Local Interpretable Model-agnostic Explanations (LIME) create an entirely new model, which aims to approximate the behaviour of a black-box model for a small part of the feature space around a specific prediction[30].

To create this explanation model, we need to create artificial data points that are close to the original prediction we want to explore. We then train a model that aims to minimise the difference of prediction from the model, while ensuring the model remains simple enough to interpret.

Although this idea sounds great on paper, even if we used this method, we might not reveal the potential bias of the underlying black-box model, which means that we cannot be sure these explanations are accurate[31]. This holds true even for the next method we are going to cover, which are SHapley Additive exPlanations (SHAP).

1.4.2.2 SHAP

This local explanation method is based on Shapley values from game theory, which describe how to distribute the reward to players in a fair manner[1]. In this case, the game is generating a prediction for a data point. The players are the feature values of the data point we are exploring. They collaborate to generate the difference between the mean predicted value and the prediction for this data point.

SHAP uses these Shapley values in a linear model. This representation connects the Shapley values and LIME, from which we get these explanations of the importance of each attribute for the specific prediction.

We can also estimate the feature importance for the model as a whole by summing up the absolute Shapley values over all predictions and dividing them. This ability to estimate feature importance and provide local explanations at the same time is one of the reasons why SHAP is so popular.

Chapter 2
Practical part

This part of the thesis will focus on the practical use of some of the models introduced in the previous chapter. We will compare the performance of three interpretable models along with one black-box model on four different datasets. The interpretable models that we used are a generalized additive model, a scoring system, and a decision tree, all of which were introduced in the theoretical part of the thesis. The black-box model is an ensemble of gradient-boosted trees.

Afterwards, we will focus on what insights these models bring into the problem domain. There will be no need for any further tools to explain the predictions of the first three models, since they are inherently interpretable. To better understand the behaviour of black-box model we will need some explainability method. We chose to use SHAP for this task, which is currently one of the most popular explainability methods.

We will start with a brief description of the datasets used in the experiment, after which we will cover the performance metrics used in our experiment.

2.1 Datasets

All datasets we used come from the well known University of California, Irvine ML repository[1]. This repository contains some of the most famous and widely used datasets. Due to its widespread use, it is almost always used when, for example, introducing a new ML model and comparing performance with some more established models.

All of the datasets that we use in this experiment lead to classification tasks. Since scoring systems are classification models and we wanted to use the same models across all the different datasets, all of them are suitable for classification.

2.1.1 Data preparation

The chosen datasets did not require too many changes before we could start training our models. All changes made were only a matter of changing the representation, for example conversion of the values “Male” and “Female” to an integer representation, where 1 corresponds to “Male” and 0 to “Female”.

Two of the models that we will be using in the experiment do not have a built-in way of dealing with missing data. Instead of removing all records with missing values, we will replace the missing values with the median value of the attribute[2]. These changes to the datasets will only be applied when we use the models unable to handle missing values themselves.

[1]https://archive.ics.uci.edu/
[2]The median is computed only on data from the training set

We also split the data into training, validation, and test sets. The training set will be used to train models when optimising hyperparameters. To evaluate which hyperparameters might achieve the best performance, we will compute the performance on the validation set and select the values which perform best. The performance on the validation set will also be used to compare between the different types of models we used in the experiment. The test set will be used to estimate what kind of performance we could expect from the final version of the model in a real-world application.

For splitting the data, as well as filling in missing values and computing metrics, we used the scikit-learn[32] library. We also used the pandas[33] library, to store the datasets in their data structure called *DataFrame*.

2.1.2 South German credit

The south German credit dataset[34], is a slightly improved version of the Statlog[35] dataset, which contains some personal information about clients, as well as their employment, savings and loans. In total, it provides 20 attributes, most of which are categorical, with only three continuous variables. The dataset does not have missing values, but it is imbalanced. There are 700 low-risk clients and 300 high-risk clients. Based on this information, the aim is to predict whether there is a high risk when lending money to this person or not.

2.1.3 Heart disease

The heart disease dataset[36] is another commonly used dataset, which contains medical information on almost 1000 patients. This dataset is nearly balanced, with about 55% of the patients being ill. There are 14 variables in total, again mostly made up of categorical variables with five exceptions. It has a number of missing values in multiple variables. Originally, there were four values to denote the severity of the disease, but later it was simplified to just detection of presence in the patient.

2.1.4 Thyroid disease

There are ten datasets on thyroid disease[37], of which we selected two, with the first focusing on detecting hypothyroidism and the second on euthyroid sick syndrome. Although both of these datasets have the same attributes, we will be training our models on each one separately, since we are trying to detect different conditions. These datasets contain 24 attributes, but we will disregard six of them, because they only suggest if some measurement was taken.

Of the 18 variables that we will consider, seven of them will be continuous. There are around 3000 data points in each of the two datasets, but both of these datasets are highly imbalanced, with the positive group representing around 5% of the observations in the hypothyroidism dataset and close to 10% in the euthyroid sick syndrome dataset. This dataset has missing values in all continuous variables as well as some missing values for patient's sex.

We will refer to the hypothyroidism dataset as the hypothyroid dataset and the euthyroid sick syndrome dataset as the euthyroid sick dataset.

2.2 Performance metrics

In this section we will cover the metrics commonly used to evaluate the performance of a model on a classification task. Before we do so, we have to introduce the confusion matrix.

2.2.1 Confusion matrix

The confusion matrix[38] for binary classification is a square matrix with 2 rows and 2 columns. To obtain the confusion matrix, we must first use the model of interest to predict the classes for each data point.

After generating the predictions, we have to compare them with the actual values. There are four possible results when comparing one prediction with its actual class:

- The true value of the data point is positive and the predicted value is positive as well. This case is called true positive (TP).
- The true value of the data point is negative, but the model incorrectly predicted that it was from the positive class. This case is called false positive (FP).
- The true value of the data point is positive, but we predicted that it was from the negative class. This case is called false negative (FN).
- The true value of the data point is negative, and the predicted value is also negative. This case is called true negative (TN).

Once we get the total numbers of each case, we can display them in a matrix form, as shown in figure 2.1.

	Actual positive	Actual negative
Predicted positive	True positives	False positives
Predicted negative	False negatives	True negatives

Figure 2.1 Confusion matrix for the binary classification task.

2.2.2 Accuracy

Accuracy[38] is the standard way of measuring the performance of a classification model. It is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.1}$$

It is the number of correctly labelled data points divided by the number of predictions. Possible accuracy values are in the range $[0, 1]$.

This measure is not suitable for imbalanced data. The reason being that if we only predicted the majority class and the 90% of the data points available were of that class, we would achieve an accuracy of 0.9, which sounds excellent on paper, but such model is not very useful. We will not consider this performance measure in our later comparison due to this problem.

2.2.3 Precision

Precision[38], otherwise known as positive predictive value, is defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.2}$$

This metric measures the ratio of data points correctly as from the positive class and the total number of predictions of the positive class.

2.2.4 Recall

Recall[38], also known as sensitivity or true positive rate, is defined as:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

Recall measures the ratio of the correctly predicted data points from the positive class and the total number of data points from the positive class.

2.2.5 F1 score

F1 score[38] is a metric that incorporates the recall and precision mentioned in the following way:

$$\text{F1 score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

Thanks to scaling the value by two, the possible values of the F1 score are from the range $[0, 1]$. This metric is the one we will use for comparison of the models' performance, while keeping an eye on the two metrics that combine to make the F1 score.

2.3 GAM

The first of the three interpretable models that we will use in our experiment is a generalized additive model with interaction between pairs of features, called a GA^{2}M[20].

The implementation of this model that we used in the experiment is from a Python package called InterpretML[21]. This package has a lot more to offer than just the implementation of the GA^{2}M model, which they call an Explainable boosting machine (EBM). The EBM learns each feature function individually, employing bagging and boosting techniques. Learning each feature function helps mitigate the impact of feature collinearity. When using the EBM, we do not have to worry about which interactions between features might be useful since the EBM detects them automatically.

InterpretML also provides interactive graphs for the EBM, which visualise each feature function, as well as a summary plot that gives an overview of each feature's importance. We can also visualise an individual prediction and how each feature contributed to the prediction.

The authors of InterpretML also compare performance of their model with two different black-box models, one of them being random forests and the other gradient boosted trees. In their comparison, the EBM provides the best performance for all tested datasets. Overall, this package is well documented and easy to use.

We selected three hyperparameters to tune for this model. The first of them controls how many pairwise interactions between attributes we allow the model to use. We also tried using different learning rate values, which control the degree to which the model adjusts when encountered with the error during the fitting of the model. The last hyperparameter with which we experimented was the maximum number of leaves, which affects how we cut the input space.

2.3.1 Performance evaluation

An overview of the performance of the EBM classifier can be found in table 2.1.

In the South German credit dataset, the best-performing model was one with the following hyperparameter values: number of interactions = 4, learning rate = 0.02 and maximum leaves = 3. This model achieved a precision of 0.76, but a recall of only 0.46, bringing the F1 score down to 0.58.

	Precision	Recall	F1 Score
Credit	0.76	0.46	0.58
Heart	0.83	0.84	0.84
Hypothyroid	1.00	0.83	0.91
Euthyroid	0.93	0.86	0.89

Table 2.1 Performance of the EBM on each dataset in terms of different metrics.

For the heart disease dataset, we achieved the best performance with four interactions, learning rate = 0.005 and maximum leaves = 4. This EBM achieved very similar values in both the precision and recall, with 0.83 and 0.84 respectively. The F1 score of this model was 0.84.

For both the hypothyroid and sick euthyroid datasets, the best performance was obtained by models that did not include any interactions. In the hypothyroid dataset, the other two hyperparameter values that resulted in the best performance were: learning rate = 0.0005 and maximum leaves = 4. The model with these hyperparameter values had a precision of 1, which is the best possible value, a recall of 0.83 and an F1 score of 0.91.

In the sick euthyroid dataset we found that learning rate = 0.15 and maximum leaves = 3 provided the best performance. This model achieved high values for each metric. The precision was 0.93, the recall was 0.86, and the F1 score was 0.89.

2.3.2 Domain insights

EBM is an interpretable model, which means that we do not need any explainability method to provide explanations for its predictions. To understand predictions made by an EBM, we need to know the feature functions, pairwise interactions, and the link function. This is quite complicated to understand without the graphing options provided by the InterpretML package.

Unfortunately, the InterpretML package does not provide a way to save the summary graph, so we had to recreate the results using the seaborn[39] package. You can see an example of how this summary graph could look like in figure 2.2, which was created with an EBM classifier trained on the sick euthyroid dataset.

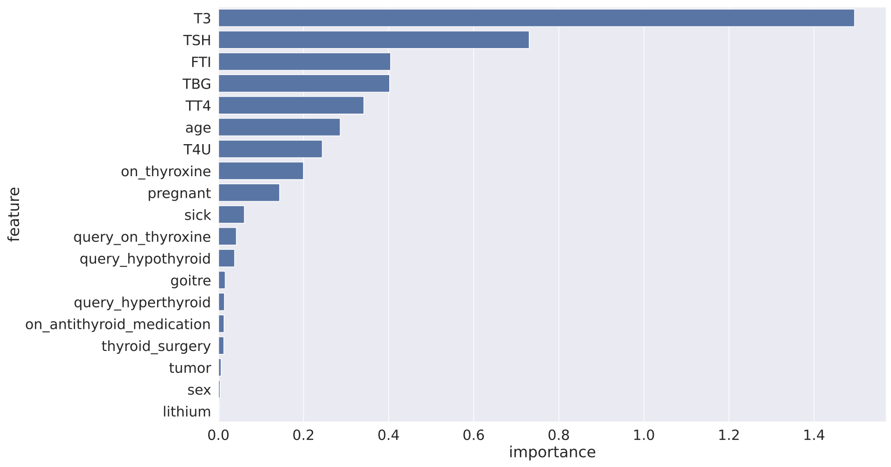

Figure 2.2 Plot summarising the global importance for each term.

This graph can provide an overview of the features that are of the most importance to the model. With expert knowledge of the domain, we can determine whether the model comes to its conclusions based on the correct reasoning and uncover any bias in the model.

In figure 2.3, we provide a visualisation of one feature function along with a histogram of the attribute values found in the dataset. Like in the previous figure, the model we used to create this visualisation was an EBM classifier trained on the sick euthyroid dataset.

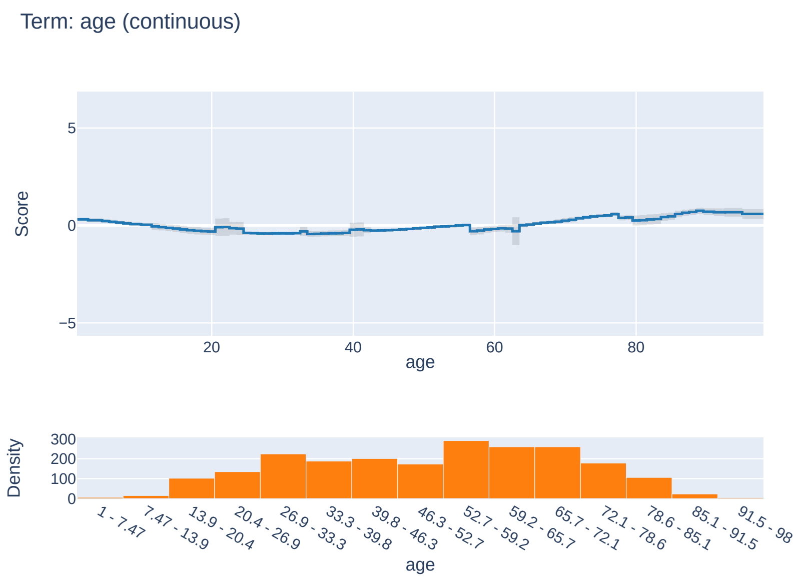

Figure 2.3 Feature function plot along with a histogram of this attribute.

This graph can provide further details on each feature function. We can analyse each function and perhaps find some relationship between one of the attributes and the target variable, which we might not have been aware of. On the other hand, if our knowledge of the domain leads us to believe that as this feature's value increases, so should the probability of the target variable being true, but it is not the case in our model, we can use a monotonicity constraint to ensure this relationship.

We can also examine the misclassified data points when trying to improve our model. In figure 2.4, we provide an example graph that can help us understand how the model came to its prediction. This graph was created using the EBM classifier trained on the heart disease dataset.

2.4 GOSDT

GOSDT is the second model that we will use in our experiment. More concretely, we will use the slightly improved version established in [15]. The authors of this paper also provide an implementation of their improvements. Unfortunately, this implementation made available by the authors does not work with the latest versions of Python, but there is a modified version compatible with them[3], which is the implementation we are going to use for our experiment.

The authors of the improved GOSDT, provided a package itself is relatively easy to understand and use. The only complaint we have is about the lack of a method to visualise the trees. The only method the authors of this package provide is representation as a list of rules. Unfortunately, we could not find another library that would be able to visualise these decision trees in a nice way, so we will not have a nice graphical representation for them.

Compared to the original implementation of the GOSDT[13], this package provides two ways to speed up training. One of them involves guessing good splits of continuous data into bins using

[3]https://github.com/ubc-systopia/gosdt-guesses/tree/sklearn-fix

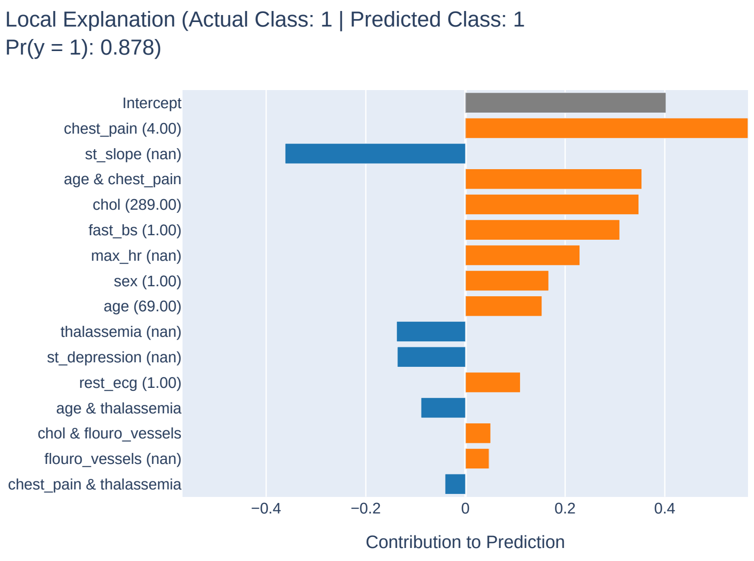

Figure 2.4 Breakdown of how each feature value contributed to the predicted value.

a reference black-box and the other using another black-box to limit the search space.

We first tried to use the package without any of these two methods, but found the training time for these models to be way too long. Then we used the first method, which helped dramatically decrease training time. Decision trees obtained this way are probably not optimal, but their performance should be comparable to the reference black-box[15].

We also wanted to use the second method, but we found that there was a significant decrease in the performance of our models when using it. In the end, we decided not to use it, but we are unsure of what could be the cause of this performance drop because we used this method exactly as the authors provided in their examples.

For this model, we had three hyperparameters to tune once again. The first being if the model should treat each data point with the same importance or not. We also tuned the maximum depth allowed for the final tree, as well as a parameter regularising the amount of leaves in the tree.

This model does not natively handle missing values, so we used modified versions of all datasets that contain missing values.

2.4.1 Performance evaluation

An overview of the performance of the GOSDT classifier can be found in table 2.2.

	Precision	Recall	F1 Score
Credit	0.45	0.80	0.58
Heart	0.76	0.84	0.80
Hypothyroid	1.00	0.93	0.97
Euthyroid	0.82	0.86	0.84

Table 2.2 Performance of the GOSDT on each dataset in terms of different metrics.

Starting with the South German credit dataset, the best-performing hyperparameter values were balance = true, regularisation = 0.005 and maximum depth = 5. The precision of the model was 0.45, which is quite low, but the recall was pretty good at 0.80, which meant that the F1 score = 0.58.

In the heart disease dataset, we saw much better values for all performance metrics. The best hyperparameter values were the same as for the best model on the South German credit dataset, except for balance being false. The best performing model had a precision of 0.76, a recall of 0.84, and an F1 score of 0.80.

For the hypothyroid dataset, the best hyperparameter values were surprisingly the same as for the heart disease dataset. But all of the performance metrics were significantly better. This model achieved a precision of 1, which is the best possible value, while the recall was 0.93, which is still very impressive. The F1 score was 0.97, which is outstanding.

For the sick euthyroid dataset, the hyperparameter values with the best performance remained the same once again. All metrics had lower values than on the previous dataset. The model had a precision of 0.82, while the recall was 0.86, which meant that the F1 score was 0.84.

2.4.2 Domain insights

GOSDT is an interpretable model whose true power lies in its simplicity. Unlike the EBM classifier, we do not need any extra visualisations, except its textual or graphical representation, to be able to predict its behaviour. This model is an excellent choice for applications where it is required that we remember the model, since we can easily limit the maximum allowed depth of this tree.

In figure 2.5, we can see a rule list representation of the model that achieved an F1 score of 0.97 on the hypothyroid dataset. In this case, it could even be simplified into only two rules, using one rule for the positive class and an "else" rule, which will predict the negative class.

if $(FTI <= 64.5)$ **and** $(TSH <= 6.45)$ **then predict** *negative*
else if $(FTI <= 64.5)$ **and** $(TSH > 6.45)$ **then predict** *hypothyroid*
else if $(FTI > 64.5)$ **then predict** *negative*

Figure 2.5 Rule list representation of the best performing GOSDT model on the hypothyroid dataset.

With this rule list representation, we can analyse each rule separately and easily tweak them if previous domain knowledge suggests something else than what the rules do.

The graphical representation of decision trees has one advantage. It would help us to understand which features are the most important to the model. If this model had high performance, we could then assume that this importance ranking holds true for the problem domain itself.

2.5 SLIM

The last interpretable model we will use in our experiment is the SLIM scoring system[4]. The original implementation of the SLIM was developed many years ago, and we had to make some small adjustments to make it work with the latest versions of Python. This modification is available as part of the implementation associated with this thesis.

This package is probably the least user-friendly of all the packages used in this experiment. There is a lot of preparation needed before we can even think about fitting the classifier, which would probably be dealt with automatically if this package were developed more recently. We also found that the format of the output only made sense when we had a dataset with only binary values. Since we had such success with the function used to find thresholds in the GOSDT package, we chose to use it here as well.

[4]https://github.com/ustunb/slim-python

There are also not many hyperparameters we can tune, with just one regularisation penalty, which controls the maximum amount of performance sacrificed to remove a feature from the solution. An important variable that is worth specifying is the weight of each class.

SLIM does not have a built-in way to handle missing values, so we used the modified versions of all datasets that contain missing values.

2.5.1 Performance evaluation

An overview of the different performance metrics of the SLIM scoring system can be found in table 2.3.

	Precision	Recall	F1 Score
Credit	0.45	0.80	0.58
Heart	0.81	0.82	0.81
Hypothyroid	0.67	0.93	0.78
Euthyroid	0.80	0.88	0.84

Table 2.3 Performance of the SLIM on each dataset in terms of different metrics.

The hyperparameter tuning for this model found that every model performed the best with the regularisation parameter set to the lowest value we allowed. The value of 0.001 still produced high-performing models, while not including too many features in the resulting scoring system.

For the South German credit dataset, SLIM achieved a poor precision of 0.45, but better recall of 0.80 and an F1 score of 0.58.

Performance values on the heart disease dataset were much more balanced with solid values in each metric. The precision was 0.81, the recall was 0.82, and the F1 score was 0.81.

On the hypothyroid dataset we were back to very different values of precision and recall. Although the best model achieved an impressive recall of 0.93, the precision was only 0.67 bringing the total F1 score to 0.78.

The precision on the sick euthyroid dataset was significantly better than on the previous dataset. The value the best model achieved was 0.80, with an impressive recall value of 0.88. The F1 score for this model was 0.84.

2.5.2 Domain insights

In table 2.4, we provide an example SLIM scoring system, which was trained on the South German credit dataset.

Predict high risk if score $\geq$ -1		
Condition	**Points**	**Sum**
status $\leq$ 2.5	4 points	...
credit_history $\leq$ 1.5	4 points	+ ...
status $\leq$ 3.5	2 points	+ ...
purpose $\leq$ 0.5	2 points	+ ...
duration $\leq$ 8.5	-4 points	+ ...
amount $\leq$ 11101.5	-4 points	+ ...
property $\leq$ 1.5	-4 points	+ ...
Add points from 1 to 7		**Score** = ...

Table 2.4 SLIM scoring system for the credit dataset

With a scoring system, we can extract even more information from the model than from a decision tree. Not only can we see which features are most important for the model when it comes to making a decision, we can also see how different values of this attribute might affect the decision less or more, as can be seen in the example scoring system with the attribute status, where if the value is between 2.5 and 3.5 it has a much lower effect than if the value were from the range 1.5 to 2.5.

We can also find that some features might be more important when predicting the negative class rather than the positive class, since the system allows us to reward negative points for certain values of an attribute.

Overall, this scoring system can make a great starting point when trying to get an understanding of the domain.

2.6 CatBoost

The black-box model we will use to compare the performance of the chosen interpretable models is a model using gradient-boosted trees.

The implementation we will use is called CatBoost[40]. This model implements a new approach to process categorical features as well as an improved version of the standard way to use gradient boosting for decision trees, called ordinal boosting. Many of the features included in our datasets are categorical, so this implementation is an excellent option for our experiment.

The CatBoost package is overall very pleasant to use. The documentation of this package is superb, but we did not really need it, since it was very intuitive to use. Probably the biggest advantage of this model is the fact that we do not need to play around with different hyperparameter values, in order to achieve high performance. Although that is the case, we still tried many values for the following hyperparameters: learning rate, maximum depth of the trees, and the total number of trees in the ensemble.

To provide explanations of this black-box model, we chose to use the SHAP package[5].

2.6.1 Performance evaluation

The performance of the CatBoost classifier on each dataset can be found in table 2.5.

	Precision	Recall	F1 Score
Credit	0.82	0.41	0.55
Heart	0.82	0.83	0.83
Hypothyroid	0.90	0.87	0.88
Euthyroid	0.91	0.85	0.88

Table 2.5 Performance of CatBoost on each dataset in terms of different metrics.

For the South German credit dataset, the CatBoost model achieved a precision of 0.82, but a pretty poor recall of 0.41, bringing the F1 score down to 0.55. This was done using a model with the default hyperparameter values.

The best performing hyperparameter values on the heart disease dataset, were the default values. With these values, the model achieved a very similar score in terms of precision and recall, with 0.82 and 0.83, with the F1 score of 0.83.

This classifier achieved excellent performance on both thyroid datasets, with an F1 score of 0.88 for each dataset.

On the hypothyroid dataset, the precision achieved by the model was 0.90, with the recall being 0.87. This result was achieved by the model with default parameters.

[5]https://github.com/shap/shap

For the sick euthyroid dataset, we found a slightly higher precision of 0.91, but a slightly worse recall of 0.85. This is the only dataset where we managed to find hyperparameter values that performed better than the default. This model used 60 decision trees with a maximum depth of 11 and a learning rate of 0.0005.

2.6.2 Domain insights

Due to CatBoost being a black-box model, in order to gain any insight on what might be important when this model makes its prediction, we need to use some explainability method.

One of the most popular methods to use is SHAP, which can provide an approximation of which features might have been important for each prediction, as well as providing an overview of the importance of each feature.

We can find an example of how an explanation of one prediction could look like in figure 2.6.

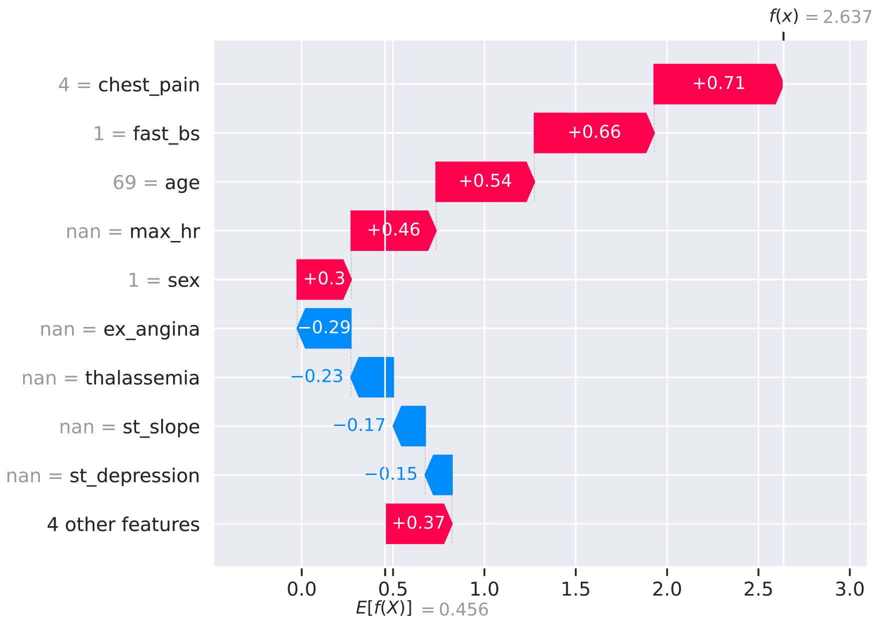

Figure 2.6 An explanation of one of the predictions made by the CatBoost classifier using SHAP.

The problem with this explanation method is that we can not tell if it is an accurate representation of each feature's contribution to the prediction. If so, it would be great to use these explanations when examining misclassified data points. Even if these explanations were truly representative of the underlying black-box model, if we were to find that the model relies too heavily on some feature or has some bias towards a particular group, which we believe it should not, we could not easily prevent the model from doing so.

SHAP also provides an estimate of the importance of each feature in all the prediction, an example of which can be found in figure 2.7. This graph represents the mean absolute value of the estimated contribution over all predictions.

The problem with this graph is that the inaccuracies in each explanation of a prediction might add up together and reveal a much different graph than one that would present an accurate importance of each feature. This behaviour might be acceptable for some applications, but not for the domains in our experiment.

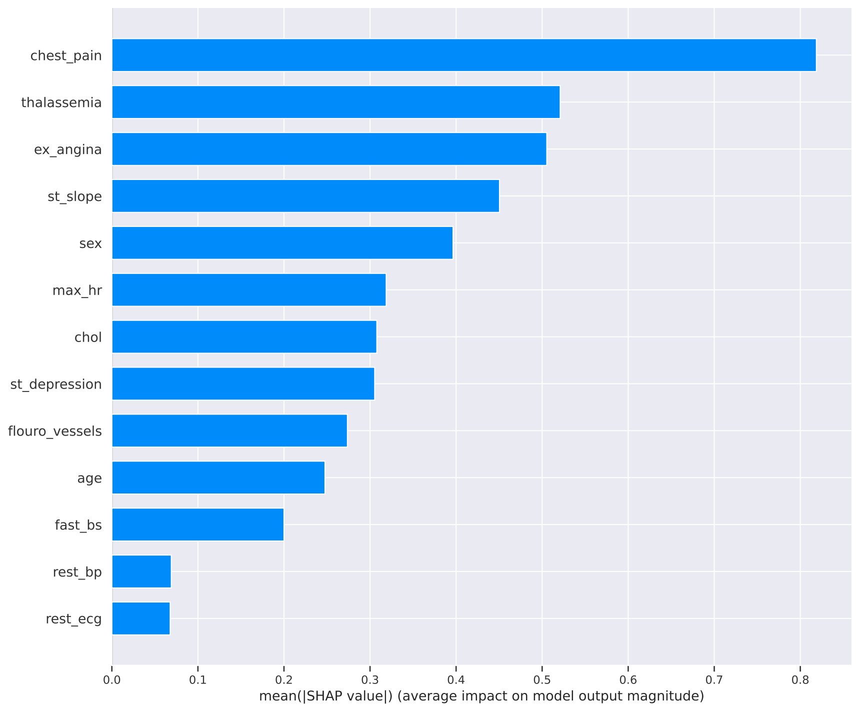

Figure 2.7 An approximation of importance of each feature for a CatBoost model trained on the heart disease dataset.

2.7 Comparative analysis

The table 2.6, represents a comparison of F1 scores for each model on each dataset, with the highest scores achieved for each dataset in bold.

	EBM	GOSDT	SLIM	CatBoost
Credit	**0.58**	**0.58**	**0.58**	0.55
Heart	**0.84**	0.80	0.81	0.83
Hypothyroid	0.91	**0.97**	0.78	0.88
Euthyroid	**0.89**	0.84	0.84	0.88

Table 2.6 F1 score comparison of all used models on validation sets for each dataset.

As we can see in table 2.6, the interpretable models managed to slightly outperform CatBoost on all datasets. The model that performed best overall was definitely the EBM classifier, which provided the highest F1 score on three out of four datasets, only being outperformed by the GOSDT classifier on the hypothyroid dataset. The GOSDT classifier performed the second best overall. It performed exceptionally well on the hypothyroid dataset compared to all other models. CatBoost performed very slightly worse than EBM on all datasets. The worst performing model was the SLIM scoring system, but still on some datasets it was not far behind the best achieved F1 scores.

With this table in mind, we selected the best performing interpretable models for each dataset and compared their performance against a black-box on unseen data to estimate how well they could perform in a real-world scenario.

2.7.1 South German credit dataset

For this dataset we could have used any of the interpretable models, since all of them had the same F1 score. We chose to use SLIM in this case because this was the only dataset in which it had the highest F1 score.

The SLIM scoring system achieved precision of 0.54, a recall of 0.61 and an F1 score of 0.57. It only suffered a 0.01 drop in terms of F1 compared to the previous comparison, while the CatBoost classifier experienced a 0.1 drop in F1 score, with a good precision score of 0.62, but a poor recall of 0.35.

In figure 2.8, we can see a visualisation of the confusion matrix for each model.

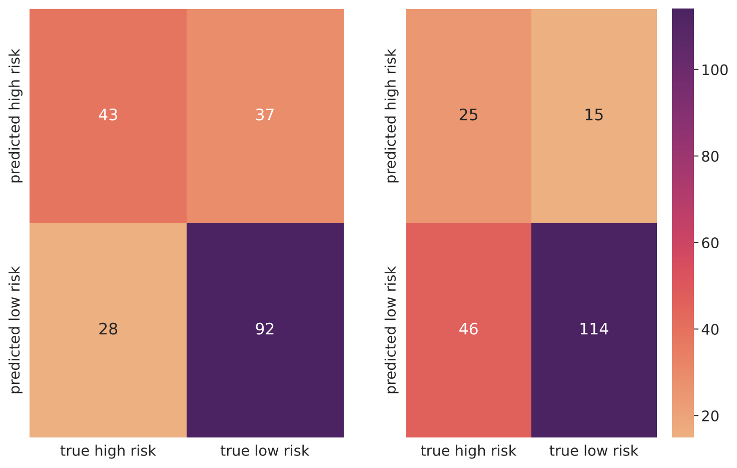

Figure 2.8 Visualisation of the confusion matrices for two models trained on the South German credit dataset. The left matrix belongs to the SLIM scoring system and the matrix on the right to CatBoost.

From the left matrix, which belongs to the SLIM scoring system, we can deduce that it is more likely to predict that a client is high risk even though in reality he is not. This is a classifier that provides much safer predictions than CatBoost, which produces many false negatives. In the context of this domain, the SLIM scoring system is a much better choice.

2.7.2 Heart disease dataset

The two best performing classifiers for this dataset were the EBM and the CatBoost.

The F1 score for the EBM only dropped by 0.01 to 0.83. The precision of the model was 0.80 and the recall was 0.88. CatBoost achieved a slightly better F1 score of 0.85. The precision had decreased to 0.81 while the recall improved to 0.90

In figure 2.9, we can see a visualisation of the confusion matrix for each model.

Both confusion matrices are very similar, with the EBM producing two extra false positives and two false negatives, compared to the CatBoost. However, we believe that this minor improvement in performance is far outweighed by the interpretability of the EBM classifier.

2.7.3 Hypothyroid dataset

The models we will use in this comparison are GOSDT and CatBoost.

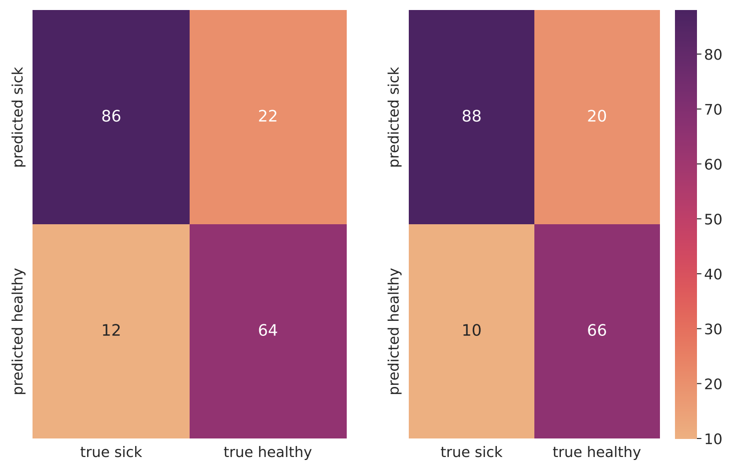

Figure 2.9 Visualisation of the confusion matrices for two models trained on the heart disease dataset. The left matrix belongs to the EBM classifier and the matrix on the right to CatBoost.

The precision was the same for both classifiers at 0.90. However, they differed slightly in terms of recall, with GOSDT having a recall of 0.87 compared to the recall of 0.90 for CatBoost. This meant that CatBoost had an F1 score of 0.90 while GOSDT achieved an F1 score of 0.88.

The confusion matrices of these models are visualised in figure 2.10.

The only extra incorrect prediction that caused the difference in performance was one false negative. Similarly to the previous dataset, we saw minimal improvement when choosing not to use an interpretable model.

2.7.4 Euthyroid dataset

We used the EBM as the interpretable model for this dataset.

The recall of both classifiers was the same at 0.88. The EBM had a slightly lower precision of 0.84 compared to the CatBoost precision of 0.86. The resulting F1 score was 0.86 for the EBM and 0.87 for the CatBoost.

You can see their respective confusion matrices in figure 2.11

There were only two additional misclassifications by the EBM, which were both false positives, but in the context of the domain, this might be preferred to extra false negatives. Once again, the difference between performance of the black-box model is not significant.

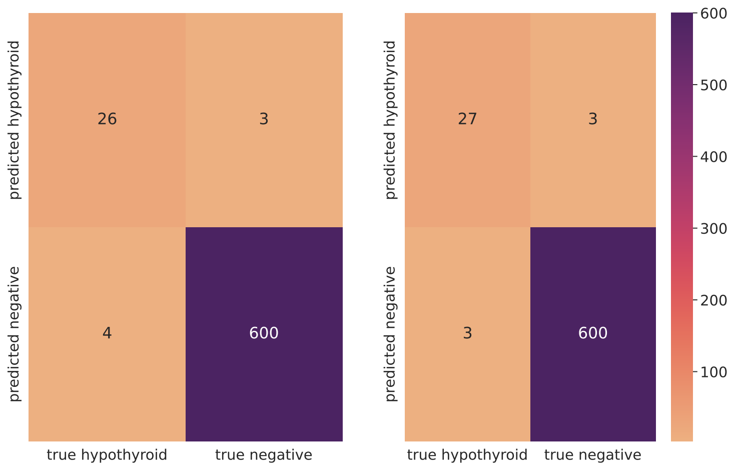

Figure 2.10 Visualisation of the confusion matrices for two models trained on the hypothyroid dataset. The left matrix belongs to the GOSDT classifier and the matrix on the right to CatBoost.

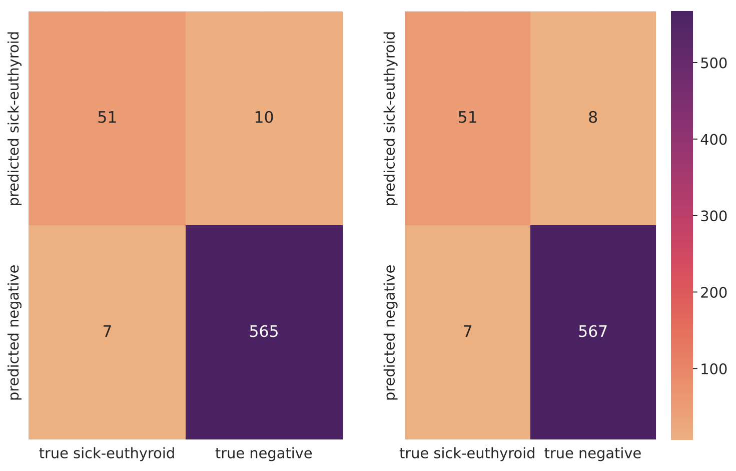

Figure 2.11 Visualisation of the confusion matrices for two models trained on the sick euthyroid dataset. The left matrix belongs to the EBM classifier and the matrix on the right to CatBoost.

Chapter 3

Conclusion

The main goal of this thesis was to compare the performance of interpretable machine learning models with a black-box model on a select few tabular datasets. This thesis also aimed to analyse the insights into the problem domain that come with using interpretable models compared to the insights brought by explainability methods.

The theoretical part introduced many interpretable models, along with what they bring to the table in terms of insight into the domain and their advantages over other types. We payed closer attention to interpretable models that we chose to use in the practical part of this thesis or upon which the models we used were built on, while mentioning their performance compared to black-box models if their authors provided such comparison.

We then introduced the black-box model that we chose for our practical part as well as the two main approaches to explain behaviour of black-box models. Afterwards, we provided some examples for both of them, once again with more focus on the specific approach we used in the next part of the thesis.

In the practical part of the thesis, we conducted the promised experiment with three interpretable models and one black-box model. The first of the interpretable models we used was a generalised additive model. We chose to use the implementation named EBM, whose authors showed that it slightly outperforms traditional black-box models on multiple datasets. To prevent our experiment from being too similar, we used a different black-box model called CatBoost, which is more suitable for the datasets we chose.

The second model we used was an optimal sparse decision tree, called GOSDT. We chose this model mainly because of its ease of interpretation and wondered if perhaps this simpler model might still provide high performance on some datasets.

The last interpretable model we used was a scoring system named SLIM. We included a scoring system because three out of the four datasets were healthcare related, and scoring systems seem to be a commonly used model in this field.

Although we tried to use a better black-box model, there was not a single dataset where we found its performance to be the best out of these four models. The EBM showed very slightly better performance on all datasets. There was even one data set where all interpretable models performed better than the CatBoost.

We then tried to simulate real-world performance comparison for the best performing interpretable model and the CatBoost. In these comparisons, except for one dataset, we found the black-box to perform very slightly better than its interpretable counterparts. We feel as though this insignificant difference in performance is more than compensated for with the interpretability of the other models, which enables us to use these models for high-stakes decisions and trust its predictions.

Bibliography

1. MOLNAR, Christoph. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable* [online]. 2nd ed. 2022. [visited on 2024-04-10]. Available from: `https://christophm.github.io/interpretable-ml-book`.
2. DOSHI-VELEZ, Finale; KIM, Been. *Towards A Rigorous Science of Interpretable Machine Learning* [online]. 2017. [visited on 2024-04-10]. Available from arXiv: `1702.08608 [stat.ML]`.
3. FREITAS, Alex A. Comprehensible classification models: a position paper. *SIGKDD Explor. Newsl.* [online]. 2014, vol. 15, no. 1, pp. 1–10 [visited on 2024-04-11]. ISSN 1931-0145. Available from DOI: `10.1145/2594473.2594475`.
4. HUYSMANS, Johan; DEJAEGER, Karel; MUES, Christophe; VANTHIENEN, Jan; BAESENS, Bart. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems* [online]. 2011, vol. 51, no. 1, pp. 141–154 [visited on 2024-04-11]. ISSN 0167-9236. Available from DOI: `https://doi.org/10.1016/j.dss.2010.12.003`.
5. RUDIN, Cynthia; CHEN, Chaofan; CHEN, Zhi; HUANG, Haiyang; SEMENOVA, Lesia; ZHONG, Chudi. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistic Surveys* [online]. 2022, vol. 16, pp. 1–85 [visited on 2024-04-12]. Available from arXiv: `2103.11251 [cs.LG]`.
6. ASHOORI, Maryam; WEISZ, Justin D. *In AI We Trust? Factors That Influence Trustworthiness of AI-infused Decision-Making Processes* [online]. 2019. [visited on 2024-04-13]. Available from arXiv: `1912.02675 [cs.CY]`.
7. THIEBES, Scott; LINS, Sebastian; SUNYAEV, Ali. Trustworthy artificial intelligence. *Electronic Markets* [online]. 2021, vol. 31, no. 2, pp. 447–464 [visited on 2024-04-13]. ISSN 1422-8890. Available from DOI: `10.1007/s12525-020-00441-4`.
8. ANGELINO, Elaine; LARUS-STONE, Nicholas; ALABI, Daniel; SELTZER, Margo; RUDIN, Cynthia. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research* [online]. 2018, vol. 18, no. 234, pp. 1–78 [visited on 2024-04-17]. Available from arXiv: `1704.01701 [stat.ML]`.
9. ELOMAA, Tapio. In Defense of C4.5: Notes on Learning One-Level Decision Trees. In: COHEN, William W.; HIRSH, Haym (eds.). *Machine Learning Proceedings 1994* [online]. San Francisco (CA): Morgan Kaufmann, 1994, pp. 62–69 [visited on 2024-04-24]. ISBN 978-1-55860-335-6. Available from DOI: `https://doi.org/10.1016/B978-1-55860-335-6.50016-7`.

10. SALZBERG, Steven L. C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993. *Machine Learning* [online]. 1994, vol. 16, no. 3, pp. 235–240 [visited on 2024-04-25]. ISSN 1573-0565. Available from DOI: `10.1007/BF00993309`.
11. WNEK, J. *MONK's Problems* [UCI Machine Learning Repository]. 1992. [visited on 2024-04-26]. Available from DOI: `https://doi.org/10.24432/C5R30R`.
12. HYAFIL, Laurent; RIVEST, Ronald L. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters* [online]. 1976, vol. 5, no. 1, pp. 15–17 [visited on 2024-04-26]. ISSN 0020-0190. Available from DOI: `https://doi.org/10.1016/0020-0190(76)90095-8`.
13. LIN, Jimmy; ZHONG, Chudi; HU, Diane; RUDIN, Cynthia; SELTZER, Margo. *Generalized and Scalable Optimal Sparse Decision Trees* [online]. 2022. [visited on 2024-04-26]. Available from arXiv: `2006.08690 [cs.LG]`.
14. KLIVANS, Adam R.; SERVEDIO, Rocco A. Toward Attribute Efficient Learning of Decision Lists and Parities. In: SHAWE-TAYLOR, John; SINGER, Yoram (eds.). *Learning Theory* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 224–238 [visited on 2024-04-29]. ISBN 978-3-540-27819-1. Available from DOI: `https://doi.org/10.1007/978-3-540-27819-1_16`.
15. MCTAVISH, Hayden; ZHONG, Chudi; ACHERMANN, Reto; KARIMALIS, Ilias; CHEN, Jacques; RUDIN, Cynthia; SELTZER, Margo. *Fast Sparse Decision Tree Optimization via Reference Ensembles* [online]. 2022. [visited on 2024-04-29]. Available from arXiv: `2112.00798 [cs.LG]`.
16. Simple Linear Regression. In: *Linear Models in Statistics* [online]. John Wiley & Sons, Ltd, 2007, chap. 6, pp. 127–136 [visited on 2024-05-03]. ISBN 9780470192610. Available from DOI: `https://doi.org/10.1002/9780470192610.ch6`.
17. SOKOLOVSKA, Nataliya; CHEVALEYRE, Yann; ZUCKER, Jean-Daniel. A Provable Algorithm for Learning Interpretable Scoring Systems. In: STORKEY, Amos; PEREZ-CRUZ, Fernando (eds.). *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics* [online]. PMLR, 2018, vol. 84, pp. 566–574 [visited on 2024-05-14]. Proceedings of Machine Learning Research. Available from: `https://proceedings.mlr.press/v84/sokolovska18a.html`.
18. USTUN, Berk; WESTOVER, M Brandon; RUDIN, Cynthia; BIANCHI, Matt. Clinical Prediction Models for Sleep Apnea: The Importance of Medical History over Symptoms. *Journal of clinical sleep medicine : JCSM : official publication of the American Academy of Sleep Medicine* [online]. 2015, vol. 12 [visited on 2024-05-15]. Available from DOI: `10.5664/jcsm.5476`.
19. USTUN, Berk; TRACÀ, Stefano; RUDIN, Cynthia. *Supersparse Linear Integer Models for Interpretable Classification* [online]. 2014. [visited on 2024-05-13]. Available from arXiv: `1306.6677 [stat.ML]`.
20. LOU, Yin; CARUANA, Rich; GEHRKE, Johannes; HOOKER, Giles. Accurate intelligible models with pairwise interactions. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* [online]. Chicago, Illinois, USA: Association for Computing Machinery, 2013, pp. 623–631 [visited on 2024-05-12]. KDD '13. ISBN 9781450321747. Available from DOI: `10.1145/2487575.2487579`.
21. NORI, Harsha; JENKINS, Samuel; KOCH, Paul; CARUANA, Rich. InterpretML: A Unified Framework for Machine Learning Interpretability [online]. 2019 [visited on 2024-05-12]. Available from arXiv: `1909.09223 [cs.LG]`.

22. RUDIN, Cynthia. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* [online]. 2019, vol. 1, no. 5, pp. 206–215 [visited on 2024-05-01]. ISSN 2522-5839. Available from DOI: `10.1038/s42256-019-0048-x`.

23. LAPUSCHKIN, Sebastian; WÄLDCHEN, Stephan; BINDER, Alexander; MONTAVON, Grégoire; SAMEK, Wojciech; MÜLLER, Klaus-Robert. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature Communications* [online]. 2019, vol. 10, no. 1 [visited on 2024-04-23]. ISSN 2041-1723. Available from DOI: `10.1038/s41467-019-08987-4`.

24. SCHRAMOWSKI, Patrick; STAMMER, Wolfgang; TESO, Stefano; BRUGGER, Anna; SHAO, Xiaoting; LUIGS, Hans-Georg; MAHLEIN, Anne-Katrin; KERSTING, Kristian. *Making deep neural networks right for the right scientific reasons by interacting with their explanations* [online]. 2024. [visited on 2024-04-23]. Available from arXiv: `2001.05371 [cs.LG]`.

25. ZECH, John R.; BADGELEY, Marcus A.; LIU, Manway; COSTA, Anthony B.; TITANO, Joseph J.; OERMANN, Eric Karl. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. *PLOS Medicine* [online]. 2018, vol. 15, no. 11, pp. 1–17 [visited on 2024-04-23]. Available from DOI: `10.1371/journal.pmed.1002683`.

26. FRIEDMAN, Jerome H. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* [online]. 2001, vol. 29, no. 5, pp. 1189–1232 [visited on 2024-05-15]. Available from DOI: `10.1214/aos/1013203451`.

27. CARUANA, Rich; LOU, Yin; GEHRKE, Johannes; KOCH, Paul; STURM, Marc; ELHADAD, Noemie. Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* [online]. Sydney, NSW, Australia: Association for Computing Machinery, 2015, pp. 1721–1730 [visited on 2024-04-24]. KDD '15. ISBN 9781450336642. Available from DOI: `10.1145/2783258.2788613`.

28. CHOI, Edward; BAHADORI, Mohammad Taha; SUN, Jimeng; KULAS, Joshua; SCHUETZ, Andy; STEWART, Walter. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. In: LEE, D.; SUGIYAMA, M.; LUXBURG, U.; GUYON, I.; GARNETT, R. (eds.). *Advances in Neural Information Processing Systems* [online]. Curran Associates, Inc., 2016, vol. 29 [visited on 2024-04-24]. Available from: `https://doi.org/10.48550/arXiv.1608.05745`.

29. FISHER, Aaron; RUDIN, Cynthia; DOMINICI, Francesca. *All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously* [online]. 2019. [visited on 2024-05-15]. Available from arXiv: `1801.01489 [stat.ME]`.

30. RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. *"Why Should I Trust You?": Explaining the Predictions of Any Classifier* [online]. 2016. [visited on 2024-05-15]. Available from arXiv: `1602.04938 [cs.LG]`.

31. SLACK, Dylan; HILGARD, Sophie; JIA, Emily; SINGH, Sameer; LAKKARAJU, Himabindu. *Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods* [online]. 2020. [visited on 2024-05-15]. Available from arXiv: `1911.02508 [cs.LG]`.

32. PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* [online]. 2011, vol. 12, pp. 2825–2830 [visited on 2024-05-12].

33. MCKINNEY, Wes. Data Structures for Statistical Computing in Python. In: WALT, Stéfan van der; MILLMAN, Jarrod (eds.). *Proceedings of the 9th Python in Science Conference* [online]. 2010, pp. 56–61 [visited on 2024-05-12]. Available from DOI: `10.25080/Majora-92bf1922-00a`.

34. *South German Credit* [UCI Machine Learning Repository]. 2020. [visited on 2024-05-11]. Available from DOI: `https://doi.org/10.24432/C5QG88`.

35. HOFMANN, Hans. *Statlog (German Credit Data)* [UCI Machine Learning Repository]. 1994. [visited on 2024-05-11]. Available from DOI: `https://doi.org/10.24432/C5NC77`.

36. JANOSI, Andras; STEINBRUNN, William; PFISTERER, Matthias; DETRANO, Robert. *Heart Disease* [UCI Machine Learning Repository]. 1988. [visited on 2024-05-11]. Available from DOI: `https://doi.org/10.24432/C52P4X`.

37. QUINLAN, Ross. *Thyroid Disease* [UCI Machine Learning Repository]. 1987. [visited on 2024-05-11]. Available from DOI: `https://doi.org/10.24432/C5D010`.

38. CHICCO, Davide; JURMAN, Giuseppe. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* [online]. 2020, vol. 21, no. 1, p. 6 [visited on 2024-05-12]. ISSN 1471-2164. Available from DOI: `10.1186/s12864-019-6413-7`.

39. WASKOM, Michael L. seaborn: statistical data visualization. *Journal of Open Source Software* [online]. 2021, vol. 6, no. 60, p. 3021 [visited on 2024-05-12]. Available from DOI: `10.21105/joss.03021`.

40. PROKHORENKOVA, Liudmila; GUSEV, Gleb; VOROBEV, Aleksandr; DOROGUSH, Anna Veronika; GULIN, Andrey. *CatBoost: unbiased boosting with categorical features* [online]. 2019. [visited on 2024-05-13]. Available from arXiv: `1706.09516 [cs.LG]`.

Contents of the attachment

```
│ practical ............................ practical part of the thesis
│ ├── impl ........................ source codes for the practical part
│ ├── data ......................................... used datasets
│ ├── README.md ...................................... user guide
│ └── requirements.txt ........................ required packages
└── thesis
  ├── src ................................ source files for the thesis
  └── thesis.pdf ........................................ full thesis
```