FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Model-agnostic methods for explaining local predictions of a black-box classifier
Student:	Adam Skluzáček
Supervisor:	Ing. Markéta Jůzlová
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of winter semester 2021/22

Instructions

Explainability methods for a machine learning model try to explain the decisions of the model. This can help a practitioner to build the model and user to adopt it. Many different explanation methods were proposed. The methods differ in output type, whether they are designed to a specific model or they are model-agnostic, thus, they can be applied to any black-box model, or whether they interpret the whole model or just local predictions. There are also different properties of the explainability methods, e.g. comprehensibility, stability, faithfulness to the interpreted model.

1. Review and describe model-agnostic techniques for black-box classification model interpretation. Focus on methods for explaining local predictions.
2. Use or implement at least three of the reviewed model-agnostic methods and experimentally compare their performance on various data sets and classifiers in terms of their faithfulness to the interpreted model.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D. Head of Department	doc. RNDr. Ing. Marcel Jiřina, Ph.D. Dean

Prague February 18, 2020

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Model-agnostic methods for explaining local predictions of a black-box classifier

Adam Skluzáček

Department of Applied Mathematics

Supervisor: Ing. Markéta Jůzlová

June 3, 2020

Acknowledgements

I would like to thank my supervisor Ing. Markéta Jůzlová for introducing me to the field of Explainable Artificial Intelligence and for her kind guidance throughout writing of this thesis. I wish to also thank my girlfriend, family and friends for their invaluable support and patience with my endless work on this thesis.

Declaration

In Prague on June 3, 2020 …………………

Citation of this thesis

Skluzáček, Adam. *Model-agnostic methods for explaining local predictions of a black-box classifier.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2020.

Abstrakt

Cílem lokálních univerzálních vysvětlovacích metod je vysvětlit jednotlivé predikce libovolného modelu strojového učení pouze za pomoci vstupů a odpovídajících výstupů daného modelu. Vysvětlování predikcí složitého modelu strojového učení pomáhá odborníkům vylepšovat daný model a zvyšuje uživatelskou důvěru v predikce modelu. Tato práce zkoumá tři z nejmodernějších lokálních univerzálních vysvětlovacích metod – LIME, Anchors a SHAP. Zkoumané metody jsou detailně popsány a experimentálně vyhodnoceny s ohledem na věrnost jejich vysvětlení vzhledem k vysvětlovanému modelu. Vyhodnocení je provedeno na různých klasifikátorech natrénovaných na uměle vygenerovaných datech i na reálných datech. Umělá data jsou vygenerována na základě známých závislostí, což umožňuje spočítat optimální vysvětlení a porovnat ho s vysvětleními vygenerovanými vysvětlovacími metodami. Výsledky experimentů ukazují, že SHAP je nejrobustnější vůči vlastnostem modelované funkce z uvažovaných vysvětlovacích metod. LIME i Anchors v určitých situacích neprodukují přesná vysvětlení, nicméně v experimentu s reálnými daty obě metody vyprodukovaly přesná vysvětlení.

Klíčová slova lokální, univerzální, vysvětlovací metody, interpretovatelné strojové učení, vysvětlitelná umělá inteligence, LIME, Anchors, SHAP

Abstract

Local model-agnostic explanation methods aim to explain a single prediction of an arbitrary machine learning model by studying the model only through its inputs and corresponding outputs. Explaining predictions of a complex machine learning model helps practitioner to debug the model and build user's trust in the predictions. This thesis reviews and describes three of the state-of-the-art local model-agnostic explanation methods – LIME, Anchors and SHAP. The described methods are evaluated in terms of faithfulness of their explanaions to the model being explained. Evaluation is performed on various classifiers trained on artificially generated datasets as well as a real-world divorce dataset. The artificial datasets are generated based on known dependencies which allows to calculate optimal explanations and compare them to the explanations produced by the explanation methods. The experiments show that SHAP is the most robust out of the considered explanation methods. LIME and Anchors fail to produce faithful explanations in specific cases, however, they both managed to produce faithful explanations in experiment with real-world dataset.

Keywords local, model-agnostic, explanation methods, interpretable machine learning, explainable artificial intelligence, LIME, Anchors, SHAP

Contents

List of Figures

List of Tables

Introduction

The recent years have witnessed a notable widespread of Artificial Intelligence across practically all the businesses, and machine learning models are becoming ubiquitous in people's everyday lives. Thanks to the rise of computational power and increasing sizes of datasets the state-of-the-art machine learning models have evolved from simple models such as Linear Regression, whose decision making can be usually understood even by laypeople, into complex models such as Artificial Neural Network with milions of parameters that contribute to the predictions. Such complex models are able to outperform humans on specific tasks, such as image classification or playing Go, however understanding their decision making is often unattainable even for machine learning experts. The uninterpretability makes for a challenging evaluation of other desired properties of machine learning models such as nondiscrimination in loan approval models or safety in models that drive autonomous vehicles. The emerging need for understanding the decision making of complex models has led to a developement of the field called Explainable Artificial Intelligence that has become one of the main interests of the recent AI research. Explaining a complex model allows for inspection of the model's decision making while retaining its high level of performance. Explanations also help users trust and adopt the model. This is crucial in fields such as medicine where the model's predictions are presented to a human decision maker who then makes the final decision.

Explanation methods can aim to explain model's behaviour for all possible inputs. Such explanations are called global and while they may be desirable and provide new valuable insight in tasks in which machine learning models generally outperform humans, it is usually unattainable for the complex model to be explained succinctly. Local explanations provide an often feasible fallback as they aim to explain model's behaviour only for a single prediction.

Model-agnostic explanation methods draw no assumptions about the underlying model, thus they can be applied on any machine learning model or even proprietary predictive system. The ability to explain arbitrary models with the same method is especially usefull for comparing multiple machine learning models and choosing the optimal model for deployment. Model-specific explanation methods, on the other hand, are tied to certain families of machine learning models and require full acces to the model's internals which allow such methods to be more computationally efficient.

This work focuses on local model-agnostic explanation methods that explain predictions of classification models for tabular input data. The goals of this work are to describe and compare three of the state-of-the-art local model-agnostic explanation methods – LIME, Anchors and SHAP. Explanation methods are evaluated in terms of faithfulness of their explanations to the model being explained.

The work is organized as follows. Chapter 1 is theoretical part, where in sections 1.1 through 1.3 and overview of the field of Explainable AI is given together with definitions of machine learning model, explainability and Black-box model. Taxonomy of explanation methods is described in section 1.4 and metrics for evaluating various properties of explanations are presented in section 1.5. The individual model-agnostic explanation methods are theoretically described in sections 1.6 and 1.7. In Chapter 2, the reviewed local model-agnostic explanation methods are evaluated in terms of faithfulness on artificial datasets and real-world divorce dataset.

Chapter 1

Theoretical part

1.1 Machine learning model and algorithm

Machine learning model is a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, where $\mathbb{R}^n$ is a n- -dimensional feature space and $\mathbb{R}$ is an output space. In the case of a classifier the output is either a class label or a probability mass function [1].

Family of machine learning models is a set of machine learning models with the same internal structure.

An example of a family of machine learning models is Logistic Regression used for binary classification, where the two classes are labeled as "0" and "1". The output is given by:

$$P(y = 1) = S(w^T x + b), \tag{1.1}$$

where $P(y = 1)$ denotes the probability of class 1, $x \in \mathbb{R}^n$ is the input vector, $w \in \mathbb{R}^n$ is the vector of weights, $b \in \mathbb{R}$ is the bias term and S denotes the sigmoid function defined by the formula: $S(x) = \frac{e^x}{e^x+1}$ [2]. Different machine learning models that are instaces of Logistic Regression have the same form of Equation 1.1 but differ in the weights and bias values (also called parameters of the model).

Machine learning algorithm adjusts parameters of a machine learning model in order for the model to perfrom a given task. An example of such task is disease diagnosis where the model is supposed to predict whether a given patient is sick or healthy. Performance of such model can be measured as the ratio of correctly diagnosed patients. The adjustment of model's parameters is also called training of the model. Supervised machine learning algorithm trains a model on a set of labeled data called training set to maximize the ratio of correct predictions of the model. Training set contains set of input vectors (patients) with corresponding target outputs (true diagnoses). Most machine learning algorithms also have parameters called hyperparameters whose values must be set before the machine learning algorithm is used. For example an algorithm that trains a tree ensemble model have hyperparameters such as number of trees or maximum depth of an individual tree.

1.2 Interpretability and Explainability of machine learning models

Large part of literature on Explainable Artificial Intelligence use terms interpretability and explainability interchangeably. However, Arrieta et. al. in [3] argue that these terms are being misused and they do not refer to the same concepts. They further argues that the misuse of these terms is one of the issues that hinders the establishment of common grounds in the field of Explainable Artificial Intelligence as there is a lack of consensus on definitions of these terms. For the purpose of this thesis interpretability and explainability are defined separately as follows.

Interpretability is a passive property of a machine learning model that refers to the level at which a given audience can understand model's decision making [3]. Interpretability is a characteristic of White-box models that are discussed in section 1.3.

Explainability of a machine learning model refers to the level at which the given model can be explained to a given audience by a certain procedure [3].

Condsider a task of disease diagnosis, where a Random Forest model is the machine learning model and a doctor is the audience. The doctor has to understand the decision making of the model in order to make the final diagnosis based on the model's prediction. If the doctor understands the prediction based on the structure of the Random Forest model, the model is interpretable. Otherwise the model has to be explained for example through feature importance scores that would tell the doctor how each feature influenced the model's diagnosis.

Explanation produced by an explanation method serves as an interface between an audience and an uninterpretable machine learning model. Being an explanation is not a passive property of statements as explanation is always an interaction that depends on the knowledge and the goals of its audience [3]. The knowledge of the audience mainly affects the type of the explanation, for example an audience with decent knowledge in statistics might be able to understand a Bayesian Network, however for people without such knowledge, sparse Decision Rule would be more appropriate [4].

The goals of the audience affects not only the type of the explanation but also it's granularity, for example an users to which a certain product is recommended might be satisfied with the most relevant product that they have recently purchased as an explanation. However a doctor performing a diagnosis might require a complete explanation of how each feature influenced the model's decision.

Explanations are produced after the model has been trained, thus such explanations are called post-hoc explanations. Post-hoc explanations are implicit and the term post-hoc is omitted throughout the thesis.

Many fields, such as philosophy and cognitive psychology have been long engaged in the questions such as how to define, generate or evaluate explanations. Social sciences therefore provide substantial insight into what kinds of explanations people generally want and understand. The field of Explainable Artificial Intelligence can benefit from building on this research which is reviewed in [5]. Relevant findings from cognitive psychology are mentioned throughout the thesis.

1.3 White-box and Black-box machine learning models

Machine learning model is referred to as Black-box when it is unattainable even for a machine learning practitioner to understand model's decision making by inspecting it's internals (parameters or submodels in case of ensembles). Thus, if we want to understand Black-box model, an explanation method must be employed. On the contrary model is referred to as White-box when it is interpretable for a given audience [3]. White-box models present an alternative to explainability methods as they are interpretable on their own [6].

Even though some families of machine learning models are generally deemed as White-box (e.g. Decision Trees), every machine learning algorithm can produce both White-box and Black-box models. For example a shallow Decision Tree with depth of 3 can be easily interpretable (Figure 1.1 shows an example of an interpretable Decision Tree), whereas a very deep Decision Tree will become uninterpretable and therefore Black-box. In order to interpret a Decision Tree's output a person has to simultaneously store and understand all the rules on a given path from a root node to a leaf node. An approximate depth threshold above which a Decision Tree becomes Black-box is 7 as an average person is said to be unable to store more than 7 pieces of information at a time [7]. In practice this threshold might be lower or higher depending on the audience and complexity of the node rules. Furthermore any model utilizing uninterpretable features such as components generated by Principal Component Analysis is inherently Black-box. Similarly Artificial Neural Networks are generally deemed as Black-box, however an Artificial Neural Network model without hidden layers that consists of couple of neurons may be interpretable.

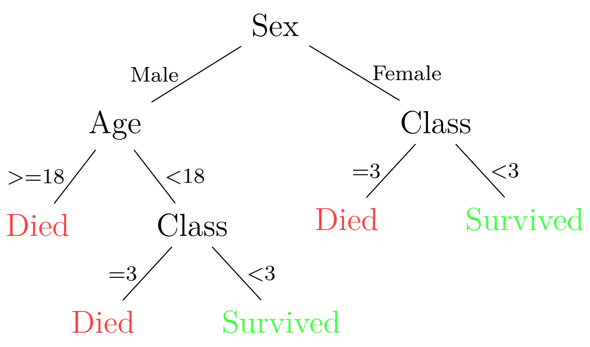

Figure 1.1: Decision Tree for prediction of whether a Titanic passenger have survived the sinking or not. All the paths from the root node to leaf nodes are easy to interpret, which makes this Decision Tree a White-box model.

1.3.1 Accuracy and Interpretability trade-off

Accuracy and interpretability trade-off refers to a consensus among the majority of machine learning practitioners that more accurate models usually tend to be less interpretable and vice versa [3].

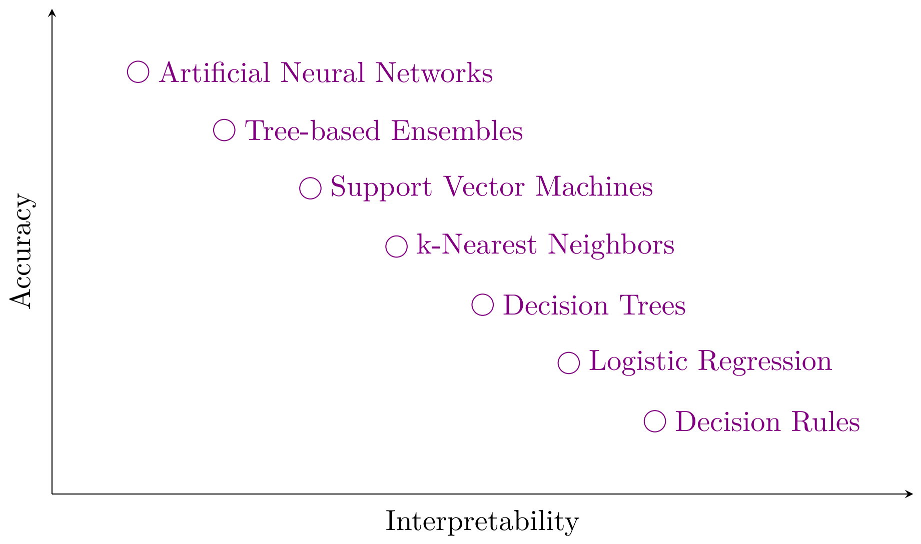

Figure 1.2: Visual representation of the trade-off between model accuracy and interpretability. All the points in the space are illustrative to show which families of machine learning models tend to be generally more accurate and less interpretable.

Some researchers point out that it is not necessarily true that more complex models are inherently more accurate than simple models. Explicitly in cases where the data is well structured and features at our disposal are of high quality and value [8].

Undeniably though complex models have higher capacity than their simpler counterparts, which gives them the flexibility to approximate more complex functions. Real world function that one wishes to approximate might entail such level of complexity a simple model is unable to represent [3].

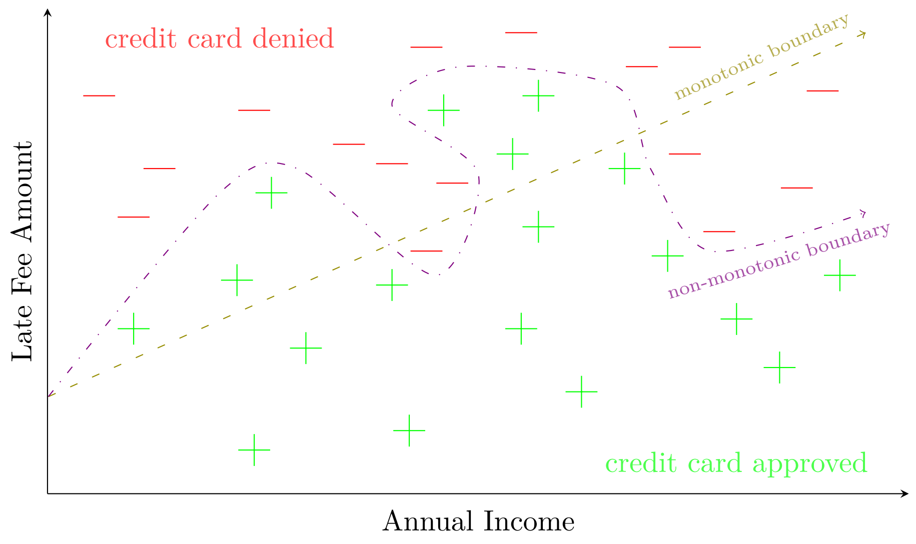

Figure 1.3: Example of a credit card approval task, where each point represents a single customer and plus and minus signs represent the target predictions. Monotonic boundary that could be represented by a White-box Logistic Regression model is easily interpretable but unable to capture the true decision boundary. To capture the true non-monotonic decision boundary, a more complex Black-box model has to be used (such as Random Forest or Artificial Neural Network) [9].

In most cases, model with the highest accuracy is an ensemble of several independently trained models as different models tend to be prone to different errors on the test set [1]. The power of Black-box methods is also evident from the interviews with machine learning competitions winners on kaggle.com who mostly use Ensemble models and Artificial Neural Networks [6]. In computer vision, Convolutional Neural Networks have outperformed humans on tasks such as image classification which is beyond the possibilites of White-box models [10].

Choosing explanation methods over a White-box model to achieve human understanding of the model's decision making allows for the deployment of the best performing model regardless of it's complexity.

1.3.2 Explanation flexibility

Explanation flexibility refers to the ability of different exlpanation methods to provide different types of explanations of the same machine learning model. Examples of explanation types are feature importance scores, the most influential training example for a given prediction, or a spare Decision Rule. In contrast White-box models are limited to only one type of explanation depending on the White-box model.

For example, if in order to achieve human understanding of the model a White--box Logistic Regression model is deployed, the only type of explanation it can provide are the weights of corresponding features (Equation 1.1). If a given audience required a different kind of explanation (for example a sparse Decision Rule for a different deparment of a corporation) an appropriate explanation method would have to be used to provide such explanation.

1.4 Explanation methods taxonomy

Explanation methods are primarily divided based on the scope of their explanations – whether they explain a machine learning model as a whole or only partially, and by model specificity – whether they can be applied to any machine learning model or only to a particular family of machine learning models.

Explanation methods can be further divided based on the type of their explanations (feature importance scores, Decision Rules, etc.), whether they can be applied to any data or only to tabular/text/image data and whether they can be applied on classification and/or regression tasks.

1.4.1 Explanation scope

1.4.1.1 Global explanation methods

Global explanation methods aim to explain a model as a whole. In other words they aim to help an audience to understand the full logic of the model [11]. The audience that is provided a global explanation should also be able to correctly simulate the model's prediction for any input. Global explanations can therefore provide new valuable insight and improve humans decision making in tasks where machine learning models generally outperform humans. However, it is often infeasible to provide an accurate and succinct global explanation of a complex model.

1.4.1.2 Local explanation methods

Local explanation methods aim to explain a single prediction of a model. The idea of local explanation methods is that while a given model is globally too complex to be explained succinctly, explaining model's decision making for an individual prediction makes the explanation task feasible [12]. Locally the predictions might only depend linearly or monotonically on some subset of features rather than having a complex dependence on all the features [6].

Local explanation of a single prediction may provide understanding of the model's global decision making, however, it is not sufficient to evaluate and assess trust in the model as whole. Local explanations of multiple predictions can provide some insight into the global decision making of the model if the chosen set of explanations is non-redundant and representative. Choosing such set of explanations may be a feasible task even when providing a global explanation of the model is unfeasible [12].

Local explanation methods are also useful in tasks where only explanations of single predictions are required. For example in loan-approval systems, financial institutions are obligated by law in both European Union and United States to provide a reason why a given loan application was denied [13, 14]. Explanation of a single loan application denial corresponds to a local explanation, while global explainability can be sacrificed for the sake of better performance.

1.4.2 Model specificity

1.4.2.1 Model-agnostic explanation methods

Model-agnostic explanation methods treats a model being explained as a black--box function. In this case term black-box does not refer to a complex uninterpretable model but rather to a model that can be studied only through its inputs and corresponding outputs without any knowledge of it's internals [4]. Model-agnostic explanation methods therefore provide a model flexibility as they can be applied to any predictive system regardless of its complexity or inner workings.

Model-agnostic methods also allow to study models that provide access only to its inputs and corresponding outputs via APIs. This limitation occurs in situations where a company treats a model as a proprietary software, thus is unwilling to release the whole model arguing that it contains company's trade secrets. For example COMPAS (Correctional Offender Management Profiling for Alternative Sanctions) is a proprietary recidivism prediction tool that is in widespread use in the U.S. Justice system for predicting the probability that a criminal will be arrested again after their release. COMPAS was not created by any standard machine learning algorithm, it was rather designed by experts based on carefully designed surveys and expertise, however, its internals are kept as a trade secret. Even though it is not a machine learning model, it is still a predictive system that can be studied through model-agnostic methods [8]. Model-agnostic analysis of the COMPAS have shown that the system is racially biased against black people [15].

When solving a particular machine learning task practitioners usually produce multiple models often generated by different machine learning algorithms. Competing models are subsequently compared in order to select the one that will have the best performance on the real data. This comparison is usually done by evaluating models' performances on a held-out subset of annotated data called test set. Relying solely on such evaluation can be misleading as model's performance on the test set may not correspond to its performance once the model is deployed "in the wild". One of many reasons for this might be an accidental information share between the sets training and testing data during preprocessing called Data leakage. Model-agnostic methods flexibility allows for two models generated by different machine learning algorithms to be explained by the same method. These explanations can then be used as a complementary method for comparison of the competing models [4, 12].

1.4.2.2 Model-specific explanation methods

Model-specific explanation methods are tied specifically to certain families of machine learning models and require full access to the model's internals. Their model specific design and ability to access model's internals (such as gradients) makes such methods generally more computationaly efficient than model-agnostic methods.

Model-specific explanation methods are especially interesting in the fields that are dominated by Deep learning, such as computer vision, natural language processing and one dimensional signal processing. The best performing models in such fields are almost exclusively versions of Artifical Neural Networks and therefore the flexibility of model-agnostic methods is unnecessary. Deep learning models benefits from the model-specific approach for two main reasons. First, Artifical Neural Networks learn features and concepts in their hidden layers that are not uncovered when using the model-agnostic approach. Second, the gradient can be utilized for higher computational efficiency [6].

1.5 Evaluation of explanation methods

There are two main properties of an explanation method that one can wish to evaluate – faithfulness and comprehensibility. Faithfulness refers to the ability of the explanation method to provide explanations that accurately capture model's true decision making. Comprehensibility refers to the ability of the explanation method to explain a given model in a human understandable fashion. There are other properties such as robustness. These are mentioned in the end of this section.

The gold standard for evaluating explanation's comprehensibility are human-based experiments. Objective and universal metrics for evaluating comprehensibility without human-based experiments and faithfulness of explanations remain an active field of research and one of the main challenges in the field of Explainable Artificial Intelligence.

1.5.1 Faithfulness

Faithfulness should always be the first assessed property of an explanation as other properties of an unfaithful explanation are meaningless. So far no universal metrics to evaluate faithfulness of explanations have been developed.

The current metrics of faithfulness evaluation can be divided into two categories presented in the subsections below.

1.5.1.1 Evaluation through known dependencies

Evaluation through known dependencies means that instead of evaluating faithfulness with respect to a Black-box model on a real task, the faithfulness is evaluated either through a White-box model or an artificial task with known dependencies. These evaluation methods are particularly usefull for general evaluation of explanation methods.

White-box models are interpretable, therefore the faithfulness of an explanation of a White-box model can be easily evaluated as the underlying model's decision making is known. For example Ribeiro et. al. in [12] trained Logistic Regression and Decision Tree classifiers such that the classifiers did not use more than 10 features. Since these features were considered the most important by the classifiers, they called them the gold features. Then they evaluated each explanation by computing explanation's recall on the gold features.

Artifical datasets generated by predefined dependencies can be leveraged for faithfulness evaluation in a similar fashion. If a machine learning model is able to learn the artifical task well enough (achieve high accuracy), evaluating explanations of such model is again simple as the data generating process is known.

1.5.1.2 Explanation-specific evaluation

Explanation-specific evaluation metrics can only be applied to certain types of explanations (e.g. Decision Rules). Such metrics are sufficient when a particular type of an explanation is required but can not be used for comparing explanations of different types.

1.5.1.2.1 Accuracy

Many global and local explanation methods provide a White-box machine learning model as an explanation. The explanatory model is trained on data annotated by a Black-box model. The goal of the White-box model is therefore to approximate the predictions of the underlying Black-box model. In such case accuracy, F1 scores and similar metrics can be measured by comparing the explanatory White-box model's predictions to the underlying Black-box model's predictions.

1.5.1.2.2 Replacing features with no-op values

IBM researchers in [16] proposed a metric for evaluating faithfulness of local feature importance explanations by gradually replacing input features with some no information value (no-op value) and measuring how the explained model's prediction probabilities change for each feature. For example for the MNIST dataset of handwritten digits, the no-op value can be 0 representing a black pixel.

Let $\theta = (\theta_{x_1}, \theta_{x_2}, \ldots, \theta_{x_n})$ denote the feature importance vector (local explanation), where θ is sorted in descending order such that $\theta_{x_1} >= \theta_{x_2} >= \ldots >= \theta_{x_n}$ and θ_{x_i} denotes the feature importance of input feature x_i. Let $p = (p_{x_1}, p_{x_2}, \ldots, p_{x_n})$ denote a vector, where p_{x_i} is the prediction probability after the value of input feature x_i is replaced by no-op value and let ρ denote Pearson correlation. The metric is then given by:

$$\phi = -\rho(\theta, p),$$

where higher value of ϕ indicates a better explanation.

The intuition behind this metric is that removing the most important feature in favor of the given prediction should lower the probability of such prediction more than removing the least important feature. However a similar intuition is behind most of the explanation methods that calculate feature importance and some methods even use the no-op values to calculate the feature importance as well. This method therefore evaluates an explanation through another explanation and it requires the model that is being explained to output probability of a given class, rather than just the class.

This metric, along with Monotonicity described in subsubsection 1.5.3.2, is included in IBM's open-source explainability framework AI Explainability 360.

The IBM researchers highlight the fact that out of all the Explainable Artificial Intelligence frameworks, theirs is the only one that includes any faithfulness evaluation metrics. This further highlights the difficulty of developing such metrics.

1.5.2 Comprehensibility

Velez and Kim in [17] propose three levels for evaluating comprehensibility of explanations – Application-grounded, Human-grounded and Functionally-grounded.

1.5.2.1 Application-grounded Evaluation: Domain experts, real tasks

Evaluating explanations with respect to the real task (the task for which a particular machine learning model was designed) by domain experts is deemed by many researchers as the best way of evaluating explanations comprehensibility. For example consider a task of diagnosing patients with a particular disease. The explanations would be presented to a doctor who would then evaluate whether the explanations justifies the model's predictions and whether the explanations are sufficient for the doctor to take actions based on the model's predictions.

1.5.2.2 Human-grounded Evaluation: Laypeople, simplified tasks

Another method for human-based evaluation of explanations' comprehensibility consist in conducting a simplified task that maintains the essence of the real task. Evaluations through a simplified task can be done by laypeople which is appealing when experiments with the domain experts are too expensive. An example of a simplified task is forward simulation, where laypeople are presented with an explanation and an input and tries to simulate the model's prediction. The explanation can then be evaluated based on whether people correctly simulated model's prediction and how quickly they were able to do it.

1.5.2.3 Functionally-grounded Evaluation: No humans, Proxy tasks

Human-based experiments require a substantial amount of time and effort which may be beyond the resources of a machine learning practitioner. Functionally-grounded evaluation uses some formal definition of comprehensibility as a proxy. Defining and quantifying comprehensibility is difficult and therefore the explanation complexity is usually evaluated instead. Many global and local explanation methods provide a White-box machine learning model as an explanation. Complexity of such explanation can then be evaluated with respect to the particular White-box model. For example complexity of a Decision Tree model can be measured as its depth or number of splits, whereas for Linear Regression number of features with non-zero weights can be measured. However it is unclear how to compare the complexity measures of different families of machine learning models. Moreover for some explanations, such as the most influential examples, the complexity metric does not exist so far.

1.5.3 Evaluation of other properties

1.5.3.1 Robustness

Robustness is measured as the change of an explanation after adding some noise into the input or to the model being explained. The evaluation metrics varies in what kind of noise is added. The most popular methods in the context of Explainable Artificial Inteligence are adding randomly valued features to the input space or adding Gaussian noise to the data [18].

1.5.3.2 Monotonicity

Monotonicity is the second metric in IBM's AI Explainability 360 framework for evaluating local feature importance explanations [16]. Monotonicity measures whether gradual provision of the features increases the probability of a given class. The method starts with no-op values input and incrementally replaces the features in increasing order of importance (provided by a local explanation) with their actual value. If the features are positively and independently correlated with the given class, then the class probabilities should monotonically increase every time an actual value of the feature is added to the input.

1.5.3.3 Usefulness in model selection

Ribeiro et. al. in [12] tried to simulate whether their local explanation method can aid in selecting the better performing model as one of the methods to evaluate their explanation method. First they have added noisy binary features, such that they appeared in 10% training examples of one class, 20% training examples of the other class and in 10% testing examples of both classes. The idea was to simulate the situation where the model uses spurious correlations that doesn't appear in the real world. Then they've trained two classifiers such that their validation accuracies were within 0.1% of each other, while their test set accuracies differed by at least 5%. The similar validation accuracies made it impossible to choose a better model via only validation accuracies. Subsequently they have used the explanation method to provide explanations of both classifiers on all the validation data and marked each explanation as untrustworthy if it treated at least one of the noisy features as an important feature and trustworthy otherwise. The simulated user then picked the model with less untrustworthy explanations. Finally the authors measured how many times the simulated user picked the better classifier. They also measured how the probability of picking the better classifier changed with the number of explanations provided to the simulated user.

1.6 Global model-agnostic explanation methods

Global model-agnostic explanation methods aim to explain a machine learning model for all possible inputs by studying the model only through its inputs and corresponding outputs. While it is usually unatainable to globally explain a complex model, the global surrogate explanation model is described in the section below as many local explanation methods are build upon the same idea.

1.6.1 Global surrogate explanation model

A global surrogate model is a machine learning model that approximates a Black-box machine learning model that is being explained. The idea is that when the global surrogate model is a White-box model and accurately approximates the model that is being explained, then it can be used as an explanation of the underlying Black-box model.

The pseudocode of generating a global surrogate explanation model is given in Algorithm 1. To generate an explanatory global surrogate model an appropriate family of machine learning model, training algorithm and its hyperparameters have to be chosen first, in order to provide a model that is interpretable to a given audience. Next, a training set of data for training the surrogate model has to be chosen. The training set for the surrogate model can be arbitrarily large, since it is annotated by the underlying Black-box model. Often enough in order to prevent overfitting and achieve high generalization, a model requires higher capacity than is necessary for the task due to the limited number of the training examples. Thanks to the theoretically unlimited number of training examples for the surrogate model, it is possible for the surrogate model to approximate the Black-box model while remaining interpretable [1]. Note that this is possible only in cases when the function represented by the Black-box model is not too complex to be represented by a White-box model as illustrated in Figure 1.3.

It is neccessary to choose a testing set for the surrogate model, also annotated by the model being explained, and an accuracy metric to use to asses the quality of surrogate model's approximation. Finally, an accuracy threshold bellow which the surrogate model is insufficient as an explanation should be chosen. The optimal accuracy threshold value is unclear and even when the suroggate model's accuracy seems high, there can be a subset of data for which the surrogate model is highly inaccurate [6].

Algorithm 1 Global surrogate explanation model

```
Require: Machine learning algorithm $A$ with hyperparameters $h$
Require: Training set $\Pi_{train}$ and testing set $\Pi_{test}$ annoted by the model
    being explained
Require: Accuracy metric $accuracy$ and accuracy threshold $t$
1: $g \leftarrow A_h(\Pi_{train})$                    ▷ Train the global surrogate model
2: if $accuracy(g(\Pi_{test})) > t$ then              ▷ Accuracy condition
       return $g$
3: else
       return ”Insufficient accuracy, choose a different machine learning
    algorithm or generate additional training data”
```

1.7 Local model-agnostic explanation methods

Local model-agnostic explanation methods aim to explain a single prediction of a machine learning model by studying the model only through its inputs and corresponding outputs. This chapter describes three of the most popular and widely cited local model-agnostic methods for explaining predictions of classification models. LIME method described in section 1.7.1 produces an explanation in form of a sparse linear local surrogate model. The local surrogate model is also utilized by Anchors method described in section 1.7.2 that produces a sparse Decision Rule as an explanation. SHAP method for calculating feature importance scores based on Shapley values from game theory is described in section section 1.7.3.

1.7.1 LIME

LIME is a local model-agnostic explanation method created by Ribeiro et. al. [12]. LIME explains a given prediction of a Black-box classifier by producing a sparse linear model that is locally faithful the Black-box model. LIME is essentially a linear local surrogate model that tries to approximate the Black-box model around the input of a given prediction, rather than trying to approximate the Black-box model for all the possible inputs like global surrogate models.

Subsection 1.8.1.1 describes a general framework for producing a local surrogate explanation models. This framework was originally referred to as LIME with Sparse linear explanations beeing only an instance of the framework that uses linear model as the surrogate model. However, after the paper was published, the Sparse linear explanations began to be called LIME even by the authors themselves [19].

1.7.1.1 Local surrogate explanation model

The goal of a local surrogate model is to approximate a Black-box model only around a given prediction, therefore the locality around the prediction has to be emphasized during the training of the surrogate model. This can be achieved either by sampling the training data only around the corresponding input of the prediction or by weighting the training samples by some proximity measure. LIME uses the later approach where the training data for the surrogate model are sampled from the distribution of the training data of the model being explained and weighted by a proximity measure such that instances farther from the input being explained are assigned lower weights. Each of the input features are sampled individually from Gaussian distribution with their means and standard deviations taken from the training data of the underlying Black-box model.

Formally, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ denote the model being explained, $x \in \mathbb{R}^n$ denote the input of the prediction being explained, $f(x)$ denote the corresponding prediction (class label or probability of a class) and $\pi_x : \mathbb{R}^n \mapsto \mathbb{R}$ denote the proximity measure between x and another instance from $\mathbb{R}^n$.

Furthemore, let $g \in G$ denote an explanatory local surrogate model that belongs to a family of potentialy White-box models G such as Decision Rule or Linear Regression. As discussed in section 1.3 every family of machine learning models contains both White-box and Black-box models. Therefore in order to ensure that the surrogate model is interpretable, let $\Omega(g)$ denote a complexity metric of a model g. The choice of a complexity metric depends on the chosen family of models G, for Decision Rules $\Omega(g)$ may be the number of conditions, while for Linear Regression, $\Omega(g)$ may be the number of features with non-zero weights.

Finally, let $\mathcal{L}(f, g, \pi_x)$ be a loss function that measures how unfaithful the surrogate model g is in approximating the underlying Black-box model f in the locality defined by the proximity measure π_x. The explanatory local surrogate model is then given by:

$$\hat{g}(x) = \underset{g \in G}{\arg\min}\, \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{1.2}$$

This formulation optimizes both interpretability and local faithfulness in order to find an optimal trade-off between interpretability and accuracy as discussed in subsection 1.3.1. Thresholds can also be introduced to guarantee a certain level of interpretability or local faithfulness.

1.7.1.2 Sparse linear explanations

The Equation 1.2 allows for arbitrary choice of a family of machine learning models G, local faithfulness measure $\mathcal{L}$, proximity measure π_x and complexity metric Ω. This section describes the configuration chosen by Ribeiro et. al. originally called Sparse linear explanations that is implemented in the LIME framework available on GitHub. The illustrative example of Sparse linear explanation (LIME) is given in Figure 1.4.

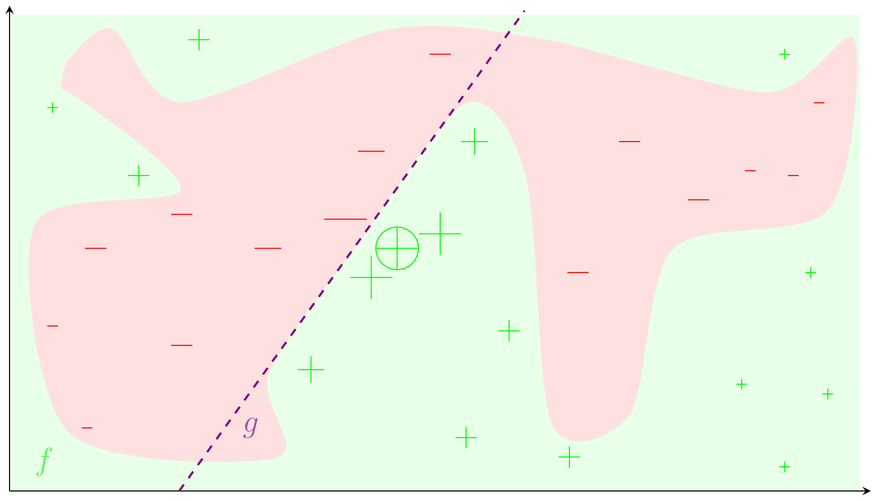

Figure 1.4: Illustration of Sparse linear explanation (LIME). The green and red background shows the decision boundary of the underlying Black-box classifier f, the points in the space represent the samples for training the local surrogate model g with sizes of the samples indicating their weights. In this example LIME is able to produce an explanation model g (violet line) that is locally accurate around the instance being explained (circled point). However in cases where the underlying Black-box model is highly non-linear, LIME may be unable to produce an explanation that is locally faithful.

The pseudocode of producing a sparse linear explanation (LIME) is given in Algorithm 2. As the name suggests the choice of G is a family of Linear Regression models, where the explanatory local surrogate model is given by $g(x') = w_g^T x'$. The $x' \in \mathbb{R}^d$ denotes a sparse representation of input $x \in \mathbb{R}^n$ that contains the $d \leq n$ most important features of x. The sparse representation is often required in order to produce an interpretable explanation, as the original representation x might be high dimensional. When n is low enough d can be set as $d = n$. The number of features used in the explanation is forced by the complexity measure Ω given by $\Omega(g) = \infty \mathbb{1}_{\|w_g\|_0 > d}$, where $\|w_g\|_0$ denotes the ℓ_0 "norm" defined as the number of non-zero elements of a vector. This particular choice of Ω makes solving Equation 1.2 directly intractable as the number of possible feature combinations grows exponentially with d. Therefore an approximation is produced instead where d of the most important features are preselected. The d most important features are by default chosen by training a locally weighted Linear Regression with L2 regularization

(Ridge Regression) and then selecting the d features with the highest weights.

After the d features are selected the explanatory model is obtained by minimizing the locally weighted square loss with L2 regularization $\mathcal{L}$, where the weights are given by the the exponential kernel π_x on Euclidean distance D.

$$\mathcal{L}(f, g, \pi_x) = \sum_{z,z' \in \mathcal{Z}} exp\left(\frac{-D(x,z)^2}{\ell^2}\right)\left(f(z) - g(z')\right)^2 + \lambda \|w_g\|_2^2, \tag{1.3}$$

where $\mathcal{Z}$ denotes the training set for the explanatory model sampled from the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with the mean μ and variance σ^2 of each feature taken from the underlying model's training set and ℓ denotes the kernel width which is by default defined as $\ell = 0.75\sqrt{d}$.

Algorithm 2 LIME (Sparse linear explanations)

```
Require: Classifier $f$, Input $x$ of the prediction being explained
Require: Number of training samples $M$, Length of explanation $d$
Require: Means $\mu$ and variances $\sigma^2$ of all features from the classifier's $f$
    training data
1: $\mathcal{Z} \leftarrow \{\}$
2: for $i \in \{1, 2, \ldots, M\}$ do
3:     $z_i \leftarrow \mathcal{N}(\mu, \sigma^2)$
4:     $\mathcal{Z} \cup \langle z_i, f(z_i), \pi_x(z_i) \rangle$
5: $g_{feature\_selection} \leftarrow$ Ridge($\mathcal{Z}$)
6: Select the $d$ features with the highest weights in $g_{feature\_selection}$
7: $g \leftarrow$ Ridge($\mathcal{Z}$)                ▷ with $z'_i$ as features and $f(z)$ as target
8: return $g$
```

1.7.2 Anchors

Anchors is a local model-agnostic explanation method created by the same researchers as LIME - Ribeiro et. al. [19]. Anchors explain a given prediction of a Black-box classifier by producing a Decision Rule. The motivation behind the method is that explanations from other local explanation methods have unclear coverage. For example a local explanation produced by the LIME method can be locally faithful, however, it is unclear whether the explanation also applies to some other instance. Ribeiro et. al. argue that the unclear coverage can mislead the audience as they may think that the explanation applies to some other instance when it does not. Anchors aim to produce a Decision Rule such that features that are not included in the rule's predicates does not influence the prediction of the model being explained. Such explanation have clear coverage as it applies to all instances that satisfy the rule's conditions as illustrated in Figure 1.5. Decision Rules are also highly interpretable for humans which makes them a goot fit for explanations [20].

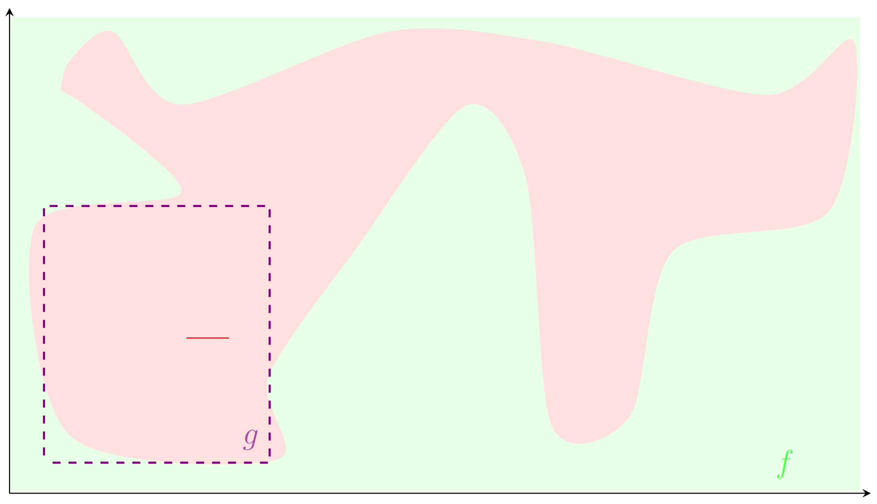

Figure 1.5: Illustration of Anchors explanation. The green and red background shows the decision boundary of the underlying Black-box classifier f, the point in the space shows the instance being explained x and the explanation rule g produced by the Anchors method is illustrated as the purple rectangle. The rule's coverage is clear as it only applies to instances inside the rectangle (rule's scope) where the rule is also highly efficient.

Explanations produced by Anchors are essentially local surrogate models with Decision Rules as the family of explanatory machine learning models. However the method is not based on the local surrogate explanation model framework described in subsection 1.8.1.1. Instead, the method utilizes techniques from reinforcement learning and graph search in order to be computationally efficient.

Formally, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ denote the model being explained, $x \in \mathbb{R}^n$ denote the input of the prediction being explained and $f(x)$ denote the corresponding prediction (which must be a class label). Let A be a Decision Rule, such that $A(x)$ returns 1 if x satisfies all the rule's predicates. Moreover, let $\mathcal{D}$ denote a distribution from which the training samples for the explanation are drawn and let $\mathcal{D}(\cdot|A)$ denote a conditional distribution when the rule A is satisfied. For a desired level of precision τ, explanation rule A must satisfy the following:

$$A(x) = 1, \, \mathbb{E}_{\mathcal{D}(z|A)}[\mathbb{1}_{f(x)=f(z)}] \geq \tau, \tag{1.4}$$

where $\mathbb{E}_{\mathcal{D}(z|A)}[\mathbb{1}_{f(x)=f(z)}] = prec(A)$. For a highly dimensional input space it is infeasible to evaluate $\mathbb{1}_{f(x)=f(z)}$ for all $z \in \mathcal{D}(\cdot|A)$. Therefore a probabilistic precision is evaluated instead with parameter $\delta \in (0, 1)$ specifing the desired level of statistical confidence:

$$P(prec(A) \geq \tau) \geq 1 - \delta. \tag{1.5}$$

From all the rules that satisfy this precision condition, the explanation is selected as the rule A with the highest coverage $cov(A)$:

$$\max_{A \; s.t. \; P(prec(A) \geq \tau) \geq 1-\delta} \mathbb{E}_{\mathcal{D}(z)}[A(z)], \tag{1.6}$$

where $\mathbb{E}_{\mathcal{D}(z)}[A(z)] = cov(A)$. Through maximizing the coverage, interpretability of the explanation is indirectly also maximized as Decision Rules with less predicates tend to have higher coverage. On the other hand, Decision Rules with more predicates tend to have lower coverage and higher precision which yields a trade-off between precision and coverage.

Evaluating $A(z)$ for all $z \in \mathcal{D}$ is infeasible, therefore the $cov(A)$ is estimated based on a fixed number of samples as the ratio of samples that satisfy the given rule A. However, solving Equation 1.6 directly remains intractable as the number of all possible rules grows exponentially with n (number of features).

In order to produce an explanation that is an approximation of the Equation 1.6, Beam search over the space of potential explanations is performed. Beam search is a search algorithm that uses a heuristic to expand only B best candidates in each depth [21]. The Beam search therefore have two main parts. In the first part, a new set of rule candidates is generated out of the set of B best rule candidates from the previous step and in the second part, new B best rule candidates with the highest precision are selected for the next step of the Beam search. Algorithms for both of these parts are described below.

Generating rule candidates by expanding rules The pseudocode of generating rule candidates is given in Algorithm 3. The algorithm starts with an empty set of rules $\mathcal{A}_{cand}$ and in each iteration, each rule $A \in \mathcal{A}$ is extended by one additional feature predicate for each predicate a_i, such that a_i is satisfied by x and $cov(A \wedge a_i) > cov(A^*)$, where A^* denotes the best rule found so far with respect to the Equation 1.6. This condition prunes the search space as the rule's coverage can only decrease when adding more predicates. For the purpose of generating feature predicates, continuous features are discretized into bins of equal height (quartiles or deciles) as predicates are generated for each value of each feature.

Algorithm 3 Generating rule candidates by expanding rules

```
Require: Input $x$ of the prediction being explained
Require: Set of the current candidate rules $\mathcal{A}$
Require: Coverage of the best rule found so far $cov(A^*)$
1: function GenerateCandidates($x$, $\mathcal{A}$, $cov(A^*)$)
2:     $\mathcal{A}_{cand} \leftarrow \emptyset$
3:     for all $A \in \mathcal{A}$ do
4:         for all $a_i$ s.t. $a_i$ is satisfied by $x$ and $a_i \notin A$ do
5:             if $cov(A \wedge a_i) > cov(A^*)$ then    ▷ Prunning of the search space
6:                 $\mathcal{A}_{cand} \leftarrow \mathcal{A}_{cand} \cup (A \wedge a_i)$
7:     return $\mathcal{A}_{cand}$
```

Selecting the best candidate rules After generating the set of rules $\mathcal{A}$, B candidate rules with the highest precision from $\mathcal{A}_{cand}$ have to be selected for the next iteration. As disscused earlier, it is infeasible to evaluate all the samples from $\mathcal{D}(\cdot|A)$ to calculate the true precision of a rule A and therefore an approximation have to be calculated instead. The approximation of the rule's A precision is calculated on a limited number of samples $z \in \mathcal{D}(\cdot|A)$. Choosing a fixed number of samples may lead to inaccurate estimations in some cases and high computational complexity in other cases. The goal is to accurately approximate precisions using a minimal number of samples from $\mathcal{D}$. To achieve this goal, the problem of selecting B rules with the highest precision is reformulated as an instance of explore-m multi-armed bandit problem.

Multi-armed bandit problem is a reinforcement learning problem inspired by a gambler who can choose from a number of one-armed bandit slot machines. Each slot machine yields a reward with a different probability. The reward probabilities are unknown to the gambler, however, he is allowed to play the slot machines arbitrarily. In the explore-m setting, gambler's goal is to identify a subset of the highest rewarding slot machines with a minimal number of plays [22]. Formally, consider a finite number of arms $K \geq 2$, where each sample of arm k yields a reward generated from a Bernoulli distribution with mean $p_k \in [0, 1]$. The arms can be sorted based on the corresponding means such that $p_1 \geq p_2 \geq \ldots \geq p_K$. The goal is to identify m arms with the highest means using a minimal number of samples. In the context of selecting rules with the highest precision, each arm k corresponds to a rule A, mean p_k corresponds to precision $prec(A)$ and sampling arm k corresponds to drawing a sample $z \in \mathcal{D}(\cdot|A)$ and evaluating $\mathbb{1}_{f(x)=f(z)}$ [23].

To solve this task, Ribeiro et. al. proposed to use the KL-LUCB algorithm. Informally, the KL-LUCB algorithm works by constructing confidence regions for the precision estimate of each rule. In each iteration, the current set of B best rules is selected and only the two critical rules are sampled - the worst rule from the set of the best rules and the best rule outside of the set of the best rules. The iterations stop when the critical rules' confidence regions are far enough apart and the current set of B best rules is returned.

The pseudocode of KL-LUCB algorithm is given in Algorithm 4. In the first step, the algorithm samples $z \in \mathcal{D}(\cdot|A)$ for each rule A to initialize the precision estimate $\widehat{prec}(A)$ as well as confidence region's bounds $prec_{lb}(A)$ and $prec_{ub}(A)$ computed based on Kullback-Leibler divergence:

$$\begin{aligned} prec_{lb}(A) &= min\{q \in [0, \widehat{prec}(A)] : N_A(t) KL(\widehat{prec}(A), q) \leq \beta(t, \delta)\}, \\ prec_{ub}(A) &= max\{q \in [\widehat{prec}(A), 1] : N_A(t) KL(\widehat{prec}(A), q) \leq \beta(t, \delta)\}, \end{aligned} \tag{1.7}$$

where t denotes the time (step of the algorithm), $N_A(t)$ denotes the number of samples $z \in \mathcal{D}(\cdot|A)$ drawn up to time t (number of updates of $\widehat{prec}(A)$), δ denotes the desired level of statistical confidence on the precision estimate, KL denotes the Kullback-Leibler divergence between two Bernoulli distributions, given as $KL(p_x, p_y) = p_x \mathrm{log}(\frac{p_x}{p_y}) + (1-p_x)\mathrm{log}(\frac{1-p_x}{1-p_y})$ and β denotes the exploration rate which is by default defined as $\beta(t, \delta) = log(\frac{kt^\alpha K}{\delta}) + log\, log(\frac{kt^\alpha K}{\delta})$, where $\alpha > 1$ and $k > 1 + \frac{1}{\alpha - 1}$ are parameters [24].

Then in each iteration, the algorithm selects the set $\mathcal{A}_{best}$ of B rules with the highest precision estimates and two critical rules A_u and A_l such that:

$$A_u = \underset{A \notin \mathcal{A}_{best}}{\arg\max}\, prec_{ub}(A),\ A_l = \underset{A \in \mathcal{A}_{best}}{\arg\min}\, prec_{lb}(A) \tag{1.8}$$

In other words, rule A_u is the rule with the highest precision upper bound outside of $\mathcal{A}_{best}$ and rule A_l is the rule with he lowest precision lower bound from $\mathcal{A}_{best}$. In each iteration, only the two critical rules are sampled and updated and the iterations stops after the difference between the precision upper bound of A_u and the precision lower bound of A_l is lower than a selected tolerance $\epsilon \in [0, 1]$. The KL-LUCB algorithm returns a set $\mathcal{A}_{best}$ of B rules that satisfy the following (proved in [24]):

$$P(\min_{A \in \mathcal{A}_{best}} prec(A) \geq \min_{A' \in \mathcal{A'}_{best}} prec(A') - \epsilon) \geq 1 - \delta, \tag{1.9}$$

where $\mathcal{A'}_{best}$ is the true set of B rules with the highest precision.

Algorithm 4 Selecting B best rule cadidates via KL-LUCB algorithm

```
Require: Classifier $f$, Input $x$ of the prediction being explained
Require: Distribution $\mathcal{D}$ of the training data for explanation
Require: Set of rules $\mathcal{A}$, Number of rules to be chosen $B$
Require: Desired level of statistical confidence $\delta$
Require: Tolerance $\epsilon$, Exploration rate $\beta$
 1: function B-BestCandidates($f, x, \mathcal{A}, \mathcal{D}, B, \delta, \epsilon, \beta$)
 2:     for all $A \in \mathcal{A}$ do                                  ▷ Initialization
 3:         sample $z \in \mathcal{D}(\cdot|A)$
 4:         initialize $\widehat{prec}(A), prec_{lb}(A), prec_{ub}(A)$
 5:         initialize $\mathcal{A}_{best}$, $A_u$, $A_l$
 6:     while $prec_{ub}(A_u) - prec_{lb}(A_l) > \epsilon$ do
 7:         sample $z_u \in \mathcal{D}(\cdot|A_u)$
 8:         update $\widehat{prec}(A_u), prec_{lb}(A_u), prec_{ub}(A_u)$
 9:         sample $z_l \in \mathcal{D}(\cdot|A_l)$
10:         update $\widehat{prec}(A_l), prec_{lb}(A_l), prec_{ub}(A_l)$
11:         update $\mathcal{A}_{best}$, $A_u$, $A_l$
12:     return $\mathcal{A}_{best}$
```

The final pseudocode of the Beam search for generating the explanation rule is given in Algorithm 5. The Beam search iteratively search the space of potential candidate rules using the GenerateCandidates method for generating a new set of rule cadidates and B-BestCandidates method for selecting the B rules with the highest precision. In the end of each iteration, the best rule A^* is potentially updated as the rule with the precision lower bound higher than the desired level of rule precision τ and the highest coverage found so far. The search ends when there is no rule candidate that could potentially surprass the best rule found so far.

Algorithm 5 Anchors - Beam Search

```
Require: Classifier $f$, Input $x$ of the prediction being explained
Require: Distribution $\mathcal{D}$ of the training data for explanation
Require: Beam width $B$
Require: Desired levels of precision $\tau$ and statistical confidence $\delta$
Require: Tolerance $\epsilon$, Exploration rate $\beta$
 1: $\mathcal{A}_0 \leftarrow \emptyset$                                  ▷ Initialize the set of candidate rules
 2: $A^* \leftarrow$ null                                ▷ Initialize the best rule found so far
 3: loop
 4:     $\mathcal{A}_{tcand} \leftarrow$ GenerateCandidates($x, \mathcal{A}_{t-1}, cov(A^*)$)
 5:     $\mathcal{A}_t \leftarrow$ B-BestCandidates($f, x, \mathcal{A}_{tcand}, \mathcal{D}, B, \delta, \epsilon, \beta$)
 6:     if $\mathcal{A}_t = \emptyset$ then                               ▷ No candidate rules left
 7:         break loop
 8:     for all $A \in \mathcal{A}_t$ s.t. $prec_{lb}(A) > \tau$ do                ▷ Satisfy Equation 1.5
 9:         if $cov(A) > cov(A^*)$ then                  ▷ Highest coverage found so far
10:             $A^* \leftarrow A$
11: return $A^*$
```

1.7.3 SHAP

Shapley Additive Explanations (SHAP) is a local model-agnostic explanation method that explains a given prediction of a Black-box model through feature importance scores. Feature importance explanation shows how each of the input's features influenced the prediction. Such explanations are popular as there are many model-agnostic and model-specific methods for calculating global or local feature importance scores [3]. SHAP is based on Shapley values that come from cooperative game theory. Cooperative game is a game where multiple players cooperate in order to acquire a payout and Shapley values show how to fairly divide the payout between the individual players based on their contributions to the payout. When used on a machine learning model, the Shapley values show the influence that each feature has on the prediction.

Formally, a cooperative (or coalitional) game is a tuple $\langle P, v \rangle$, where P is a finite set of players and $v : 2^P \mapsto \mathbb{R}$ is a characteristic function that maps every possible subset of players $S \subseteq P$ (coalition) to a number representing coalition's collective payout. The characteristic function must satisfy that $v(\emptyset) = 0$, where $\emptyset$ denotes the empty set of players. Intuitively, the marginal contribution to the payout of a player i in a coalition S could be obtained as the difference between the payouts of coalitions with and without the player i: $v(S \cup \{i\}) - v(S)$. However, this equation calculates the marginal contribution only in the situation when the player i joins the coalition as the last player. The marginal contribution of the player i may therefore be too low as the player's skillset could already be present in the coalition S before player's addition to the coalition. Similarly, if the player i is added as the first player to the coalition, the player's marginal contribution may be too high, bringing down other players' contributions. Shapley value $\phi_i(v)$ is defined as the average marginal contribution of the player i over all possible permutations of the coalition. Shapley value $\phi_i(v)$ is given by:

$$\phi_i(v) = \sum_{S \subseteq P \setminus \{i\}} \frac{|S|!(|P| - |S| - 1)!}{|P|!} (v(S \cup \{i\}) - v(S)), \tag{1.10}$$

where $|S|$ denotes the number of elements of S. Subset S represents players that were picked before the player i and subset $P \setminus \{S \cup \{i\}\}$ represents players that were picked after the player i. Since the player i is not affected by the exact order within the subsets of the players picked before and after the player i, the payout differences $v(S \cup \{i\}) - v(S)$ are weighted by the term ${}^{|S|!(|P|-|S|-1)!}/_{|P|!}$ that calculates how many permutations are represented by the particular pick of S.

Lloyd Shapley has proved in [25] that the Shapley values are the only solution to the problem of fair distribution of the payout that satisfy the following axioms.

Efficiency axiom The sum of all Shapley values is equal to the collective payout:

$$\sum_{i \in P} \phi_i(v) = v(P) \tag{1.11}$$

Symmetry axiom If two players increase the payout of every coalition by the same amount, then their Shapley values are equal:

$$\forall i, j \in P : \forall S \subset P \;\; v(S \cup \{i\}) = v(S \cup \{j\}) \land i, j \notin S \Rightarrow \phi_i(v) = \phi_j(v) \tag{1.12}$$

Dummy axiom If a player does not increase the payout of any coalition, then its Shapley value is equal to zero:

$$\forall i \in P : \forall S \subset P \;\; v(S \cup \{i\}) = v(S) \Rightarrow \phi_i(v) = 0 \tag{1.13}$$

Additivity axiom If a characteristic function of a game can be represented as a sum of two charactaristic functions, then the corresponding Shapley values can be decomposed in the same way:

$$\begin{aligned} \forall \langle P, v + w \rangle : \forall S \subset P \;\; (v + w)(S) = v(s) + w(S) \Rightarrow \\ \Rightarrow \forall i \in P \;\; \phi_i(v + w) = \phi_i(v) + \phi_i(w) \end{aligned} \tag{1.14}$$

These axioms demonstrate that SHAP explanations are built on a solid theory which may make them more appealing in some use cases or to some audiences [6]. Patrick Hall, who is a co-founder of an AI-focused law firm called bnh.ai, suggests using SHAP explanations in fields regulated by the law such as banks or insurance companies [26].

The Equation 1.10 can not be directly used to generate an explanation of a machine learning model as most of the machine learning models can not handle missing data and must be provided with a value for each input feature. Moreover, the Equation 1.10 requires evaluation of every subset which is unfeasible as the number of subsets grows exponentially with the number of players (features).

Štrumbelj et. al. in [27, 28] propose an approach to overcome both of these issues. The pseudocode of their algorithm for generating explanations based on Shapley values is given in Algorithm 6. First, an equivalent formulation of the Equation 1.10 that computes the average marginal contribution over the set of all ordered permutations $\Psi(P)$ of players P is given by:

$$\phi_i(v) = \frac{1}{|P|!} \sum_{\psi \in \Psi(P)} (v(Pre^i(\psi) \cup \{i\}) - v(Pre^i(\psi))), \tag{1.15}$$

where $Pre^i(\psi)$ denotes the set of all players that precede the player i in permutation $\psi \in \Psi(P)$.

Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ denote the model being explained, $x \in \mathbb{R}^n$ denote the input of the prediction being explained and $f(x)$ denote the corresponding prediction (class label or probability of a class). Let the set of players be represented as the set of n features denoted by their corresponding feature indexes $N = \{1, 2, \dots n\}$ and the payout be represented as the difference between the prediction being explained $f(x)$ and the expected prediction of the model being explained $\mathbb{E}[f]$. The goal is to fairly divide the payout $f(x) - \mathbb{E}[f]$ between the individual features of input x based on features' contributions. To simulate the situation where a subset of features is missing from the input, the values of the features that are supposed to be missing are replaced by randomly sampled values. Formally, for a permutation $\psi \in \Psi(N)$ and a sample $z \in \mathbb{R}^n$ drawn from a distribution $\mathcal{D}$ (distribution from which the training samples for the explanation are drawn) two instances $x_{+i}, x_{-i} \in \mathbb{R}^n$ are constructed as:

$$\begin{aligned} x_{+i} &= (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \dots, x_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \dots, z_{\psi^{-1}(n)}), \\ x_{-i} &= (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \dots, z_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \dots, z_{\psi^{-1}(n)}), \end{aligned} \tag{1.16}$$

Let $\phi_i(f,x)$ denote the Shapley value of feature i for the model f and input x. To compute the exact value of $\phi_i(f,x)$, the whole set of ordered permutations $\Psi(N)$ would have to be evaluated. Instead, only a fixed number of M randomly sampled permutations $\psi \in \Psi(N)$ and samples $z \in \mathcal{D}$ are evaluated to produce an approximation $\widehat{\phi_i}(f,x)$:

$$\widehat{\phi_i}(f,x) = \frac{1}{M} \sum_{\substack{m=1 \\ \psi_m \in \Psi(N) \\ z_m \in \mathcal{D}}}^{M} (f((x_{+i})_m) - f((x_{-i})_m)), \tag{1.17}$$

where $(x_{+i})_m$ and $(x_{-i})_m$ denote instances constructed as in Equation 1.16 based on permutation ψ_m and sample z_m.

$\widehat{\phi_i}(f,x)$ is an unbiased and consistent estimator of $\phi_i(f,x)$ and is approximately normally distributed: $\widehat{\phi_i}(f,x) \approx \mathcal{N}(\phi_i(f,x), \frac{\sigma_i^2}{M_i})$, where σ_i^2 denotes the variance of the feature i in the distribution $\mathcal{D}$ and M_i denotes the number of samples used for calculating $\widehat{\phi_i}(f,x)$. As $\widehat{\phi_i}(f,x) - \phi_i(f,x) \approx \mathcal{N}(0, \frac{\sigma_i^2}{M_i})$, the number of samples needed to accurately approximate $\phi_i(f,x)$ depends solely on the feature's i variance σ_i^2. Choosing the same number of samples for all the features is undesirable as the features are likely to have different variances. Therefore a minimal number of samples for each feature M_{min} and a maximal number of samples for all the features M_{max} are chosen. After each feature is sampled M_{min} times, the algorithm chooses which feature to sample next by choosing the feature j with the highest $(\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j+1})$, where m_j denotes the number of samples for the feature j drawn so far. The algorithm stop after drawing M_{max} samples. This sampling strategy minimizes the squared loss $\sum_{i=1}^{n}(\widehat{\phi_i}(f,x) - \phi_i(f,x))^2$. The same approach can be used even in situations where the variances σ_i^2 are unknown as they can be estimated from the samples $z \in \mathcal{D}$ drawn so far for example by Knuth's incremental algorithm [29].

Algorithm 6 SHAP

```
Require: Classifier $f$, Input $x$ of the prediction being explained
Require: Distribution $\mathcal{D}$ of the training data for explanation
Require: Minimal number of samples for each feature $M_{min}$
Require: Maximal number of samples for all the features $M_{max}$
 1: for $i = 1$ to $n$ do                                          ▷ Initialization
 2:     $m_i \leftarrow 0$
 3:     $\phi_i \leftarrow 0$                                          ▷ $\widehat{\phi_i}(f, x)$
 4: while $\sum_{i=1}^{n} m_i < M_{max}$ do
 5:     if $\exists i : m_i < M_{min}$ then
 6:         pick feauture $j$ to be sampled s.t. $m_j < M_{min}$
 7:     else
 8:         pick feature $j$ to be sampled with the highest $(\frac{\sigma_j^2}{m_j} - \frac{\sigma_j^2}{m_j+1})$
 9:     $\psi \leftarrow \Psi(N)$
10:     $z \leftarrow \mathcal{D}$
11:     $x_{+i} \leftarrow (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, x_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)})$
12:     $x_{-i} \leftarrow (x_{\psi^{-1}(1)}, x_{\psi^{-1}(2)}, \ldots, z_{\psi^{-1}(i)}, z_{\psi^{-1}(i+1)}, \ldots, z_{\psi^{-1}(n)})$
13:     $\phi_i \leftarrow \phi_i + (f(x_{+i}) - f(x_{-i}))$
14:     $m_j \leftarrow m_j + 1$
15: for $i = 1$ to $n$ do
16:     $\phi_i \leftarrow \frac{\phi_i}{m_i}$
17: return $\phi$                                          ▷ Vector of Shapley values
```

1.7.3.1 Kernel SHAP

Kernel SHAP is a method for approximating Shapley values introduced by Lundberg et. al. in [30]. The authors argue that Kernel SHAP achieves similar accuracy while being computationally more efficient compared to the Algorithm 6. The goal of Kernel SHAP is to learn an explanation model g:

$$g(z') = \phi_0 + \sum_{i=1}^{n} \phi_i z'_i, \tag{1.18}$$

where $z' \in \{0, 1\}^n$ denotes a coalition vector for an input of n features where the value of z'_i specify whether the feature i is present ($z'_i = 1$) or absent ($z'_i = 0$) [6]. The Shapley value ϕ_0 denotes the expected prediction of the model being explained $\mathbb{E}[f]$. For the input x of the prediction being explained, the coalition vector is defined as a vector of all ones: $x' = (1, 1, \ldots, 1)$. The explanation model g must satisfy the following:

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{n} \phi_i x'_i \tag{1.19}$$

This equation resembles the linear explanation model and Shapley values are the only solution to this equation that satisfies the axioms in Equations 1.11-1.14 as proved by Lloyd Shapley in [25]. Lundberg et. al. have proved, that the LIME framework described in subsection 1.7.1.1 is able to recover the Shapley values with the following choices of loss function $\mathcal{L}$, weighting kernel $\pi_{x'}$ and complexity metric Ω:

$$\Omega(g) = 0, \tag{1.20a}$$

$$\pi_{x'}(z') = \frac{(n-1)}{(n\ choose\ \|z'\|_0)\|z'\|_0(n-\|z'\|_0)}, \tag{1.20b}$$

$$\mathcal{L}(f, g, \pi_{x'}) = \sum_{z' \in \mathcal{B}} \pi_{x'}(z')(f(h_x(z')) - g(z'))^2, \tag{1.20c}$$

where where $\|z'\|_0$ denotes the ℓ_0 "norm" defined as the number of non-zero elements of vector z', $\mathcal{B}$ denotes a distribution of binary vectors and $h_x : \{0,1\}^n \mapsto \mathbb{R}^n$ is a function that maps a coalition vector to the original feature space in the following way:

$$(h_x(z'))_i = \begin{cases} x_i \in x & \text{if } z'_i = 1 \\ z_i \in z & \text{if } z'_i = 0 \end{cases}, \tag{1.21}$$

where $(h_x(z'))_i$ denotes the value of $h_x(z')$ on index i and z denotes a sample drawn from a distribution $\mathcal{D}$. The function h_x therefore replaces the feature values of x that are supposed to be absent with a value from a random sample z which is a similar approach as in the Algorithm 6.

In the LIME framework for linear models the complexity metric Ω is supposed to limit the number of features used by the explanation model to ensure interpretability. For calculating Shapley values the complexity metric Ω is ommited as the Shapley values must be calculated for all the input features to ensure that the axioms in Equations 1.11-1.14 are satisfied. The intuition behind the weighting kernel $\pi_{x'}$ is similar as in the weighting term in Equation 1.10, where the weights are assigned based on how many permutations are represended by the particular pick of present and absent players (features). $\pi_{x'}$ is therefore not a proximity measure that defines the locality around the input x as in the LIME framework. Finally, the linear explanation model whose weights correspond to Shapley values is obtained by minimizing the weighted square loss $\mathcal{L}$. The pseudocode of Kernel SHAP is given in Algorithm 7.

Algorithm 7 Kernel SHAP

```
Require: Classifier $f$, Input $x$ of the prediction being explained
Require: Distribution $\mathcal{D}$ of the training data for explanation
Require: Distribution $\mathcal{B}$ of binary vectors
Require: Number of training samples $M$
1: $\mathcal{Z} \leftarrow \emptyset$                                  ▷ Training data
2: for $i \in \{1, 2, \ldots, M\}$ do
3:     $z' \leftarrow \mathcal{B}$
4:     $z \leftarrow \mathcal{D}$
5:     $\mathcal{Z} \cup \langle z', f(h_x(z')), \pi_{x'}(z') \rangle$
6: $\phi \leftarrow$ LinearRegression($\mathcal{Z}$)
7: return $\phi$
```

It is important to note that the linear explanation model returned by Kernel SHAP is not a local surrogate explanation model as it is designed to approximate Shapley values and not to approximate the model being explained.

While Kernel SHAP was designed to be computationally more efficient than the Algorithm 6, Molnar et. al. in [6] argue, that Kernel SHAP is still very time consuming for some practical applications. Lundberg et. al. in [30] propose and review several model-specific methods for approximating Shapley values such as Linear SHAP for linear models, Tree SHAP for tree-based models and Deep SHAP for Artificial Neural Networks. These methods achieve higher computational efficiency as they are allowed to access model's internals such as gradient values. By being based on the same solid theory of Shapley values, these model-specific explanation methods can also be used as a complementary method for selection of the better performing model.

CHAPTER 2

Experimental part

2.1 Implementation

The experiments are written in the Python programming language (version 3.8) [31], which is a object-oriented, high-level language with a wide range of machine learning related packages and libraries available. The evaluated local model-agnostic explanation methods - LIME, Anchors and SHAP (KernelSHAP) are all implemented in Python by their respective authors [12, 19, 30]. Some of the evaluated methods are also implemented in explainability libraries such as Alibi [32] or XAI360 [16]. These libraries provide a unified interface for all their implemented explanation methods, however, to the best of our knowledge no such library currently includes all of the evaluated methods.

Artificial datasets used in section 2.2 were generated using the numpy library [33], which is a library for handling multidimensional arrays. Other libraries used in the experiments are – pandas [34] for handling tabular data and scikit-learn [35] that implements all the machine learning algorithms used in the experiments.

The experiments were run and are available in Jupyter notebooks [36] which is a web-based environment for running Python code.

2.2 Faithfulness evaluation on artificial datasets

In this section, the local model-agnostic explanation methods described in Section 1.7 are experimentally evaluated on artificial datasets in terms of their faithfulness . Faithfulness refers to the accuracy of an explanation with respect to the model being explained. Artificial datasets are generated based on known predefined dependencies, which allows for the optimal explanations to be calculated. Explanations of machine learning models that are highly accurate on a given artificial dataset can then be compared to the optimal explanation in order to evaluate explanation's faithfulness.

2.2.1 Artificial datasets

The artificial datasets described below are inspired by Robnik-Šikonja et. al. who use artificial datasets to evaluate their local model-agnostic explanation method in [37]. Each artificial dataset is associated with a suitable family of machine learning models.

condInd The condInd dataset consists of eight binary features and binary class with 50% probability of 1. Four of the features are important as they have equal value to the class in 90, 80, 70 and 60% of the cases, respectively. The other four features are unimportant as they are unrelated to the class (have equal value to the class in 50% cases). Since the features are conditionally independent, Naive Bayes classifier is suitable for this dataset.

xor The xor dataset consists of six binary features and binary class. Three of the features are important and – class is equal to 1 when an odd number of important features are equal to 1, and 0 otherwise. The other three features are unimportant as they are unrelated to the class. Decision Tree is suitable for this dataset as it can capture the necessary conditions.

cross The cross dataset consists of four continuous features and binary class. Two of the features are important and they are generated into a cross sign as shown in Figure 2.1. The class value is 1 if $(I_1 - 0.5)(I_2 - 0.5) > 0$, where I_1 and I_2 denote the two important features. The other two features are unimportant as they are unrelated to the class. Random Forest is the chosen model with the best performance on this dataset.

groups The groups dataset consists of four continuous features and binary class. Two of the features are important. The datapoints in 2-dimensional space of these important features form 16 clusters, where adjacent clusters have different class values as shown in Figure 2.2. The other two features are unimportant as they are unrelated to the class. Since each group is clustered together, k-Nearest Neighbors algorithm is suitable for this dataset.

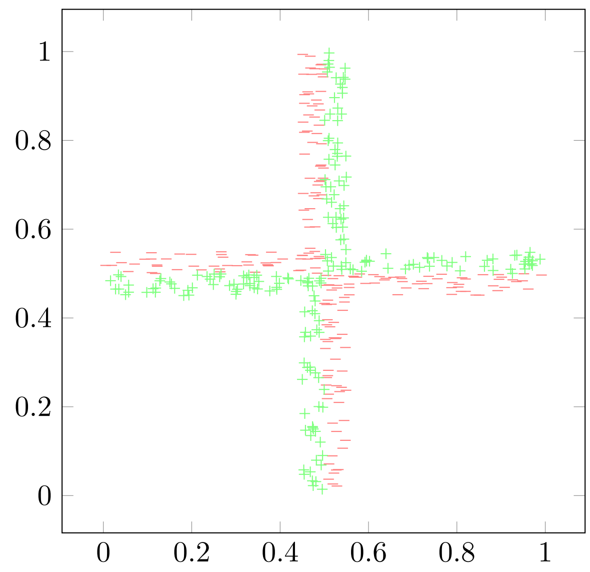

Figure 2.1: Visualization of the two important features in the cross dataset. Red minuses represent data points of class 0 and green pluses represent data points of class 1.

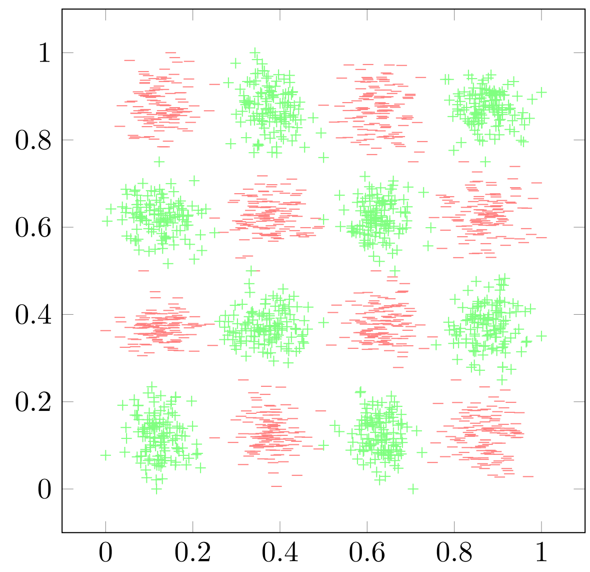

Figure 2.2: Visualization of the two important features in the groups dataset. Red minuses represent data points of class 0 and green pluses represent data points of class 1.

2.2.2 Metrics

Metrics that are used for the evaluation of faithfulness on artificial datasets are described below. Note that since each of the evaluated explanation methods produce different type of explanation not all of the metrics are applicable on every evaluated explanation method.

Recall on important features and False discovery rate on unimportant features Each artificial dataset is designed such that it contains equal number of important and unimportant features. This allows the explanation methods to be evaluated based on the ratio of important and unimportant features included in their explanations. For this evaluation, LIME's parameters are set such that its explanations include number of features equal to the number of important features in a given artificial dataset. Anchors and SHAP methods does not allow to set the number of features in their explanations. Anchors produces sparse explanations in order to maximize coverage of the decision rules. Features used in the decision rule explanations are considered important by the Anchors method. SHAP, however, always includes all of the features in its explanations, therefore a Shapley value threshold above which a feature is considered to be included in the explanation was experimentally set to 0.05. Higher threshold value leads to higher recall on important features and higher false discovery rate on unimportant features and vice versa.

Mean absolute error The true feature importance scores are known for all the artificial datasets. The correct feature importance scores in absolute values are the following: 0.4, 0.3, 0.2 and 0.1 for important features in the condInt dataset, 1/3 for each important feature in the xor dataset and 1/2 for each important feature in both cross and groups datasets. The unimportant features should be assigned zero feature importance scores in all the artificial datasets. Both LIME explanations' feature contributions and SHAP explanations' Shapley values can therefore be evaluated based on mean absolute error:

$$\frac{\Sigma_{i=1}^{n}|c_i - w_i|}{n}, \tag{2.1}$$

where c_i denotes the correct feature importance of feature i and w_i denotes the explanation's feature importance of feature i. Note that the correct feature importance scores have their sums normalized to 1, therefore the explanations' feature importance scores sums have to be also normalized to 1 (excluding the intercept). For this evaluation, LIME's parameters are set such that its explanations include all the features. SHAP explanations' always include all the features and Anchors explanations' can not be evaluated by this metric.

Accuracy and Coverage LIME and Anchors methods produce local surrogate models as explanations. Local surrogate models can be evaluated based on accuracy of their approximation of the model being explained. Accuracy of a local surrogate model g and a model being explained f is given by:

$$\frac{1}{|\Lambda_{test}|} \sum_{x \in \Lambda_{test}} \mathbb{1}_{g(x)=f(x)}, \tag{2.2}$$

where $\Lambda_{test} \subseteq \Pi_{test}$ denotes the set of testing data for the local surrogate model, Π_{test} denotes the set of all the testing data and $|\Lambda_{test}|$ denotes the number of elements in the set. Local surrogate model aims to approximate the model being explained locally, which means that the explanation does not apply to all possible inputs. Accuracy is therefore evaluated only on a subset of the testing data to which a given explanation apply. Proportion of the data to which the explanation apply is called coverage and is calculated as:

$$\frac{|\Lambda_{test}|}{|\Pi_{test}|}. \tag{2.3}$$

For Anchors, the subset Λ_{test} contains the testing instances that satisfy explanation's rule conditions. For LIME, the subset Λ_{test} contains the testing instances for which the explanation produces confident predictions. The threshold for confident predictions was set differently for each explained model. SHAP's accuracy and coverage can not be evaluated as the method does not produce a local surrogate model as an explanation.

2.2.3 Results

Table 2.1 shows results of the faithfulness evaluation. 2000 samples were generated for each artificial dataset, with half of the samples used for training and other half for testing. The same sets of training and testing data were used for the models being explained and for the explanations. Explanations were generated for each sample in the test set and the values listed in the table are mean metric values of all the explanations. For each artificial dataset, the chosen family of machine learning models is specified and accuracy of the model being explained is given in parentheses. Models' hyperparameters were set manually based on knowledge of the artifical dataset and explanations' parameters were set to default.

On the condInd dataset, all the explanation methods produced faithful local explanations. Anchors achieved a seemingly low recall on the important features as its goal is to maximize explanation's coverage while producing an accurate explanation. To achieve this goal on the Naive Bayes classifier trained on the condInd dataset, Anchors does not have to use all of the important features in its explanations. For example, when the features that have the same value as class in 90 and 70% of the cases have equal value, the model predicts this value as the class regardless of the other features.

Explanation method		condInd NB (0.92)	xor DT (1.00)	cross RF (1.00)	groups KNN (0.94)
LIME	Imp Recall	0.99	0.81	0.67	0.54
	Unimp FDR	0.01	0.19	0.33	0.46
	MAE	0.053	0.105	0.213	0.240
	Accuracy	0.93	∅	∅	∅
	Coverage	0.55	0.0	0.0	0.0
Anchors	Imp Recall	0.62	1.0	1.0	0.99
	Unimp FDR	0.06	0.71	0.39	0.82
	MAE	/	/	/	/
	Accuracy	0.92	1.0	0.97	0.88
	Coverage	0.36	0.06	0.13	0.02
SHAP	Imp Recall	0.79	1.0	1.0	0.95
	Unimp FDR	0.0	0.0	0.05	0.27
	MAE	0.046	0.008	0.025	0.082
	Accuracy	/	/	/	/
	Coverage	/	/	/	/

Table 2.1: Results of faithfulness evaluation on artificial datasets

On the xor dataset, LIME failed to produce faithful local surrogate models, despite being able to identify the important features and assigning reasonably accurate weights to the features. LIME explanations failed to produce any confident predictions which resulted in 0 coverage. This behaviour was expected as linear models are unable to learn even the two variable version of the xor function since it is highly non-linear [1]. Anchors produced explanations with low coverage due to the inclusion of unimportant features. Anchors produces the explanation rules iteratively by adding one feature condition to the rule in each step. However in the xor dataset, all feature conditions have the same discriminative value unless two of the important features are already included in the Decision Rule, which causes the high false discovery rate of unimportant features. While the xor dataset proved to be difficult for both LIME and Anchors methods, SHAP explanations achieved nearly optimal faithfulness.

The cross dataset is similar to the xor dataset with only two important features, where the features are continuous instead of binary. This causes similar issues for both LIME and Anchors methods. Looking at the Figure 2.1, the optimal coverage of the Anchors explanations should be 0.25 as opposed to the achieved coverage of 0.13.

Similar issues for LIME and Anchors also emerged on the groups dataset, where even the SHAP method included some of the unimportant features in its explanations, unlike on the other artificial datasets.

2.2.4 Summary

In this section, local model-agnostic explanation methods – LIME, Anchors and SHAP were evaluated in terms of their faithfulness to the model being explained on four artificial datasets. The artificial datasets demonstrated situations where both LIME and Anchors produce suboptimal explanations, whereas SHAP was able to produce faithful explanations on all the datasets.

LIME fails to produce faithful local explanations of highly non-linear models. Anchors produces overly specific explanations with low coverage when individual features have low discriminative value. Out of the evaluated local model-agnostic explanation methods, SHAP was evaluated to be the most robust one.

2.3 Explaining a black-box classifier learned on real-world dataset

In this section, the local model-agnostic methods are used to explain a black--box classifier learned on the real-world UCI divorce dataset [38].

The divorce dataset contains 51 features, each corresponding to a certain statement about a participant and their partner (current or ex), such as “I enjoy traveling with my wife” or “My spouse and I have similar ideas about how marriage should be”. The feature values range from -2 to 2, where -2 means the participant strongly disagrees with the statement and 2 means the participant strongly agrees with the statement. The class is binary, where 0 corresponds to a divorced participant and 1 corresponds to a married participant. The dataset contains data on 170 participants (84 divorced and 86 married) and was split into 113 training instances and 57 testing instances.

Yöntem et. al. in [38] have shown that Random Forest models achieve the highest performance on this task. The Random Forest model consisting of 50 Decision Trees with maximum depth of 4 achieved 98.2% accuracy on the set of testing data. The hyperparameters were found using grid search and cross-validation.

Yöntem et. al. have also compared performance of many different divorce prediction tools, concluding that the Random Forest models learned on this dataset achieves the highest accuracy, while also not requiring personal meeting with a couple therapist. However, a notable downside of this automated approach is that potential users are not expected to be able to understand decision making of a model that consists of 50 Decision Trees and uses 51 different features. It is easy to imagine that a couple would be more interested in the negative influences in their marriage that they can work on rather than getting an accurate prediction that they will divorce.

LIME, Anchors and SHAP explanations of one selected instance are shown below. The instance being explained is correctly predicted as divorced by the Random Forest model (the predicted probability of class 0 is 92%).

Table 2.2 shows LIME explanation of an instance predicted as divorced by the model. The number of features used by the LIME explanation was set to 7.

Feature Statement	Value	Weight	Feature's contribution
My spouse and I have similar ideas about how marriage should be.	-1	0.090	-0.090
My spouse and I have similar ideas about how roles should be in marriage.	-2	0.028	-0.057
We share the same views about being happy in our life with my spouse.	-2	0.014	-0.028
I think that one day in the future, when I look back, I see that my spouse and I have been in harmony with each other.	-1	0.024	-0.024
I know my spouse's basic anxieties.	-2	0.008	-0.015
I enjoy traveling with my wife.	-2	0.007	-0.015
My spouse and I have similar values in trust.	-2	0.005	-0.011
Explanation model's intercept	0.436		
Explanation model's prediction	0.191 ⇒ 0		

Table 2.2: LIME explanation of the instance predicted as divorced. Linear model's prediction is a sum of intercept and features' contributions, where feature's contribution is a multiplication of feature's value and corresponding weight. The LIME surrogate model's prediction is lower than 0.5 therefore it predicts class 0 (divorced).

Table 2.3 shows Anchors explanation of the same instance predicted as divorced by the model. Anchors is able to produce a very sparse explanation of the model's prediction, using only 2 of the 51 features. Method's parameters were set to default.

Feature conditions	
Our dreams with my spouse are similar and harmonious.	<= 1
We share the same views about being happy in our life with my spouse.	<= -2
Explanation model's prediction	0
Explanation model's coverage	0.47

Table 2.3: Anchors explanation of the instance predicted as divorced. When all the feature conditions are satsified by an instance, the explanation predicts class 0 (divorced). The explanation rule is satisfied by 47% instances from the test data.

Table 2.4 shows the 7 highest Shapley values genereated by the SHAP method for the same instance predicted as divorced. The SHAP method assigned non--zero Shapley values to 19 features. However, when exhaustive explanations are not required, shorter explanations should be preferred as they are easier to understand.

Feature Statement	Value	Feature importance
My spouse and I have similar ideas about how marriage should be.	-1	0.08
My spouse and I have similar ideas about how roles should be in marriage.	-2	0.07
We're compatible with my spouse about what love should be.	-1	0.04
We share the same views about being happy in our life with my spouse.	-2	0.04
I think that one day in the future, when I look back, I see that my spouse and I have been in harmony with each other.	-1	0.03
I know my spouse's basic anxieties.	-2	0.03
My spouse and I have similar values in trust.	-2	0.03
If one of us apologizes when our discussion deteriorates, the discussion ends.	-1	0.02
Explained model's expected prediction (probability of class 0)	0.47	
Explained model's prediction (probability of class 0)	0.92	

Table 2.4: 7 highest Shapley values of the instance predicted as divorced generated by the SHAP method. The model's expected probability of predicting class 0 is 0.47, whereas for this instance the predicted probability of class 0 is 0.92. Shapley values explain the difference between the expected prediction and the prediction for a particular instance.

2.3.1 Faithfulness evaluation

To evaluate fathfulness of explanations of a black-box model trained on a real-world dataset, metrics such as Recall on important features or MAE from subsection 2.2.2 can not be used, as the correct explanations are usually unknown. LIME and Anchors produce local surrogate models as explanations, which can be evaluated in terms of accuracy and coverage even on a real- - world datasets. The only metric that could be used to evaluate faithfulness of SHAP explanations is described in subsection 1.5.1.2 and based on replacing feature values with no information values (called no-op values) and measuring the impact on the predictions of the model being explained. The only sensible no-op value in the divorce dataset is 0 as it represents a neutral answer. However, for the model being explained, features with 0 value have positive impact towards not divorced predictions. When a considered instance is predicted as not divorced, replacing its feature values with no-op value changes the model's predictions for only 3 features, while the SHAP method assigns non-zero Shapley values to 19 features. This metric was therefore deemed as unreliable for evaluating SHAP explanations' faithfulness with respect to the model trained on the divorce dataset.

Table 2.5 show results of accuracy and coverage evaluation of LIME and Anchors explanations. Explanations were generated for each sample in the test set and mean values of accuracy and coverage are listed in the table. Both explanation methods achieved high accuracy therefore their explanations are considered faithful to the model being explained. LIME explanations achieved higher coverage, thus they explain a bigger part of the model, while Anchors explanations achieved better comprehensibility by using only 2 feature conditions in each explanation.

Explanation method		
LIME	Accuracy	1.0
	Coverage	0.63
Anchors	Accuracy	0.98
	Coverage	0.34

Table 2.5: Results of faithfulness evaluation of LIME and Anchors explanations on real-world divorce dataset

2.4 Discussion

The local model-agnostic explanation methods - LIME, Anchors and SHAP were used to explain and evaluated in terms of faithfulness on machine learning models learned on four artificial datasets and one real-world divorce dataset.

SHAP method produced explanations with the highest faithfulness on the artificial datasets. LIME was shown to struggle on highly non-linear models, while Anchors produced overly specific explanations with low coverage in cases where individual feature conditions had low discriminative value. The experiments suggest that when Anchors produce overly specific explanations, it may indicate that such explanations are not faithful to the model being explained.

Experiments on the real-world divorce dataset demonstrated the main disadvantage of the SHAP method, which is that to the best of our knowledge, there is no reliable metric to evaluate its explanations' faithfulness on real--world datasets. LIME and Anchors produce explanations in form of a local surrogate models which can always be evaluated in terms of accuracy on the set of testing data annotated by the model being explained. Both LIME and Anchors explanations of the Random Forest classifier trained on the divorce dataset achieved high accuracy. LIME managed to explain a bigger part of the model, while Anchors achieved better comprehensibility by using only two feature conditions in its explanations.

The experiments showed that none of the considered local model-agnostic methods can be deemed as superior overall. Anchors may be more appropriate in situations where high comprehensibility is prioritized by the audience. LIME, on the other hand, allows to specifically set the number of features used in its explanations, thus tuning both comprehensibility and faithfulness of its explanations. SHAP can be used when both LIME and Anchors fail to produce faithful explanations as it was found to be the most robust explanation method that is also based on solid theory of Shapley values. However, SHAP produces only approximation of the true Shapley values, therefore a lack of reliable evaluation metric may limit the audiences' trust to its explanations.

Chapter **3**

Conclusion

This work focuses on part of the Explainable Artificial Inteligence field concerned with local model-agnostic explanation methods that aim to explain a prediction of a given machine learning classifier by studying the model only through its inputs and corresponding outputs.

Apart from the explanation methods, theoretical part describes basic concepts in Explainable AI – motivation, definitions, taxonomy of explanation methods, various properties of explanations and evaluation metrics. Three local model-agnostic explanation methods are described in the theoretical part – LIME that produces explanations in form of linear local surrogate models, Anchors that utilizes decision rules as local surrogate models, and SHAP that calculates feature importance scores via Shapley values from coalitional game theory.

Experimental part evaluates the described local model-agnostic explanation methods in terms of faithfulness of their explanations to the model being explained. Explanations are evaluated on various classifiers trained on artificially generated datasets as well as real-world divorce dataset. Experiment with artificial datasets demonstrated that LIME fails to produce faithful explanations of highly non-linear models, while Anchors fail to produce faithful explanations in situations where individual feature conditions have low discriminative value. SHAP was found to be the most robust out of the considered explanation methods as it produced faithful explanations of each of the classifiers trained on artificial datasets. Experiment on real-world divorce dataset demonstrated the main drawback of SHAP as to the best of our knowledge there is currently no reliable metric to evaluate its explanations' faithfulness on real-world tasks where the optimal explanations are not known. Both LIME and Anchors produced faithful explanations of the model trained on divorce dataset where LIME explained a bigger part of the model, while Anchors achieved better comprehensibility of its explanations.

The implication of the experiments is that none of the considered local model-

-agnostic explanation methods can be deemed as superior overall. The optimal explanation method choice depends on the particular model being explained and goals of its audience. Even though LIME and Anchors may fail to produce faithful explanations in some situations, faithfulness of their explanations can always be evaluated. Both LIME and Anchors methods are suitable in situations when sparse explanations are preferred. SHAP is suitable in situations when both LIME and Anchors fail to produce accurate explanations as SHAP was evaluated as the most robust out of the considered explanation methods.

While the field of Explainable Artificial Intelligence is relatively new, many practically useful methods for explaining predictions of black-box classifiers have already been developed. The priority of the future research should be put in developing reliable faithfulness evaluation metrics that would allow to evaluate and compare faithfulness of various explanation types. The LIME framework could also be extended with options to use different faimilies of local surrogate models such as decision trees to address the limitations of linear models in certain situations.

Bibliography

[1] Goodfellow, I.; Bengio, Y.; et al. *Deep Learning*. MIT Press, 2016, ISBN 9780262337373, `http://www.deeplearningbook.org`.

[2] Hastie, T.; Tibshirani, R.; et al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer series in statistics, Springer, 2009, ISBN 9780387848846. Available from: `https://books.google.cz/books?id=eBSgoAEACAAJ`

[3] Arrieta, A. B.; Díaz-Rodríguez, N.; et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, volume 58, 2020: pp. 82–115, ISSN 1566-2535.

[4] Ribeiro, M. T.; Singh, S.; et al. Model-agnostic interpretability of machine learning. *ArXiv*, volume abs/1606.05386, 2016, ISSN 2331-8422.

[5] Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, volume 267, 2019: pp. 1–38, ISSN 0004-3702.

[6] Molnar, C. *Interpretable Machine Learning*. Lulu, 2019, ISBN 9780244768522, `https://christophm.github.io/interpretable-ml-book/`.

[7] Lisman, J. E.; Idiart, M. A. Storage of 7+/-2 short-term memories in oscillatory subcycles. *Science*, volume 267, no. 5203, 1995: pp. 1512–1515, ISSN 1095-9203.

[8] Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, volume 1, no. 5, 2019: pp. 206–215, ISSN 2522-5839.

[9] Choudhary, P. Interpreting predictive models with Skater: Unboxing model opacity. 2018, [Online]. Available from: `https://www.oreilly.com/content/interpreting-predictive-models-with-skater-unboxing-model-opacity/`

[10] Wu, R.; Yan, S.; et al. Deep image: Scaling up image recognition. *arXiv preprint arXiv:1501.02876*, volume 7, no. 8, 2015, ISSN 2331-8422.

[11] Guidotti, R.; Monreale, A.; et al. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, volume 51, no. 5, 2018: pp. 1–42, ISSN 1557-7341.

[12] Ribeiro, M. T.; Singh, S.; et al. ”Why should i trust you?” Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, ISBN 9781450342322, pp. 1135–1144.

[13] Council of the European Union. General Data Protection Regulation (EU) 2016/679. 2016.

[14] Office of the Federal Register. Code of Federal Regulations, Title 12 Banks and Banking. 1972.

[15] Angwin, L. J. M. S., Julia; Kirchner, L. Machine Bias. 2016, [Online]. Available from: `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`

[16] Arya, V.; Bellamy, R. K.; et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019, ISSN 2331-8422.

[17] Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017, ISSN 2331-8422.

[18] Dhurandhar, A.; Iyengar, V.; et al. Tip: Typifying the interpretability of procedures. *arXiv preprint arXiv:1706.02952*, 2017, ISSN 2331-8422.

[19] Ribeiro, M. T.; Singh, S.; et al. Anchors: High-precision model-agnostic explanations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, ISSN 2374-3468.

[20] Su, G.; Wei, D.; et al. Interpretable two-level boolean rule learning for classification. *arXiv preprint arXiv:1511.07361*, 2015, ISSN 2331-8422.

[21] Beam Search <algorithm>. [Online]. Available from: `http://foldoc.org/beam+search`

[22] Weber, R.; et al. On the Gittins index for multiarmed bandits. *The Annals of Applied Probability*, volume 2, no. 4, 1992: pp. 1024–1033, ISSN 2168-8737.

[23] Kalyanakrishnan, S.; Tewari, A.; et al. PAC Subset Selection in Stochastic Multi-armed Bandits. In *ICML*, volume 12, 2012, ISSN 2640-3498, pp. 655–662.

[24] Kaufmann, E.; Kalyanakrishnan, S. Information complexity in bandit subset selection. In *Conference on Learning Theory*, 2013, ISSN 1532-4435, pp. 228–251.

[25] Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, volume 2, no. 28, 1953: pp. 307–317, ISSN 0066-2313.

[26] Hall, P. Building Explainable Machine Learning Systems: The Good, the Bad, and the Ugly. 2018, [h2o meetup NYC 2018]. Available from: `https://www.youtube.com/watch?v=Q8rTrmqUQsU`

[27] Štrumbelj, E.; Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, volume 41, no. 3, 2014: pp. 647–665, ISSN 0219-1377.

[28] Kononenko, I.; et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, volume 11, no. Jan, 2010: pp. 1–18, ISSN 1532-4435.

[29] Knuth, D. E. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Boston: Addison-Wesley, third edition, 1997, ISBN 0201896842 9780201896848.

[30] Lundberg, S. M.; Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[31] Van Rossum, G.; Drake, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN 1441412697.

[32] Klaise, J.; Van Looveren, A.; et al. Alibi: Algorithms for monitoring and explaining machine learning models. Available from: `https://github.com/SeldonIO/alibi`

[33] Oliphant, T. E. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006, ISBN 9781517300074.

[34] Wes McKinney. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*, edited by Stéfan van der Walt; Jarrod Millman, 2010, ISSN 2575-9752, pp. 56 – 61, doi: 10.25080/Majora-92bf1922-00a.

[35] Pedregosa, F.; Varoquaux, G.; et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, volume 12, no. Oct, 2011: pp. 2825–2830, ISSN 1532-4435.

[36] Kluyver, T.; Ragan-Kelley, B.; et al. Jupyter Notebooks – a publishing format for reproducible computational workflows. 01 2016, ISBN 9781614996484, pp. 87–90.

[37] Robnik-Sikonja, M.; Kononenko, I. Explaining Classifications For Individual Instances. *IEEE Transactions on Knowledge and Data Engineering*, volume 20, 2008: pp. 589–600, ISSN 1041-4347.

[38] Yontem, M. K.; Adem, K.; et al. Divorce Prediction Using Correlation Based Feature Selection Aand Artificial Neural Networks. *Nevşehir Hacı Bektaş Veli Üniversitesi SBE Dergisi*, volume 9, 2019: pp. 259 – 273, ISSN 2149-3871.

Appendix A

Acronyms

AI Artificial Intelligence

COMPAS Correctional Offender Management Profiling for Alternative Sanctions

DT Decision Tree

FDR False discovery rate

KNN k-Nearest Neighbors algorithm

LIME Local Interpretable Model-Agnostic Explanations

MAE Mean absolute error

NB Naive Bayes

RF Random Forest

SHAP Shapley Additive Explanations

XAI Explainable Artificial Intelligence

Appendix B

Contents of enclosed CD

```
README.md ........................ the file with CD contents description
experiments ...................... the experiments files directory
  thesis_env.yml ................. the conda environment file
  dataset ........................ the directory with the divorce dataset
  notebooks ...................... the directory with the Jupyter notebooks
text ............................. the thesis text directory
  BachelorThesis_Skluzacek.pdf ... the thesis text in PDF format
  src ............................ the directory with the LaTeX source files
```