University of Zurich^UZH

MASTER'S THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Computing the Trustworthiness Level of Black Box Machine and Deep Learning Models

Dario Gagulic (18-707-257)

Supervisor: Dr. Alberto Huertas, Chao Feng
Date of Submission: August 7, 2023

University of Zurich, Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Master's Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

The field of Artificial Intelligence (AI) is rapidly evolving and increasingly being integrated into our everyday life. Black Box Machine and Deep Learning systems support humans in making important decisions in safety-critical industries, that consequently influence the lives of real people. This has raised the need for the ability to assess the model's trustworthiness. Trust is a subjective concept and depends on many factors. As Black Box models grow bigger and become more complex, it has become impossible, even for domain experts, to understand their reasoning and analyze how such models derive conclusions. Luckily, early work has developed automatic tools that allow the computation and evaluation of trust in a particular system, based on the pillars called *fairness, explainability, robustness*, and *methodology.* The algorithm computes various metrics and relies on the user to upload the model, the used dataset, and the FactSheet describing the applied training methodology. This forms a problem when computing the trustworthiness level of Black Box Machine and Deep Learning models with limited data access. Notably, the presented work identified two common definitions of the term *Black Box* established in the research community. The first focuses on complex systems with limited interpretability, and the underexplored second definition with respect to trustworthiness assessment describes systems with limited information available. Therefore, this master's thesis introduces a Black Box Taxonomy, categorizing Machine Learning models based on interpretability into different subgroups and adding another dimension distinguishing their available information levels. Further, a novel approach is proposed introducing a synthetic dataset generator to compute the trust score of Black Box models. The generator offers two approaches (MUST and MAY) to balance privacy and accuracy concerns. This solution addresses incomputable metrics, leading to a more accurate trustworthiness assessment. In order to validate the approach, the implementation was evaluated on two real-world scenarios.

Zusammenfassung

Der Bereich der Künstlichen Intelligenz (KI) entwickelt sich rasant und wird zunehmend in unseren Alltag integriert. Black Box Machine und Deep Learning Systeme unterstützen den Menschen bei wichtigen Entscheidungen in sicherheitskritischen Branchen, die das Leben von Menschen beeinflussen. Dadurch ist die Notwendigkeit entstanden, die Vertrauenswürdigkeit des Modells zu bewerten. Vertrauen ist ein subjektives Konzept und hängt von vielen Faktoren ab. Da Black Box Modelle immer grösser und komplexer werden, ist es selbst für Fachleute unmöglich geworden, die Herleitung zu verstehen und zu analysieren, wie solche Modelle Schlussfolgerungen ableiten. Glücklicherweise wurden in frühen Arbeiten Tools entwickelt, die die Berechnung und Bewertung von Vertrauen in ein bestimmtes System ermöglichen. Dies auf der Grundlage der Säulen Fairness, Erklärbarkeit, Robustheit und Methodik. Der Algorithmus berechnet verschiedene Metriken auf Basis des angewandten Modells, dem verwendeten Datensatz und dem FactSheet, das die angewandte Methodik zum Trainieren des Modells beschreibt. Dies stellt ein Problem dar bei der Berechnung der Vertrauenswürdigkeit von Black Box Machine und Deep Learning Modellen mit begrenztem Datenzugang. In der vorliegenden Arbeit wurden zwei gängige Definitionen des Begriffs *Black Box* in der Forschungsgemeinschaft identifiziert. Die erste konzentriert sich auf komplexe Systeme mit eingeschränkter Interpretierbarkeit, die zweite Definition beschreibt Systeme mit eingeschränkter Informationsverfügbarkeit, die noch zu wenig erforscht sind im Hinblick auf die Bewertung der Vertrauenswürdigkeit. Daher wird in dieser Masterarbeit eine Black-Box-Taxonomie eingeführt, die ML-Modelle anhand ihrer Interpretierbarkeit in verschiedene Untergruppen kategorisiert und eine weitere Dimension hinzufügt, die zwischen dem Level an verfügbaren Informationen unterscheidet. Darüber hinaus wird ein neuartiger Ansatz vorgeschlagen, der einen synthetischen Datensatzgenerator verwendet, um die Vertrauenswürdigkeit von Black-Box-Modellen zu berechnen. Der Generator bietet zwei Ansätze (MUST und MAY), um eine Balance zwischen Datenschutz und Genauigkeit herzustellen. Diese Lösung bietet eine Möglichkeit, um nicht berechenbare Metriken zu quantifizieren, was zu einer genaueren Bewertung der Vertrauenswürdigkeit führt. Um den Ansatz zu validieren, wurde die Implementierung anhand von zwei realen Szenarien evaluiert.

Acknowledgments

First of all, I want to thank Dr. Alberto Huertas for his professional and supportive supervision. I really appreciated his insights and enjoyed working alongside him. Also, big thanks to Prof. Dr. Burkhard Stiller and the Communication Systems Group for the opportunity to work on this interesting and relevant research topic.

Additionally, I want to express my sincere gratitude to my family and close friends for always believing in me and giving me the strength and love I needed. I am proud to finish my studies in such a promising field and looking forward to an interesting future.

Contents

Chapter 1

Introduction

1.1 Motivation

Digitalization has a greater impact on today's world than ever before, influencing various aspects of our life. Especially the field of Artificial Intelligence (AI) has experienced a rapid and remarkable evolution in the last decade and continues to influence our everyday life. Since 2016 for instance, two self-driving minibuses with passenger service can be found on the roads of Sion, and from 2019 also in Switzerland's capital city Bern [1]. TikTok's For You Page (FYP) uses advanced AI-based recommender engines to personalize the user's content feed based on interests and engagement without the need to actually follow a selected group of people [2]. Generative AI techniques create digital content, such as realistic music, visual art, creative advertisement spots, or even code [3], [4]. ChatGPT is a natural language processing (NLP) system capable of generating human-like conversations and stands out as one of the most promising AI technologies [5]. Because of its promising capabilities, AI has become increasingly used to help humans make better decisions also in safety-critical industries, such as personalized medicine, employment decisions, financial risk analysis, and legal assessment. However, this generates a form of dependency and requires the user to trust the predictions of the system.

Current solutions focus on achieving the best performance and are increasingly growing in size and complexity [6]. These systems are widespread but often difficult to interpret because they use complex Machine Learning (ML) models to make predictions, like for example Deep Neural Networks (DNNs) with millions of parameters. They are considered Gray Boxes, as they take data for input, and generate outcomes without the need for a clear understanding of the internal processes. Considering Black Box approaches, important details regarding the data or the algorithms and hyperparameters used to train the model are completely missing. Relying on opaque decision-making systems without the ability to understand their reasoning can be very problematic. It may lead to discrimination and cause trust issues because people may not feel comfortable putting their faith in something they do not fully comprehend [7].

Real-world situations have shown that systems may introduce historical human prejudices or inherit biases when, for instance, being trained on an unrepresentative dataset. For example, facial recognition algorithms have raised concerns about racial discrimination [8] and AI recruiting tools showed systematic bias against women [9]. Further, vulnerable systems that lead to unintended behavior are simple targets for various forms of adversarial attacks. Using a small piece of tape, researchers have successfully fooled Tesla's cameras, causing the car to accelerate by 50 miles per hour due to a mistake in perception [10]. This example demonstrates the risk of wrong decisions taken by such systems, as they can lead to unintended accidents and potentially cost people's lives. Therefore, the goal of AI developers is to enhance the system's ability to handle unforeseen circumstances, maintain consistent performance, and minimize the risk of exploitation.

These and similar experiences from the past have created a general agreement on the urgent need for guidelines and principles that are capable of creating safer and more trustable systems. Overall, different design principles and technical measures have been introduced to ensure AI systems align with societal values [6]. The literature has identified that trustworthy AI is a versatile concept and that different pillars like explainability, robustness, fairness, and methodology contribute to a safe and trustworthy deployment of AI systems [11]. Early work has developed an algorithmic framework to measure the trustworthiness of a trained ML model and compare different solutions with respect to the previously mentioned pillars [12]. Besides academia also innovative examples from the industry have shown useful progress toward trustworthy AI.

However, most scientific work and existing solutions focus on measuring and improving the trustworthiness of ML models where all information is available. They face a major limitation when being presented with a Black Box model, for which the used dataset is not accessible (f.i. because of privacy concerns). In order to improve existing solutions and make them applicable to the Black Box scenario, a new approach must be explored. In this context, the presented thesis reviews the existing pillars and metrics of trustworthy AI, analyzes concepts and definitions of Black Box approaches, and categorizes them in a Black Box Taxonomy. From there, a novel way of quantifying the Black Box model's trustworthiness is proposed under aggravated conditions. The primary objective is to examine the design and implementation of a synthetic dataset generator and evaluate it on real-world scenarios to uncover an improved way of assessing the trustworthiness of ML models with limited information.

1.2 Description of Work

The main goal of the presented work is to come up with a methodology and implement an algorithm that is able to measure and quantify the trustworthiness level of Black Box ML/DL models. Hence, this master's thesis consists of the following steps:

(i) Survey of Related Work

- Focus on existing pillars and metrics relevant for trusted Black Box ML/DL models

- Analyze existing solutions and understand their limitations considering the trustworthiness assessment of Black Box models
- Understand properties and characteristics of Black Box models
- Study existing approaches of opening the Black Box

(ii) Definition of Black Box Taxonomy

- Provide a formal description of existing and missing aspects
- Categorize existing models into subgroups based on their properties
- Define a comprehensive taxonomy respecting various definitions of the term Black Box

(iii) Design an algorithm suitable for quantifying the trustworthiness of Black Box

- Explore novel approaches and specify process design
- Explain the goal and benefit of a synthetic dataset generator
- Specify the different execution scenarios (MUST and MAY) and explain their (dis)advantages
- Describe the algorithmic design
- Elaborate the challenges and limitations

(iv) Implementation of the specified algorithm

- Realize the previously mentioned algorithm design
- Provide a possible integration into the Trusted AI Platform 2.0
- Provide installation guidelines

(v) Evaluation and Discussion

- Define feasible and realistic evaluation scenarios
- Find and describe appropriate datasets and train models
- Compute trust scores and test the implementation
- Compare and discuss the generated dataset and the computed trust scores

1.3 Thesis Outline

The following Chapters are structured as follows. Chapter 2 describes the *Background* in context of Black Box ML and Trustworthy AI. Chapter 3 discusses *Related Work* and existing tools in this area, and highlights their strengths as well as their limitations and missing aspects. Chapter 4 proposes a Black Box Taxonomy and describes the problem of incomputable metrics. Further, it presents the *Design and Implementation* of the synthetic dataset generator and explains the possible specifications together with their requirements and advantages/disadvantages. The experimental results are described and discussed in Chapter 5. Chapter 6 ends the paper with a *Summary and Conclusion* on existing problems and addresses further research directions.

Chapter 2

Background

2.1 Machine Learning

Machine learning (ML) is a subfield of Artificial Intelligence (AI) and describes the process of creating algorithms and models which are able to learn and make decisions based on data. Statistical methods are applied, to train and improve the model's accuracy over time. This is achieved through the optimization of a loss function, comparing the model's prediction and the actual output. The output can either be a binary classification (yes/no), a multi-classification for more than two different classes, or a number for regression problems [13]. There exist different models which have proven their efficiency for different use cases. In supervised ML, the model is trained on labeled data, where the correct output must be provided in the dataset. Differently, in unsupervised ML, the model is able to detect patterns and relationships between data points without being provided labeled data. For instance, clustering the data into distinct subgroups or anomaly detection are classical use cases of unsupervised ML [14]. Another area of ML is reinforcement learning, where the model is acting as an agent in an environment consisting of different states [15]. Depending on the model's current state, different actions are available to choose from. Based on a derived policy, the model executes the optimal action and receives feedback in the form of a new state and a set of new possible actions. For each action, the model is provided with a reward for making a good decision, or punishment for making a bad decision. Reinforcement learning is often applied to intelligent control robots [16] or tasks like drone navigation [17]. In the field of cybersecurity, new trends have shown that reinforcement learning can be used by offensive forces to dynamically adapt ransomware-specific behavior in order to remain undetected [18] as well as on the defensive side, to mitigate zero-day-attacks by dynamically altering target attack surfaces [19].

Deep Learning (DL) is a subset of ML, which shows vast performance improvements on more complex problems than classical ML algorithms, such as speech recognition and object detection. These systems consist of multiple processing layers with varying levels of abstraction and have the ability to discover systematic patterns in large datasets. From a mathematical perspective, they compute gradients and apply backpropagation during the training phase to determine how much the internal parameters should be adjusted to achieve a better accuracy [20]. In Deep Neural Networks (DNNs), multiple layers are stacked with numerous nodes and used to progressively extract higher-level features from raw input [21]. Other examples are Recurrent Neural Networks (RNNs), which are mainly used for the generation and analysis of sequential data such as text and speech, as well as Convolutional Neural Networks (CNNs), used to process audio, images, and videos. Real-world applications of Deep Reinforcement Learning (Deep RL) are complex tasks, such as intelligent game playing [22] or autonomous driving [23]. Finally, transformers apply the encoder-decoder architecture and are well-suited for translations or image compression [20].

2.2 Black Box vs. White Box

The term *Black Box* is describing a process or system in which the internal operations are unknown to an observer, due to its high complexity. Often, a level of abstraction is introduced through which complex systems can be simplified, making the big picture easier to understand. For instance, a driver of a vehicle does not have to understand how a multicylinder engine is capable of providing the desired power and how the transmission system uses the power to make the wheels of the car roll. Instead, the driver must understand how to use gas and brake pedals to let the system know the desired intentions. In this scenario, the car can be considered a Black Box for a non-technical person, since the internal logic is hidden from the user. Similarly, in the field of computer science, software programs and algorithms whose implementation is difficult to understand are considered a Black Box [24]. Hereby, the user is only required to provide valid input data to the AI model and retrieve/interpret the output without having to understand the intrinsic implementation [7]. More complex AI models like Deep Neural Networks (DNNs), Support Vector Machines (SVMs), or Markov Networks can provide valid and more accurate output to difficult real-world problems because of their ability to abstract well. However, even for highly skilled experts, the reasoning behind these algorithms' decision-making is hard to interpret. This is often a challenge when the developer of the model wants to inspect the source of potential incorrect outputs, since the mechanisms are hard to diagnose by definition. Further, a system or algorithm can also be considered a Black Box when the internal operations are simply not accessible. This alternative definition from the literature is well-known in the field of cybersecurity and software testing [25], [26]. Considering AI models, this definition applies when there is missing knowledge about the dataset used for training and testing the model or a lack of insights into the intrinsic parameters (weights, biases, hyperparameters, etc.) of the model itself. For a simpler distinction, in the presented work the former definition (describing complex systems) will be referred to as *Gray Box*, while the latter definition (describing systems with limited information) remains *Black Box*.

In contrast to Gray/Black Box systems, the term *White Box* describes systems where internal operations are transparent and interpretable. This means that the experts in the application domain can understand how the algorithm arrived at its predictions or decisions. The benefit of this approach is, that White Box models can be applied in scenarios, where transparency and explainability play a crucial role and can not be neglected.

Usually, in companies and processes where AI is used to support human decision-making, this aspect of transparency is particularly relevant for decision-makers. For instance, a doctor who uses ML models as a tool to assist a patient's diagnosis, treatment selection, or monitoring of disease requires a clear understanding of the influence that led to a concrete decision [27], [28]. Further, it is important that the models must be developed in a responsible and ethical manner. The same is true when applying ML in other scenarios, such as government agencies or financial institutions. The question arises, why not only transparent and explainable White Box models should be used in ML? Gray Box models prioritize accuracy over transparency and interpretability [29]. These models are often more complex and difficult to understand, but can achieve higher accuracy in identifying patterns and relationships in large and complex datasets like image processing. The short-term goal of making models transparent and easy to understand may not always align with the long-term goal of improving a certain industry. Scientists should be cautious about sacrificing accuracy for transparency and make sure that transparency is truly necessary, rather than just preference traditional methods.

Examples of Gray Box models are SVMs, DNNs, and models that contain a highly complex mathematical function or are difficult for experts to understand in practical settings or by the scientific community. White Box models are generally based on patterns, rules, or decision trees, and are more transparent and interpretable. They are easily understood by experts in practical applications because they provide a model closer to human language, enabling users to comprehend the reasoning behind the models' decisions [30].

2.3 Trustworthiness of ML/DL

With the increasing use of ML/DL in areas where models' decisions significantly affect people's lives, such as healthcare, law, and financial services, a need for trusting the predictions is emerging. But how can anyone be sure that the model's predictions are correct, and to what degree should people trust DL models? Incorrect results or systematic mistakes can lead to unintended consequences, such as bias and discrimination. For example, a model might mistake a stop sign for a speed limit sign, leading to dangerous driving situations. Further, DNNs can be deluded by attackers who apply adversarial samples in which they are adding minor perturbations to the original image such that the model is misclassifying the sample. Therefrom, the trustworthiness of ML may suffer. If people do not trust AI, they are unlikely to use it, which can limit the potential benefits of these systems. But is trust only affected by the belief that the model will perform well, or do other concepts influence human trust, like a detailed and fundamental comprehension of the model's implementation? The following Subsection 2.3.1 provides an answer to this question, by presenting existing pillars and metrics and explaining their main contributions to provide a comprehensive understanding of a model's trustworthiness. In the next Subsections, the aspect of *interpretability* is explained in more detail, which is an important influencing factor for trusting Gray Box solutions. Overall, the field of trustworthy AI aims to introduce mechanisms, such as design principles and technical measures, to make ML trustworthy and ensure that AI systems are aligned with societal values [29].

2.3.1 Existing Pillars and Metrics

The upcoming Subsections present and examine existing pillars established by the research community to evaluate the trustworthiness of ML models. For each pillar, the main contribution is explained, highlighting its significance through examples. The most important metrics are mentioned, which are used to assess the trustworthiness level of a model.

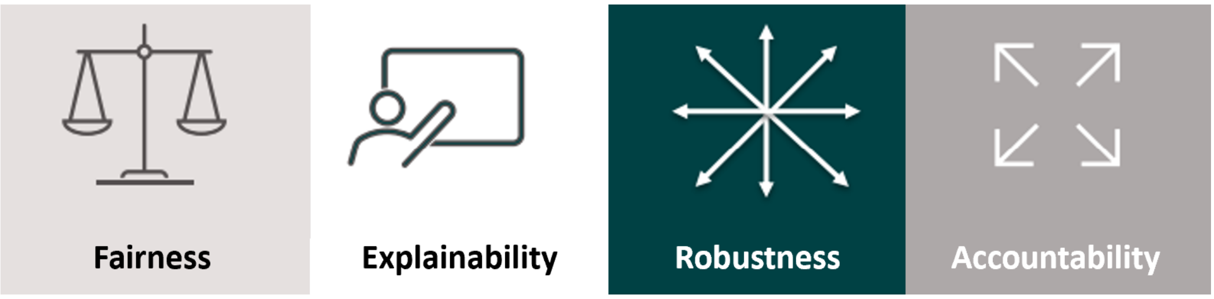


Figure 2.1: Exisitng Pillars Contributing To Trustworthy AI [6]

Fairness

The objective of the *fairness* pillar is to ensure that AI systems and their outcomes are unbiased and not prejudiced. It aims to prevent discriminatory practices, biases, or unfair treatment in decision-making processes [11]. Further, this pillar seeks to promote equal opportunities and address any inequalities or biases that may arise in AI systems. One example of deployed AI systems that did not respect the fairness principle is the case of early facial recognition systems deployed by HP and Google, which have demonstrated higher error rates and inaccuracies when identifying individuals with darker skin tones [8]. This bias in facial recognition algorithms has raised concerns about racial discrimination and unequal treatment in various domains, including law enforcement, surveillance, and hiring processes. Reasons for such biased outcomes often are related to an insufficient number of data points, an incomplete dataset, or a lacking diversity which makes the dataset not representative [11]. The decision-making process in AI is based on statistics and mathematical operations. Therefore, AI does not consciously distinguish between ethical and unethical choices. Humans bear the responsibility of addressing potential bias by considering the context of a scenario. For instance, if there are significant differences between men and women in terms of required treatment, it may be considered fair for a system to incorporate a patient's gender into individual treatment recommendations. In this example, using statistical parity as a measure to assess the model results for the protected class would be an unsuitable approach. Conversely, it would be considered unfair to employ an applicant's gender as a factor in hiring decisions. In this context, the application of this metric is justified. To ensure fairness, it is crucial to thoroughly examine the data and model for any hidden biases.

Fairness can be defined on both an individual and group level. It describes comparable handling for similar individuals or groups. This can be achieved by ensuring that the prediction probability, denoted as $f(X, P = 0) \approx f(X, P = 1)$, is independent of the protected feature for individual observations, where X is the set of observations $\{x1, x2, ..., x_N\}$, Y contains the corresponding labels $\{y1, y2, ..., y_N\}$, and $P = \{p1, p2, ..., p_N\} \in \{0, 1\}^N$ signals the protected group membership for every observation. Similarly, at the group level, fairness is achieved by grouping observations and ensuring similar treatment among the groups. This is accomplished by making the expected prediction value, denoted as $E[f(x, p = 0)] \approx E[f(x, p = 1)]$, approximately equal across both protected and unprotected groups [12]. Various approaches have been suggested in the literature to identify and mitigate bias such as the IBM AI Fairness 360 [31] open-source toolkit, which encourages the contribution of researchers to enhance existing metrics and algorithms. Developers are able to test their model on features like:

- *Statistical Parity Difference*, which computes the spread between the percentage of samples receiving a favorable outcome for protected and unprotected sample groups
- *Equal Opportunity Difference*, which measures the spread between true positive rate (TPR) and false positive rate (FPR) between different groups
- *Average Odds Difference*, which calculates the mean absolute difference in TPR and FPR between protected and unprotected groups
- *Disparate Impact*, which measures the ratio of a protected and unprotected group receiving a favorable prediction
- *Euclidean, Mahalanobis, and the Manhattan Distance*, which measure the proximity or distance between data points and determine if they can be clustered together

One can choose between ten algorithms to mitigate bias in the dataset (pre- and post-processing) or directly in the model itself (in-processing) [31]. Further, the difference in train and baseline performance can be valuable indicators describing *Underfitting* or *Overfitting*, as well as the *Class Balance*, which describes the ratio of samples belonging to different classes in the training dataset. If applied correctly, the fairness pillar helps to create unbiased models and generate equal opportunities.

Explainability

The aim of the *explainability* pillar is to provide transparency and understanding of how the AI system makes decisions. It seeks to enable humans to understand the reasoning behind AI algorithms and outputs, making the decision-making process more interpretable and accountable (see Subsection 2.3.2). This is particularly important when AI is used in high-risk use cases [11]. Despite the ability to support the criminal justice system in evaluating for instance criminal recidivism, ML models are today still not fully utilized at their potential. Reasons for this are, that even if humans are presented with evidence that machines are making on average better decisions than humans, it is natural to feel uneasy about it [32]. The explainability pillar improves trust by explaining the model's underlying reasoning, and therefore enables error detection and correction. Furthermore, it facilitates compliance with regulations because it allows regulators and stakeholders to understand and verify how certain decisions were made, ensuring that legal and ethical boundaries are respected. Simple ML models, such as linear regression or logistic regression, have a straightforward mathematical relationship between the input and output. This relationship is simple enough for humans to comprehend directly. It allows stakeholders to easily validate the algorithm's correctness. However, this is not true for more advanced models like Neural Networks (NNs) or Random Forest (RF). In these complex models, the relationship between the input and output can be extremely difficult to comprehend. Such complex Gray Box models are being applied in various fields and already have a major impact on human lives. Simply opting for explainable algorithms is insufficient, but the model itself must be designed to be explainable. Therefore, transparency is essential when it comes to trusting, understanding, and critiquing such models.

To build trust, it is necessary to comprehend the reasoning behind individual decisions made by these models. However, different personas have different requirements for explanations (e.g. person applying for a mortgage, an employee communicating the decision, or a developer implementing the tool for decision support). The AI Explainability 360 open-source toolkit [33] provides algorithms, which help users from various backgrounds and different requirements to understand the reasoning behind ML predictions. Tools like *Boolean Decision Rules via Column Generation, Generalized Linear Rule Models, ProfWeight, Teaching AI to Explain its Decisions, Contrastive Explanations Method (CEM), CEM with Monotonic Attribute Functions, Disentangled Inferred Prior VAE,* and *ProtoDash* can be applied by developers and end-users to add transparency throughout the whole AI lifecycle. Different metrics can be computed to calculate an explainability score describing the transparency and understanding of a particular AI model [11]. The *Model Size* calculates the number of parameters used by models, besides considering the used *Algorithm Class* itself. The *Correlated Features* metric computes the percentage of highly correlated features, which should be as low as possible. Correlated features affect the accuracy and influence the ability to generate explanations. Further, it complicates the interpretation of the *Feature Relevance* metric, which computes the percentage of irrelevant features for a set of predictions. Ideally, correlated features have been identified and dropped in the preprocessing phase.

Robustness

The *robustness* pillar aims to ensure that ML models are resilient and resistant to various forms of adversarial attacks, vulnerabilities, and unintended behaviors. The focus lies on creating models that are highly accurate and reliable, even when being faced with uncertainties, noisy data or adversarial inputs. ML models are created to engage with the real world, where it is impossible to control every input received. Deployed models are being exposed to data, that may reflect different properties than the data used for training and testing the model [34]. For instance, DNNs are known for their ability to abstract because of their architectural design, which allows them to solve complex problems [21]. On the contrary, it has been shown that small modifications to the inputs can change the output of the model even with high confidence. Such behavior indicates a lack of robustness and allows the model to be exploited through adversarial attacks [35]. The goal is to enhance the system's ability to maintain consistent performance, and minimize the risk of exploitation.

However, simply hiding the inner workings of a model is not enough to protect it from attackers. Therefore, different tools and methods have been proposed to measure and improve the robustness of ML models. One example is the Python library called *ART: Adversarial Robustness Toolbox* [36]. It is provided by IBM and enables developers to defend against different adversarial attacks, such as *evasion, data poisoning, extraction*, and *inference*. Similar to the previous pillars, different metrics can be computed describing the robustness of a particular AI model. The *Confidence Score* measures the probability of correctly predicting a given sample and describes how stable predictions are [37]. The *Loss Sensitivity* measures the extent to which the output of a model can change when there is a slight modification in the input [38]. If small changes in the inputs result in significant differences in the output, it indicates that the model lacks robustness. The *Cross Lipschitz Extreme Value for Network Robustness (CLEVER) Score* is developed for NNs and provides an estimation of the maximum amount by which the output can change in response to alterations in the input [39]. Further, the *Clique Method* computes the minimal adversarial perturbation needed for tree-based models [40]. Finally, the *Empirical Robustness (ER)* determines the average smallest change required in the input to alter the model's prediction [38]. It works by comparing the performance before and after a specific attack. It is important to note that certain metrics have limitations and can only be applied to specific ML algorithms [11]. Measuring and improving the robustness of a system makes it more reliable and increases the level of trust.

Accountability

The *accountability* pillar aims to evaluate if ML models are developed, deployed, and used in a responsible way. The term *methodology* is analogously found in literature and describes the same pillar. By documenting the creation process of AI models and validating its maintenance, this pillar generates traceability [11]. It verifies whether the decisions made align with established best practices. Having this information readily available enhances users' trust in the system. Similar to nutrition labels for food items, facts about the AI model allow the user of the system to better understand how the model was created. Further, it assigns clear responsibilities which hold organizations accountable for their actions, decisions, and impacts. This helps, for instance, to prevent models from being trained on prohibited datasets. However, since many roles are involved in the development of ML models, and since particular models require different details, it is difficult to come up with a generic method to document the development in a clear and understandable way. Piorkowski et al. [41] have evaluated different approaches to intelligent system documentation suitable for different personas. Based on their work, IBM has introduced AI FactSheets 360 [42]. It is a more general approach for capturing model facts from the entire AI lifecycle. The goal is to foster trust, integrity, and responsible behavior in the development and deployment of AI systems. For instance, appropriate data pre-processing steps decrease outlier effects and increase prediction stability. Having such information about taken measures present is valuable to the user of the system. Only relying on performance metrics can be misleading, and one might overlook flaws in the methodology and mistakenly place trust in the model. Valuable information for measuring the trustworthiness of the accountability pillar involves [11]:

- *Normalization*: Evaluates if the training data has been normalized or not.
- *Train/Test Split*: Measures the ratio between the number of samples used for training and testing.
- Information about how *Missing Data* was handled for the feature computation.
- Information if any *Regularization* techniques were applied.
- *FactSheet Completeness*: Measures if the FactSheet includes all necessary information that stakeholders need in order to trust the model and its predictions.

2.3.2 Interpretability and Transparency

In academic literature, the term *interpretability* is mentioned to be an important factor. However, often the term is referred to different concepts and only a few authors articulate precisely what interpretability means. Lipton [43] reasons, that the concept of interpretability lacks a clear and precise definition, and therefore statements about the interpretability of different models are not entirely scientific in nature. For instance, some papers refer to understandable models as transparent (knowing how the model works), and incomprehensible models as Gray Boxes. Other papers try to make the model more interpretable by applying post hoc explanations. These methods may clarify the predictions made by models, but do not reveal the underlying mechanisms by which the models operate. This implies, that a model is more interpretable if it is easy for a person to understand why the model made a certain prediction. According to Biran et al. [44], interpretability is the degree to which a human can understand the cause of the model's decision. There exists no mathematical definition of interpretability. In linguistics, however, *to interpret* means to bring out the meaning of something by performance or execution [45]. In the presented work, the term interpretability is defined as the ability to explain model-specific behavior based on intrinsic components, which only can be applied when information about the ML model is available and accessible, i.e. the model is transparent. To characterize how interpretable the model is (or has to be), different dimensions can be identified: First, a model may be globally or locally interpretable. Local interpretability means that the user is able to understand the mathematical reasoning behind a single prediction. Global interpretability means that the algorithmic properties themselves are interpretable to humans (e.g. most relevant features of the model) and an intuition about the model's global behavior is possible. Secondly, the amount of time the user needs to understand an explanation, which depends on the specific use case. And lastly, the nature of expertise and background knowledge of the user. Understanding these dimensions is crucial in determining the model's interpretability [7]. Applying post hoc methods to explain the model's decisions is referred to as explainability in this work. Hereby, an explanation must be an accurate representation of the Gray Box and understandable to humans (see Subsection 2.3.3).

The degree of transparency defines how much information about the model is known and accessible. A human should be able to produce a prediction for a transparent model whose input data and parameters are known in a reasonable amount of time. Therefore, the model's size and number of computations needed to make a prediction also impact the transparency. In other words, the complexity of the predictive model is a component for measuring transparency. Moreover, for a fully transparent model, each input should be understandable on its own, and models that contain complex or unidentifiable features may affect transparency and reduce interpretability. But not only information about the model's parameters is relevant. In addition, the information known about the learning of the algorithm itself is also defining the level of transparency [43].

Finally, one should also understand the reason *why* an explanation is necessary because different analytical methods may be required. Guidotti et al. [7] distinguish between the two scenarios: the need to reveal findings in data aimed at explaining why a specific decision has been returned for a particular input (applied nature), or the aim of explaining how the Gray Box itself works (theoretical nature). Also, in scenarios where no important decisions are made based on the prediction of the model, and no consequences exist for unacceptable results, it may not always be required to have an interpretable model.

2.3.3 Explainable AI (XAI)

Studies have investigated the significance of providing explanations to users in different domains, and the results consistently indicate that such explanations play a crucial role in increasing users' trust and confidence [46]. Trust is a subjective concept, where an individual may feel more comfortable with a model that they understand well, even if this understanding does not necessarily have a practical benefit. Explainable AI (XAI) is a collection of methods, technologies, and algorithms, which provide reasoning for the decision-making process through explanations. Furthermore, XAI highlights potential weaknesses of ML/DL models and is grouped into the explainability pillar. The goal is to enhance the interpretability of Gray Boxes by applying mechanisms that are of high quality, easy to understand, and interpretable by humans. Thereby, they give a sense of how the system will behave in the future. An important property of such systems is, that the explanations should be consistent for similar data points and remain stable over time. Das et al. [47] defines three categories for XAI:

1. **Scope**: Explanations can have a limited scope, focusing on specific data points or instances, as depicted in the right column of Figure 2.2. Alternatively, explanations can have a broader scope, describing the entire model and dataset, as the left column of Figure 2.2 shows. Certain explanation methods may be applicable to both *local* and *global* scopes.

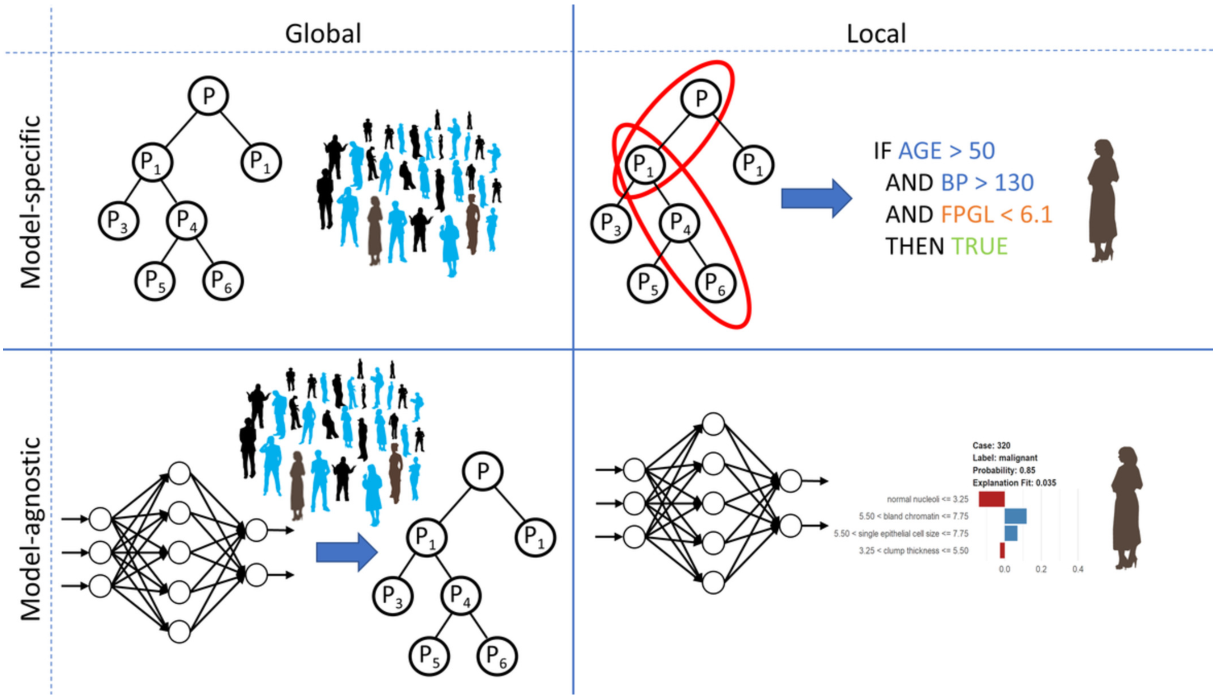


Figure 2.2: Visual Representation Of Scope And Usage Of Explainable AI [48]

2. **Methodology**: Explainable algorithms for both extent (local and global) may be either *backpropagation-based*, where a forward pass through a NN is performed and attributions during the backpropagation stage are generated by utilizing partial derivatives. A saliency mask is an example of such a method. *Perturbation-based* explainers involve changing the feature set of an input instance, either by masking certain features or by substituting them with other values using techniques like occlusion, filling operations, generative algorithms, conditional sampling, and more. These methods typically only require a forward pass through the NN to generate the attribution representations, without needing to backpropagate gradients.

3. **Usage**: The explanator can be either embedded within the model architecture (model-specific) as depicted in the top row of Figure 2.2, or applied as an external algorithm (model-agnostic), which is more flexible as it can be applied to any existing Gray Box model. Research is mostly focused on developing these model-agnostic explanations, which are able to provide explanations for every ML model used.

Chapter 3

Related Work

This Chapter reviews Related Work and gives insights into the research's state-of-the-art. It includes work attempting to open complex Gray Boxes and making them more transparent to increase trust. As described in Section 2.2, two common definitions exist for the term Black Box: The first well-known definition describes complex systems, which are even for domain experts hard to interpret. The second definition established in the research community is the one, where only limited information about the system or used dataset is available. Most related work has addressed the former definition, whereas the latter definition to my best knowledge has not yet been researched with respect to trustworthy AI so far, as the following Sections will show. For simpler distinction, in the presented work, the former definition (describing complex systems) is referred to as *Gray Box*. Further, in this Chapter an overview of existing tools used to explain complex ML models is given, and an existing solution is presented capable of evaluating the trustworthiness of any ML model. Finally, this Chapter concludes with a review of the strengths, challenges, and limitations.

3.1 Opening the Black Box

In recent years, the use of AI systems has become increasingly prevalent as they are becoming a part of everyday life. In November 2022 OpenAI introduced a chatbot called ChatGPT [49], which interacts with humans in a conversational way. This tool created awareness about how accessible it is to harness the power of AI for humans without a scientific background. As a consequence, a high interest in Trusted AI research and society evolved, particularly in the context of calculating the trustworthiness level of Gray Box models. If there is no technology that allows us to understand the reasoning behind Gray Box models, these models will continue to be viewed as mysterious and incomprehensible tools, much like oracles. However, attempts of *opening the Gray Box* are not new in research. Already in 1995, researchers have come up with an algorithm called TREPAN for extracting tree-structured representations of trained NNs [50]. Further, companies and communities have developed learning-based systems and tools that calculate reliable metrics, such as IMB 360 Trustworthy AI, LIME and SHAP (see Section 3.2). This emphasizes, that with the advancements in the field of ML and sophisticated concepts, an urge for explainability and a measure of trust is felt. On one side, Rudin [51, 52] highlights the importance of using interpretable models and emphasizes that explainable Gray Box models should be avoided in high-stake decisions. On the other side, Loyola-González [30] provides interesting perspectives to understand that both (White and Gray Box approaches) are suitable for addressing practical problems. Recently, researchers are developing approaches that make ML models more understandable to humans for specific scientific fields like genetics and genomics [53] or biology and medicine [28] as well as many others. Further, frameworks are being established that nurture trustworthiness when using ML in the field of psychiatry [54].

The increasing number of concepts and emerging solutions among different scientific communities can lead to confusion and difficulty in selecting the appropriate methods for explaining Gray Box models. While some focus on understanding how Gray Box models work, others are more interested in explaining the decisions made by these models, even if they do not fully understand their inner workings. Therefore, Giudotti et al. [7] provided a systematic organization and classification of these methodologies concerning the explanation of Gray Box models. This classification allows a simpler comparison of different solutions and an evaluation of the proposed solution. For the categorization, Guidotti et al. [7] differentiate solutions on the following four aspects for opening and understanding the Gray Box:

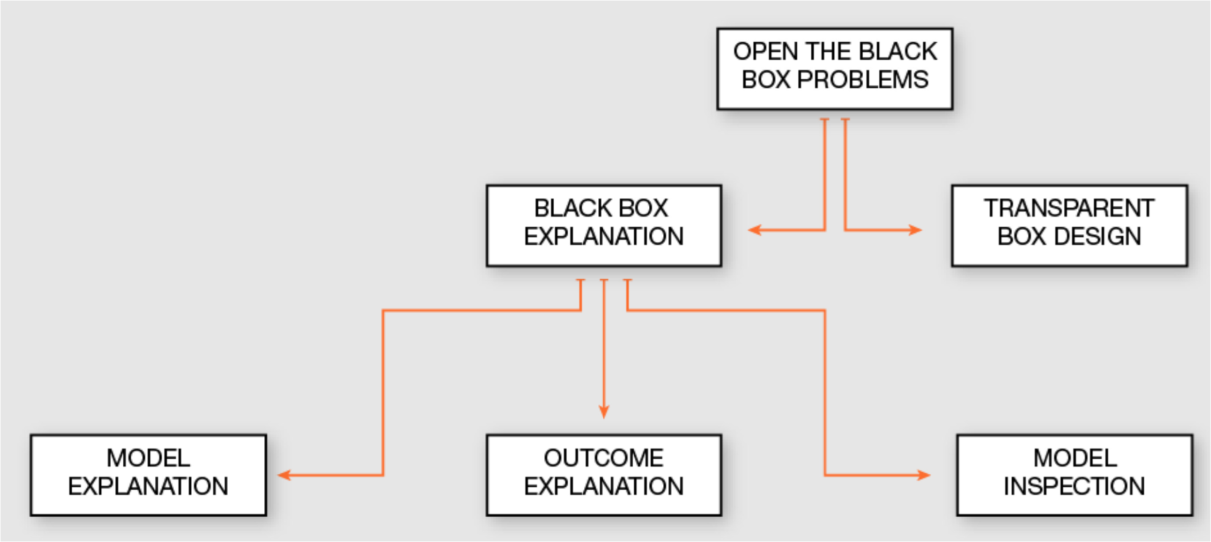


Figure 3.1: Problem Taxonomy Of Opening The Gray Box [7]

1. **Faced problem**: This aspect very much depends on the aim of the expert and the specific use case. Figure 3.1 depicts a tree-structured diagram, in which different categories of possible problems are listed. The first distinction occurs whether the aim is to directly design a transparent and interpretable classifier (*Transparent Box Design*) or to explain an existing predictor (*Gray Box Explanation*). The former approach aims to create a local or global interpretable model that maintains the accuracy of a Gray Box model designed to solve the same task. The latter is also

called *Reverse Engineering* or *Post Hoc* because this approach reconstructs an explanation for certain returned outcomes. In other words, Gray Box Explanation problems share the common target of providing an interpretable and accurate predictive model which mimics the original Gray Box model, but whose output can be comprehensibly understood by a human. The Gray Box Explanation can be further refined into *Model Explanation, Outcome Explanation,* and *Model Inspection.* Model Explanation seeks to comprehend the underlying logic behind the model, while Outcome Explanation is concerned with the relationship between the input data and the resulting decisions made by the model only for a specific record. Model Inspection lies somewhere in between and varies depending on the purpose of the research being conducted. For all problems of Gray Box Explanation, the dataset used for training the original Gray Box predictor is unknown.

2. **Type of analyzed data**: The type of data used for classification can vary, and offer different levels of interpretability for humans. Most ML models use *tables* (data in a structured format), which can be easily processed by algorithms in the form of matrices. For humans to interpret values in tables, additional meta-data needs to be provided. Differently, *images* and *texts* are more easily understood by humans since they reflect the way how people communicate. However, they must be transformed into vectors before they can be used for predictive models (usually SVM, NN, or DNN). Thus, there are some interpretable models that are not directly applicable to this type of data, which makes it difficult to obtain a human-understandable explanation or an interpretable model without using transformations. Other types of data include *sequences*, *spatio-temporal data*, and *complex networks*.

3. **Type of explanator**: The explanator is the tool or concept adopted to open the Gray Box and to provide an explanation in a way that is understandable and transparent to human users. Therefore, the explanator must be a simpler concept than the Gray Box itself. Guidotti et al. [7] differentiate between the following explanators:

   - *Decision Tree (DT)* also called *single-tree approximation*, is a popular technique and easily understandable.
   - *Decision Rules (DR)* describe a set of rules to explain the model, outcome, and design of the Gray Box.
   - *Feature Importance (FI)* is an effective solution and indicates the magnitude and weight of the features of the Gray Box.
   - *Saliency Mask (SM)* is a method capable of indicating causes of a certain outcome, often used to explain DNNs. It is especially useful for text or image input where a visual mask is highlighting specific aspects.
   - *Sensitivity Analysis (SA)* measures the uncertainty in the outcome regarding different uncertainties in the input.
   - *Partial Dependence Plot (PDP)* evaluates the relationship of a reduced feature space to the outcome of a Gray Box model.
   - *Prototype Selection (PS)* which selects representative instances from the available dataset to create an interpretable model.

- *Activation Maximization (AM)* is often applied to inspect a NN and DNN by identifying the essential neurons that are activated in response to specific input patterns, which means finding input patterns that can maximize the activation of a particular neuron in a specific layer.

4. **Type of predictor**: This aspect is a list of all Gray Box models which have been opened in reviewed papers listed by Guidotti et al. [7]. The list includes Tree Ensemble (TE), Support Vector Machines (SVMs), Neural Networks (NN), Deep Neural Networks (DNNs), and Non-Linear Models (NLM).

3.2 Existing Tools

3.2.1 IBM 360 Trustworthy AI

IBM Research is investing many resources in developing effective tools to measure and improve the trustworthiness of AI systems. Besides testing AI's reliability, the tools are able to certify the robustness of different types of attacks, explain certain predictions, and increase end-to-end transparency and fairness. The theoretical and algorithmic frameworks are developed in an extensible manner and all code is accessible and open-source, which motivates developers to contribute as well. For specific tools, intuitive web demos are provided that demonstrate available capabilities and give an intuition about the applied benefit. Developers have the ability to test examples and execute Jupyter Notebooks that demonstrate the capabilities of models and sample data in different domain applications. Additional resources, such as videos and scientific papers, provide further insights into the developed tools [6]. The following listing briefly explains available tools:

- *ART: Adversarial Robustness Toolbox* is a Python library that provides the possibility to test ML models on their adversarial robustness. Different forms of attacks can be applied to compute scores and detect possible vulnerabilities.

- *AI Privacy 360* is a framework that has the ability to reveal potential privacy risks of ML models. It checks if applicable privacy regulations are respected or disregarded. Further, the trade-off between privacy and performance can be assessed, supporting developers to make the right decisions.

- *AI Explainability 360* provides both model-specific and model-agnostic algorithm to help the user understand the reasoning behind ML models' predictions. The tool supports explanations for different stakeholders (e.g. developers, users, regulators) by regulating the form and mathematical depth of the output.

- *AI Fairness 360* contains fairness metrics which can be used to identify potential biases or prejudices in ML models. It helps developers to identify weaknesses in the datasets and improves the representativity of different subgroups.

- *AI FactSheets 360* is a standardized framework that simplifies the creation and maintenance FactSheets. These are documents containing important meta-data about the training, testing and deployment of ML models. They provide a comprehensive overview, as they are continuously updated over the entire life-cycle.

- *Uncertainty Quantification 360* is a Python package that provides the possibility to measure and improve the uncertainty of a model's predictions. Further, the communication of uncertainty is provided in the form of probability scores and prediction intervals, which generates transparency.

- *Causal Inference 360* is a Python package that supports developers to conduct a causal inference analysis and execute diverse model evaluations. It helps to identify the cause and effect behind the predictions of ML models.

3.2.2 Local Interpretable Model-Agnostic Explanations (LIME)

LIME is a tool that provides local explanations for a certain data point and helps to understand reasons for specific predictions. It allows analyzing the behavior of any ML model by applying an interpretable linear model as an explanator. Despite the fact, that the linear model does not align with the overall behavior of the Gray Box model, it supports the understanding which makes it locally useful. LIME is a model-agnostic explanator and can be used to explain various ML models, such as RFs, SVMs, and NNs [55].

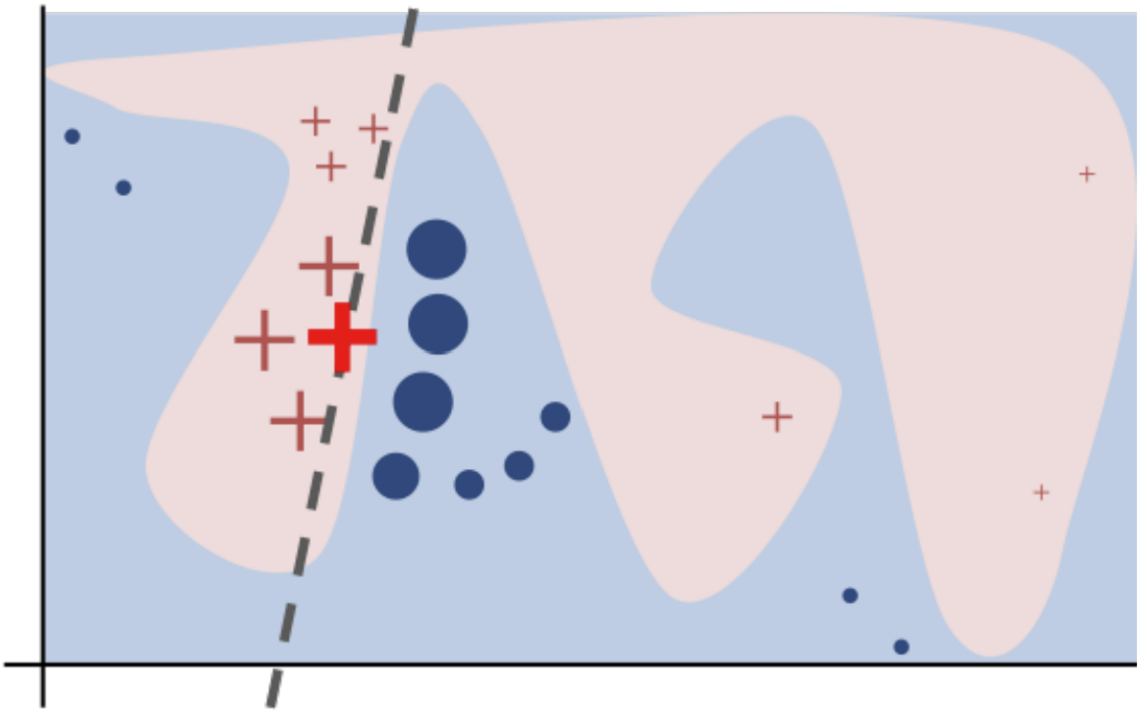


Figure 3.2: Illustration Of LIME Approximation [55]

Figure 3.2 shows an illustration from the paper of Ribeiro et al. [55], which depicts a possible explanation using LIME. The colored areas correspond to decision regions for a binary classification problem, and the bold red cross indicates the observation of interest. The blue circles and red crosses represent data of interest (based on proximity), sampled by the algorithm. Finally, the dashed line represents a simple linear model, explaining the local classification of the bold red cross. This illustration nicely shows, that the explanator is not aiming to explain the complex decision function of the Gray Box globally, but rather helps to understand the local reasoning.

3.2.3 SHapley Additive exPlanations (SHAP)

Similarly to LIME, SHAP is a local explanator that aims to provide explanations for the prediction of an individual data point. It does this by calculating the contribution of each feature to the particular prediction. These explanations are generated using Shapley values, which come from coalitional game theory and describe how the influence of the prediction can be distributed among each feature value in the model to reflect their individual contributions to the prediction. It does this by calculating the Shapley values for every possible combination of features and taking the average. However, the calculation grows exponentially for every additional feature. Representative sampling address this problem [56].

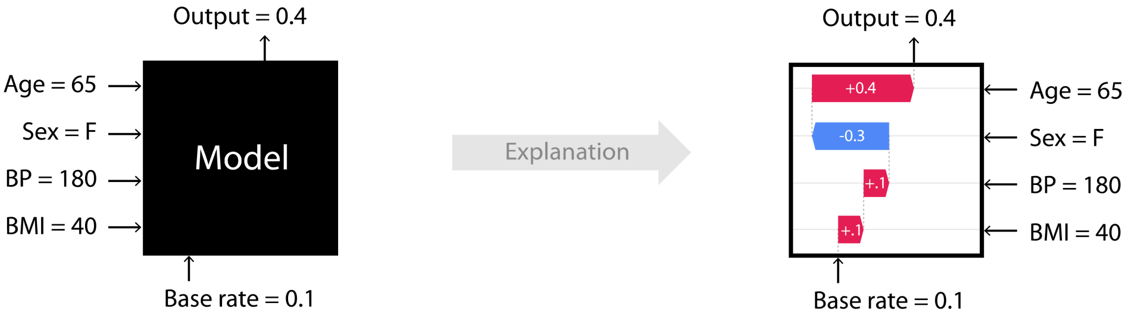


Figure 3.3: Illustration Of SHAP Tool [56]

The SHAP tool offers three distinct explainers. First, the model-agnostic KernelSHAP explanator, which utilizes local surrogate models and kernels for estimating Shapley values. Second, the model-specific TreeSHAP explanator, which is specifically designed for tree-based models, such as DT, RF, and gradient-boosted trees. It provides optimized explanations and is a fast alternative to KernelSHAP. Third, the model-specific DeepSHAP explanator, which is a fast approximation algorithm for calculating SHAP values in deep learning models. Figure 3.3 depicts one simplified example of the SHAP tool, in which one particular prediction of a Gray Box model is explained and the contribution of each feature towards the final prediction is visible. Assume, the square on the left shows an ML model that predicts the probability to suffer from diabetes. The model takes values about the age, sex, blood pressure, and body mass index from the person to be diagnosed and outputs the expected probability. Applying SHAP, the attribution of each feature towards the prediction is visualized as arrows within the square, as depicted on the right. The value of +0.4 indicates, that the age of 65 contributed positively and did increase (red) the probability of the prediction. Instead, the value of -0.3 for the sex feature contributed negatively and did decrease (blue) the probability of being diagnosed with diabetes. This can be useful for the doctor to assess if the prediction of the model can be trusted or not.

3.2.4 ELI5

ELI5 is a Python tool that helps in troubleshooting ML classifiers and providing simple explanations for their predictions. It supports different frameworks, such as Scikit-learn and Keras, and provides an interpretation of the model's weights. However, the explanation in Scikit-learn is limited to linear models and tree-based classifiers only. For Keras models, it is able to explain predictions of a wide variety of image classifiers (CNN model-families) via *Gradient-weighted Class Activation Mapping* (Grad-CAM) visualizations [57]. These visual explanations use the gradients of a classification network and let them flow into the final convolutional layer. This process generates a rough localization map, highlighting the most important regions in the image for predicting the class, without architectural changes or the need of re-training of the network [58]. Image classification models may use unexpected and inappropriate signals in the data to provide an answer. A common instance of this is the misclassification of a husky as a wolf, where the NN learned to associate the presence of snow in the background as a key indicator for classifying the animal as a wolf [59]. With the help of localization maps, developers may address this issue and improve the model's stability.

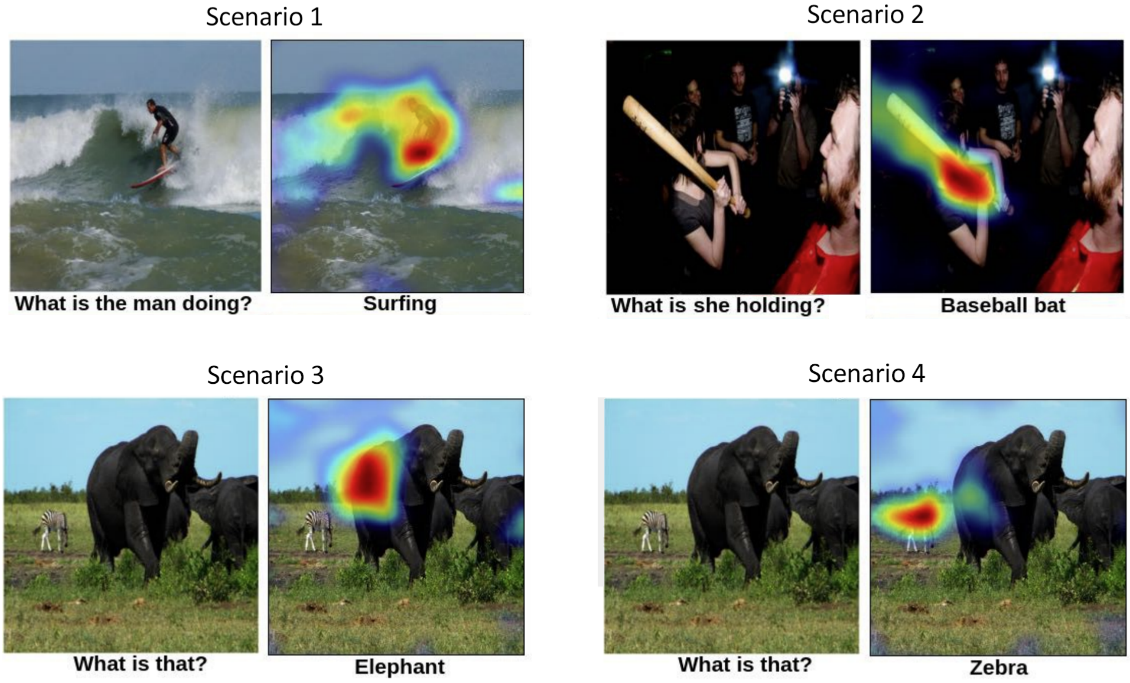


Figure 3.4: Illustration Of Grad-CAM's Localization Map, Based On [58]

Figure 3.4 shows four different scenarios, each consisting of two images. On the left, an image from the *Visual Question Answering* (VQA) [60] dataset is shown together with a question. On the right, the answer (prediction of a CNN + LSTM-based classifier) to the respective question is shown together with a localization map generated by Grad-CAM, highlighting regions relevant for predicting the answer. In Scenario 1 the model successfully recognized a human standing on a board with waves in the background as a surfer. In Scenario 2 the model correctly emphasized the lower part of the baseball bat stronger, because the question is related to the object that the woman is holding in with her hands. In Scenarios 3 and 4 the model was presented with the same image. The question is formulated in a way, such that both responses are correct, and the location map shows the attention for the respective classification. For Scenario 3 the attention lies on the elephant, whereas in Scenario 4 the zebra is highlighted. This indicates that interpretable explanations can be provided even for complex models. Further, according to Das et al. [61], Grad-CAM's visualizations showed a positive correlation of 0.136 compared to human attention maps (manually labeled), which is higher than by chance or from random attention maps (zero correlation) [58].

3.2.5 Microsoft Fairlearn

Microsoft Fairlearn is an open-source tool that is enhanced and maintained by many developers worldwide. Microsoft supports the community and fosters exchange through platforms, such as Discord, StackOverflow, GitHub, and Twitter. The aim is to provide information about the concept of fairness and to present Fairlearn's metrics and algorithms, considering their wider societal impact. Metrics allow the user to assess negatively affected groups and provide a possibility to compare different models with respect to their fairness. Algorithms on the other side, allow the user to actively mitigate discrimination in unfair scenarios [62].

3.2.6 Trusted AI

The *Trusted AI* platform is an application that consists of an algorithm capable of quantifying the trustworthiness level of ML models and visualizing the results in a user interface. Automatically generated trust reports indicate the model's potential for improvement and allow for a simple comparison between different scenarios. The trustworthiness score is computed by aggregating several metrics from distinct pillars into one global trust score.

In order to assess the trustworthiness, the user needs to upload the ML model, used training and testing datasets, and the FactSheet which is providing additional information about the AI lifecycle. The methodology of the trust score computation is depicted in Figure 3.5. Various metrics are computed for every individual pillar (*fairness*, *explainability*, *robustness*, and *methodology*) taking the provided inputs as parameters and returning the metric value as a result. As shown in the figure on the left, the metric values are then mapped into scores ($Score_{1,E}$, $Score_{2,E}$, etc.), which allow the values to be interpreted and compared. The respective mapping function is individual for every metric and maps the value to a trust score from one to five. In this scale, a score of *one* corresponds to the lowest/worst score, while a score of *five* represents the highest/best score. For this mapping, thresholds are derived from literature and the state of the art. However, the user is provided the possibility to customize the thresholds depending on the use-case-specific

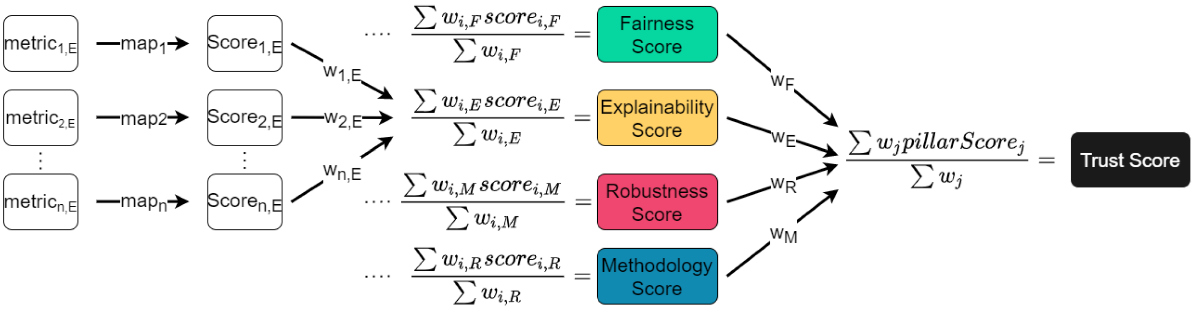
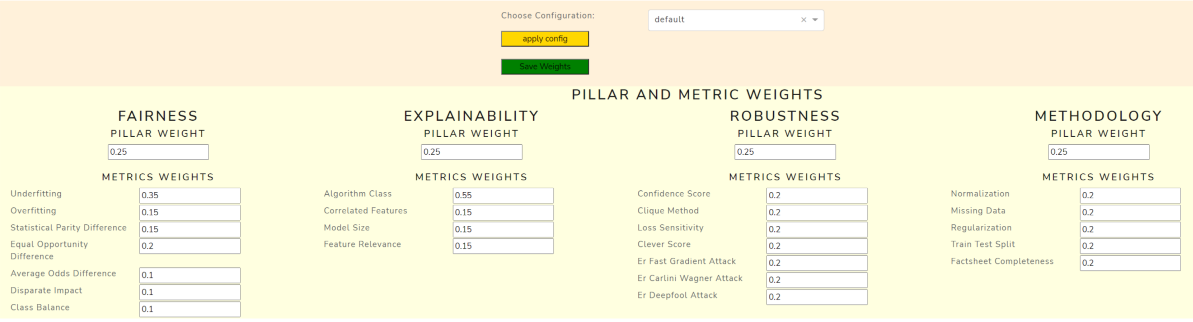

metric$_{1,E}$
—map$_1$→
Score$_{1,E}$
metric$_{2,E}$
—map2→
Score$_{2,E}$
metric$_{n,E}$
—map$_n$→
Score$_{n,E}$
w$_{1,E}$
w$_{2,E}$
w$_{n,E}$
$\frac{\sum w_{i,F} score_{i,F}}{\sum w_{i,F}} =$
$\frac{\sum w_{i,E} score_{i,E}}{\sum w_{i,E}} =$
$\frac{\sum w_{i,M} score_{i,M}}{\sum w_{i,M}} =$
$\frac{\sum w_{i,R} score_{i,R}}{\sum w_{i,R}} =$
Fairness Score
Explainability Score
Robustness Score
Methodology Score
w$_F$
w$_E$
w$_R$
w$_M$
$\frac{\sum w_j pillarScore_j}{\sum w_j} =$
Trust Score

Figure 3.5: Methodology Of The Trust Score Computation For The Trusted AI Platform [12]

context by adjusting the values in the $configuration_{mapping}$ file. Once the individual scores for all the metrics are derived, the algorithm aggregates all individual metric scores into one single score per pillar. For this, weights indicate how much importance a specific metric has and allows prioritizing certain metrics by computing the weighted average. The computation of the global trust score from the individual pillar scores follows the same mechanism, as shown in Figure 3.5 on the right. Weights are allocated to the four pillars, and the overall trust score is calculated as the weighted average of the scores assigned to each pillar. Figure 3.6 shows the default configuration setting of the weights, which can be parameterized and saved by the user. For instance, the pillar score for explainability is computed as follows, using the default weights:

$$
\begin{aligned}
Score_{Explainability} &= \sum \frac{w_{i,E} * score_{i,E}}{\sum w_{i,E}} \\
Score_{Explainability} &= 0.55 * score_{AC} + 0.15 * score_{CF} + 0.15 * score_{MS} + 0.15 * score_{FR}
\end{aligned}
\tag{3.1}
$$

The first input field per pillar corresponds to the pillar weight. The following input fields represent the metric weights. One can recognize, that for the computation of the pillar score, not all the pillars must have an equal number of metrics. This makes the algorithm extensible for future enhancements, in the case that new metrics will be added.


Choose Configuration: default

PILLAR AND METRIC WEIGHTS

FAIRNESS
PILLAR WEIGHT
0.25
METRICS WEIGHTS
Underfitting 0.35
Overfitting 0.15
Statistical Parity Difference 0.15
Equal Opportunity Difference 0.2
Average Odds Difference 0.1
Disparate Impact 0.1
Class Balance 0.1

EXPLAINABILITY
PILLAR WEIGHT
0.25
METRICS WEIGHTS
Algorithm Class 0.55
Correlated Features 0.15
Model Size 0.15
Feature Relevance 0.15

ROBUSTNESS
PILLAR WEIGHT
0.25
METRICS WEIGHTS
Confidence Score 0.2
Clique Method 0.2
Loss Sensitivity 0.2
Clever Score 0.2
Er Fast Gradient Attack 0.2
Er Carlini Wagner Attack 0.2
Er Deepfool Attack 0.2

METHODOLOGY
PILLAR WEIGHT
0.25
METRICS WEIGHTS
Normalization 0.2
Missing Data 0.2
Regularization 0.2
Train Test Split 0.2
Factsheet Completeness 0.2

Figure 3.6: Configuration Setting Of The Weights [12]

The results of the computation are then visualized in a dedicated *analyze* page, which allows an intuitive and simple interpretation of the results. Figure 3.7 depicts one example of a possible analysis. On the left, the overall score informs the user about the global trustworthiness level of the selected scenario and about which pillar has a strong/weak trust score. Additional information about each pillar is presented after clicking on the buttons below the respective pillar (e.g. fairness button). A more detailed view is depicted on the right of Figure 3.7, where the user can inspect the individual metric scores. By examining the metric scores, one can identify whether they had a positive or negative influence on the final score. For instance, a robustness score of 1.5/5 indicates, that the model's robustness is insufficient and that the model may be vulnerable to different attacks (fast gradient, carlini wagner, and deepfool). This is particularly valuable because it allows for the immediate detection of low scores and provides insight into which metrics need improvement to achieve a higher trust score.

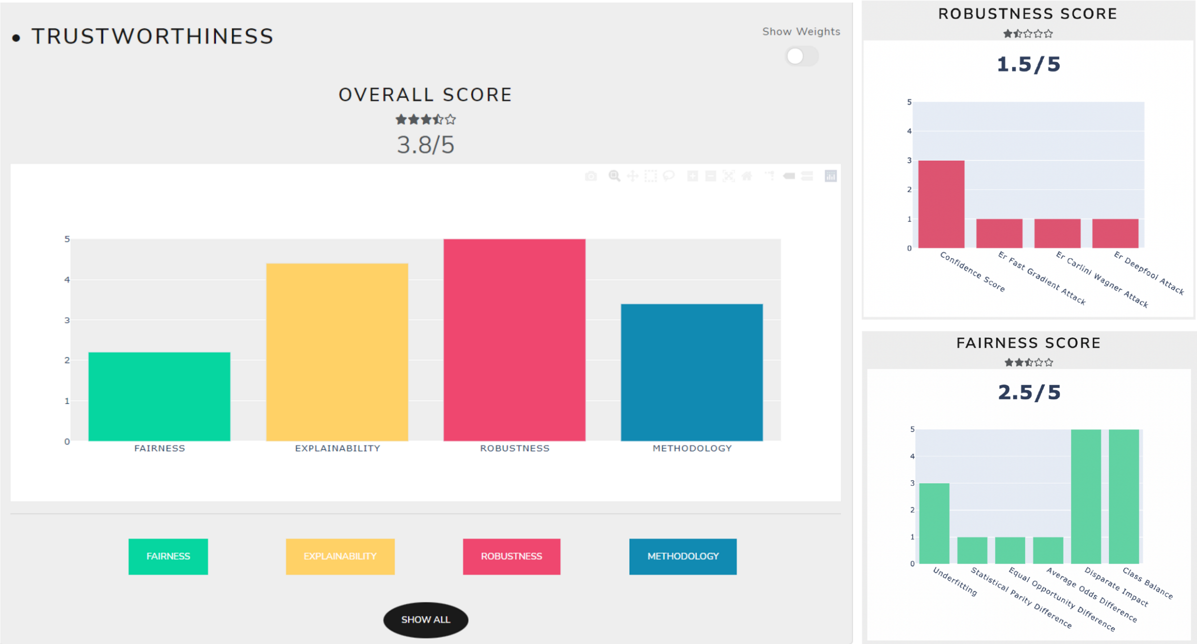


Figure 3.7: Visualization Of The Trustworthiness Score Per Pillar [12]

3.3 Limitations and Missing Aspects

In the previous Sections, important related work has been reviewed and existing tools have been analyzed. In the last few years, the field of trustworthy AI is gaining more awareness and getting increasingly researched. Especially in recent work, scientists have developed diverse methods for making ML more transparent and understandable, which increases trust in such systems. By understanding the physical implications of ML, developers can improve the model, for instance with respect to robustness or fairness. But not only academia is intensively researching this field - also innovative examples from the industry (e.g. IBM and Microsoft Research) have shown useful progress toward trustworthy AI. Approaches allow developers to interact with predictions and generate interpretable explanators even for highly complex ML models. The existing platform by Leupp et al. [12] provides the possibility to measure trustworthiness and compare different solutions with respect to fairness, explainability, robustness, and methodology. Through a deep analysis of the state of the art, one major limitation was identified. Most related work, analyzing trustworthiness for White and Gray Box ML models, assumes that all information about the model's architecture and dataset is available. Considering Black Box models, previous work primarily attempts to make highly complex models more tangible through various approaches. Despite that, to my best knowledge, no material can be found in the literature which measures the trustworthiness of Black Box ML models, when only limited information is available. However, this is an important role in trustworthy AI because depending on the application, ML models may be trained on personal data which is protected and should not be shared with a third party. For instance, a hospital or insurance company would not be willing to upload sensitive data into an online tool, in order to compute the trustworthiness of its ML model. Current methods can be used if the model architecture is already known and the used dataset is accessible. However, modern API-based services produce more challenges because of the relative Black Box nature. This makes the computation of trustworthiness for certain metrics not applicable. For the presented work, the aspect of the missing dataset is the most relevant. Therefore, the next Section explores existing solutions for data augmentation and data generation with the aim to enhance the existing solution and make it applicable to the scenario having only limited information.

3.4 Data Augmentation and Generation

Synthetic data is computer-generated information for testing and training AI models and it can be generated in various ways. The concept is not new, and methods for data augmentation have been applied especially in the field of image processing and natural language processing (NLP), where complex and data-intensive models are trained and the collection of real data may be expensive, time-consuming, or limited. For this, often generative AI is applied using transformer-based foundation models [63], diffusion models [64], and generative adversarial networks (GANs) [65]. GANs apply a generator discriminator approach proposed by Goodfellow [35] where the generator creates artificial samples from random noise, and the discriminator has to distinguish between these artificial and real samples. This approach is depicted in Figure 3.8 and results in synthetic data that contains similar characteristics to the original set. As the Figure illustrates, GANs may be well suited for data augmentation, but not for data generation. This is because it requires access to the original dataset and to feed it to the discriminator. However, this access is not applicable in the case being presented a Black Box model with limited information, since the trustworthiness score must be computed under the restriction of having limited information (no dataset at all). Recent work has also shown progress in the generation of tabular data to balance out uneven data distribution between train and test data [67]. However, also here a base dataset is required, from which the generative model is able to learn representations of the underlying data in order to generate versions in a similar style. This clearly indicates the necessity, that the presented work needs to present a novel approach, which allows the synthetic data generation from scratch.

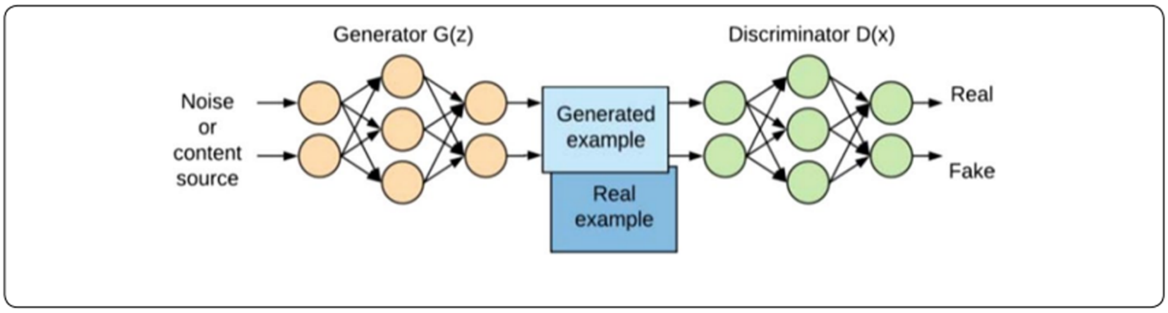


Figure 3.8: Illustration Of GAN Concept For Data Augmentation [66]

3.5 Learnings from Related Work

The thorough analysis of related work with a focus on complex Black Box models has identified work, that aim to make ML/DL models more transparent and understandable in order to increase trust. Existing tools have shown the ability to provide local and global explanations, which supports the understanding of the model's behavior. Further tools have been developed, capable of assessing the trustworthiness level of ML models by the computation of a global trust score. However, existing tools assume that all information about the model's architecture and dataset is available. This forms a clear limitation, as most metrics become incomputable when measuring the trust of a Black Box model whose original dataset is not provided, resulting in a limited trust score. Therefore, the goal of this thesis is to improve the overall trustworthiness assessment for Black Box models, by presenting a novel approach. This approach includes the design and implementation of a synthetic dataset generator that is fundamentally different from previously explored data augmentation and generation methods, as it does not require access to the original dataset.

Chapter 4

Design and Implementation

This Chapter provides insights into the Design and Implementation of an advanced approach, to improve the metric computation for the Black Box scenario, where limited information about the used dataset is available. The goal is to provide a more accurate global trust score for a specific solution under the constraint of having only limited information available. The design Section includes the creation of a Black Box taxonomy, which groups ML and DL algorithms in the respective subgroups, based on their characteristics and specific properties. Further, the taxonomy also distinguishes the level of available information. The next Section motivates the synthetic dataset generation as a solution to the before mentioned challenge and shows the process design of the dataset generation. Two different solutions are presented for the dataset generation, varying in the level of available information provided by the user of the system, as the following Sections will show. The trade-off between accuracy and privacy is described in further detail. The implementation Section explains the current algorithmic design and proposes a possible integration into the Trusted AI [12] platform for an autonomous global trust score computation.

4.1 Design of Black Box Taxonomy

A Black Box model is either too complex for any person to interpret or has missing internal knowledge, which is kept secret and thus not available (see Section 2.2). The complexity of an ML model affects how well the model can be explained. The chosen learning technique plays a major role in determining what metrics can be computed in the fairness, explainability, and robustness pillar. It influences the overall functioning of the model and limits the potential post-hoc explanations. An analysis of various ML/DL models provides details and justifications for the level of understandability and explainability achievable in their decision-making processes and lays the foundation for the first hierarchy of the Black Box Taxonomy, as depicted in Figure 4.1. The second hierarchy distinguishes the level of *Available Information* for all subgroups from the first hierarchy into *Everything Known*, *Missing Dataset*, and *Unknown Underlying Architecture*.

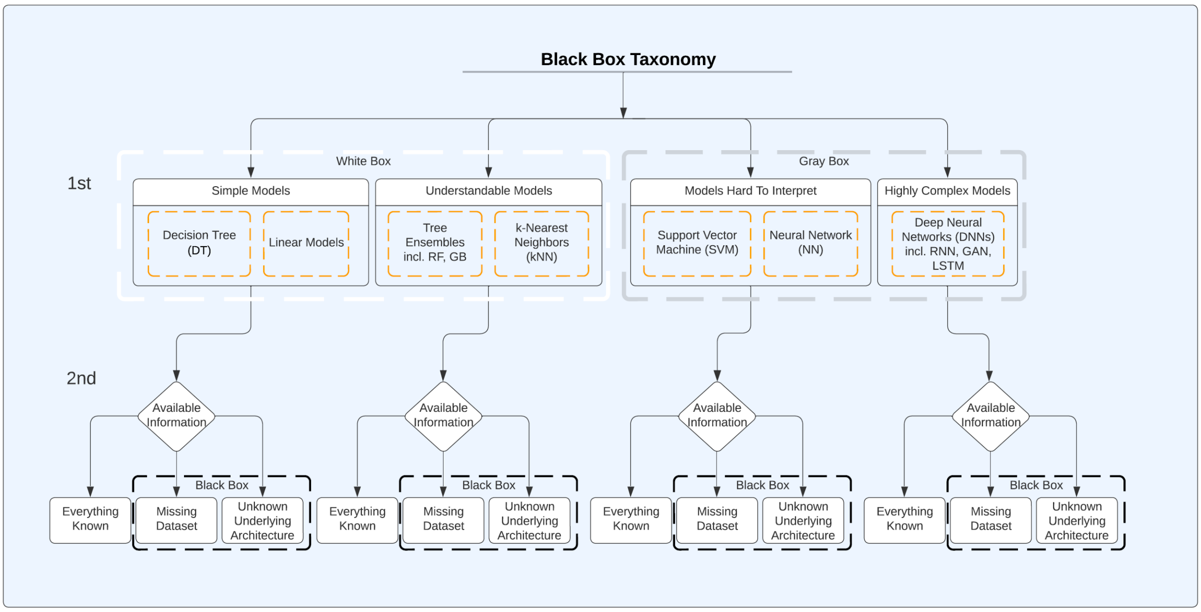


Figure 4.1: Black Box Taxonomy

The following analysis provides details and explanations of the most prevalent White Box ML models and examines their properties and possible configurations. It also groups each model into a subgroup called *Simple Models* and *Understandable Models* from the first hierarchy of the Black Box Taxonomy (see Figure 4.1), due to their high interpretability and transparency. All subgroups of the Black Box Taxonomy are expandable with ML models that have similar properties. Thus, subgroups are not a complete collection of all possible models but are meant to provide a structure to categorize future models.

- Decision Tree **(DT)** is a model that is considered easily understandable and interpretable for humans. The DT algorithm is a graphical representation of a tree, composed of a root, multiple nodes, branches, and leaves. The nodes in the tree serve as crucial points where decisions are made, based on specific conditions learned by the algorithm in the training phase. These conditions are based on the feature with the highest information gain, to optimally divide and separate the data [68]. One big advantage of DTs is the human ability to interpret the reasoning process for any prediction of the model, simply by following the branches of the tree. In fact, a DT can be transformed into a collection of decision rules in an ***if-then*** format [69], where the outcome corresponds to the class label of a leaf node:

  $$\boldsymbol{\mathit{if}}\, condition_1 \wedge condition_2 \wedge condition_3,\ \boldsymbol{\mathit{then}}\, outcome \tag{4.1}$$

  Further, not only single predictions can be explained, but the transparent properties of the model allow users to inspect the most significant features for predicting the whole dataset - the features with the highest information gain are positioned at the top of the tree, whereas features with a lower information gain are positioned lower in the tree. As described in Section 3.1, DTs are often used as explanators for *Highly Complex Models*, due to their interpretable properties by design. The size of the model depends on the tree depth and number of rules. This design ensures excessive comprehensibility and explainability, especially when it is not deeply branched and the features themselves are understandable [68]. Therefore, DT is grouped into the subgroup *Simple Models*.

- Linear Models are considered one of the simplest models in the field of ML. They are widely established and serve as the foundation for numerous sophisticated ML techniques such as DNN. Linear models may be further split into Linear Regression and Logistic Regression models. The former is used for regression tasks where the algorithm learns the linear correlation among variables, and the latter is a classification algorithm. It assumes that the dataset is linearly separable and tries to learn the weight of each feature. Mathematically, the linear regression can be written in matrix form as

$$y = X\beta + \varepsilon \quad with\ \varepsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{4.2}$$

  where y is the target variable, β is the vector of coefficients that is learned by the model, ε is the residual term for adjusting the error, and X is the provided feature matrix. When the input feature X consists of a single feature, the Linear Regression model fits the line in a two-dimensional space so that the residuals between the predicted and actual values are minimized. Commonly, the *least squares* loss function is used to optimize the vector coefficients β and ε towards the optimal solution and minimizes the residuals (distance between predicted and observed values) [70]. Linear Models have the property of being interpretable due to their simple and human-understandable mathematical computation. The weights [W] are interpretable and predictions can be clarified. Therefore, Linear Models are grouped into the subgroup *Simple Models*.

- Tree Ensemble Models, such as Random Forest **(RF)** and Gradient Boost **(GB)**, enhance the predictive capabilities of individual learning algorithms by combining multiple algorithms together. These ensembles incorporate predictions from multiple decision trees, where each tree is trained on a distinct subset of the input data, considering both features and samples. For classification tasks, the RF outputs the class that is chosen by the majority of trees (majority voting). In regression tasks, the RF returns the mean or average prediction of the individual trees. One benefit of RF models is that they reduce overfitting compared to simple DTs and help to improve accuracy. However, the interpretation is more difficult because multiple independent decision trees are applied. Besides the maximum depth of the trees, the number of trees can also be specified as a hyperparameter at the instantiation [71]. While the ensemble model loses some of their explainability as a whole, it still provides an estimate of the feature's relevance in classification. GB similarly consists of an ensemble of decision trees. However, instead of computing independent scores for each tree, it aggregates the results of each decision tree along the way to calculate the final result [72]. This improves the accuracy of the model because trees are able to correct the error of the previous tree but makes the trees more dependent on each other. This again influences the interpretability of such models. Therefore, Tree Ensemble Models are grouped into the subgroup *Understandable Models*.

- K-Nearest Neighbors **(kNN)** is a non-parametric model which classifies a data record based on the similarity of its k nearest neighbors. This simple but effective method is similar to how a human learns, and thus rather intuitive. For single instances, the decision-making process can be understood by examining the identified neighbors and their corresponding labels. However, the success of kNN depends on the definition of the parameter k and other factors, such as whether distance-based weighting is taken into account or not [73]. When difficult problems require complex distance functions to be applied and the parameter k takes a higher value, the understanding of the model's general behavior may suffer, as the model incorporates a larger number of neighbors and potentially more complex relationships. Also, kNN does not provide an implicit way of directly extracting feature importance. However, it is possible to gain insights indirectly by examining the relevance of features among the nearest neighbors. Therefore, kNN is grouped into the subgroup *Understandable Models*.

Table 4.1 shows a categorization of Gray Box ML/DL models based on findings from the previous Section 3.1. The groups and properties of the ML classes are explained in further detail below.

Table 4.1: Categorization Of Gray Box Models Into Subgroups (i-ii), And Visualization Of Black Box Terminology (iii)

	Gray Box Models			Black Box	
	(i) Hard To Interpret		(ii) Highly Complex	(iii) Missing Knowledge	
	SVM	NN	DNN	Dataset	Underlying Architecture
interpretability	medium	low	very low	*	*
complexity	medium	high	very high	*	*
transparency	medium	medium	medium	low	low
# parameter	low	medium	high	*	*

The table assigns Gray Box models into two different subgroups: (i) models whose reasoning of the results are *Hard To Interpret* even for domain experts in practical fields, and (ii) *Highly Complex Models* due to their mathematical complexity. For each model the differences in the properties *interpretability, complexity, transparency*, and the number of parameters (*# parameter*) are described. The two subgroups (i-ii) are used in the first hierarchy of the Black Box Taxonomy in Figure 4.1. The last column (iii) of the Table shows that any model can be considered a Black Box when *Missing Knowledge* is present. This category is depicted in the second hierarchy of the Black Box Taxonomy and adds another dimension for every existing subgroup.

(i) Support Vector Machines **(SVMs)** are supervised ML models, which use a subset of the training data (support vectors) to create a hyperplane, separating the data into different classes. The area around the hyperplane remains as wide as possible free of objects and serves as a decision boundary for the remaining data. It is a Linear Model capable of solving linear and non-linear problems using available kernels, such as the *Radial Basis Function* or the *Quadratic Kernel*. These and similar kernels provide an efficient method for converting data into higher dimensions but make results very difficult to comprehend. A different model of the same subgroup is the Neural Network **(NN)**, which is inspired by the biological brain, consisting of many neurons which can send and process signals through weighted connections. The information travels from the input layer through one or more hidden layers, in which signals are transformed through activation functions, before finally reaching the output layer. Since NNs are highly configurable regarding different hyperparameters, such as the number of hidden layers, dropout rate, selection of the activation function and batch size, the number of parameters may vary. Compared to SVMs, NNs contain more parameters and are considered to be more complex systems. Nevertheless, both models are hard to interpret and considered as Gray Box models but still less complex than Deep Neural Networks. That is why they belong to the subgroup called models *Hard To Interpret.*

(ii) Deep Neural Networks **(DNNs)** are special types of NNs, that can handle complex non-linear relationships by using numerous hidden layers. Due to their large size, DNNs require more time for training compared to regular NNs, but have the ability to abstract into higher dimensions. The main difference between NNs and DNNs is that DNNs are deeper and use more complex node architectures. This is also reflected in the very high number of parameters. Often, DNNs are applied in the field of computer vision and natural language processing tasks. To address the problem of sequential data, a special class of DNNs has evolved, called Recurrent Neural Networks (RNNs). They introduce feedback connections, enabling them to consider context from previous time steps while calculating the importance of the current input. Long Short-Term Memory (LSTM) is a type of RNN designed to address long-term dependencies in sequential data. Furthermore, Generative Adversarial Networks (GANs) are systems that train two NNs simultaneously, to generate realistic data such as images or text. As it can be seen, many variations of DNNs exist in practice, which all have address complex problems and as result the number of parameter grow exponentially large. Therefore, DNNs including the above-mentioned variations have their own subgroup called *Highly Complex Models.*

(iii) Lastly, the term Black Box describes any possible ML/DL model for which information about the *Dataset* (used for training/testing the model) or the *Underlying Architecture* is not known. As only limited information is available, the transparency of the model is considered low. The asterisk (*) serves as a placeholder and indicates, that these values depend on the particular model and its configurations.

4.2 Problem Definition

The analysis of related work has shown, that current solutions (see Subsection 3.2.6) are capable of assessing the trustworthiness score of ML models, with respect to fairness, explainability, robustness, and methodology. However, one major limitation is the assumption, that all information about the model's architecture and dataset is available. This causes problems in the computation of the global trustworthiness score since various metrics require access to the dataset, used for training and testing the respective model. Considering the Black Box Taxonomy from Figure 4.1, this applies to all subgroups when in the second hierarchy the situation *Missing Dataset* or *Unknown Underlying Architecture* is true. The consequence of having fewer metrics computed per pillar is, that it can lead to a limited and potentially incomplete understanding of the underlying trustworthiness of the system being evaluated. Relying on a few metrics can overlook important aspects and nuances, resulting in an oversimplified representation of the pillar score. This is especially critical if certain pillars end up having no quantifiable metrics at all, making it difficult to compute a global trust score without neglecting the respective pillar. This incomplete assessment also makes the comparison of models more challenging. Therefore, this work approaches an alternative way to generate estimations for such incomputable metrics using a synthetic dataset.

4.2.1 Existing Metrics

This Subsection provides an overview of existing metrics in the Trusted AI platform [12]. For every metric, the following properties are defined:

- **Pillar**: Name of the pillar towards which the metric is contributing to.
- **Metric**: Name to describe and reference the metric.
- **Input**: Variables that are required for the computation of the metric (e.g. Model, Training Dataset, Testing Dataset, FactSheet).
- **Condition**: Special condition which restricts the applicability of the metric. The minus sign (-) means, that no restrictions exist and the metric can be applied to all scenarios.
- **Output**: The result of the metric computation, used for the later score mapping.

Since the output of the metrics can not be compared directly, due to the outputs being of different data types, scales, and meanings, the values have to be mapped onto a normalized scale (one to five) to achieve a comparable trust score. The respective mapping functions are defined based on good practices that are defined in the literature. Since the mapping function and their parameter are not influenced in the scenario of computing the trustworthiness score with only limited information, the presented work is not further elaborating on the mapping. For more information, it is referred to read Section 4.1 *Algorithm Design* of the work by Leupp et al. [12]. Table 4.2 provides an overview of all metrics, including the properties as defined above. In the column named *Input*, the term *Dataset* is highlighted in bold. This indicates that many metrics rely on this information and consequently will get incomputable, under the restriction of not having access to the original dataset.

Table 4.2: Metrics Calculated By The Trusted AI Platform And Their Properties [11]

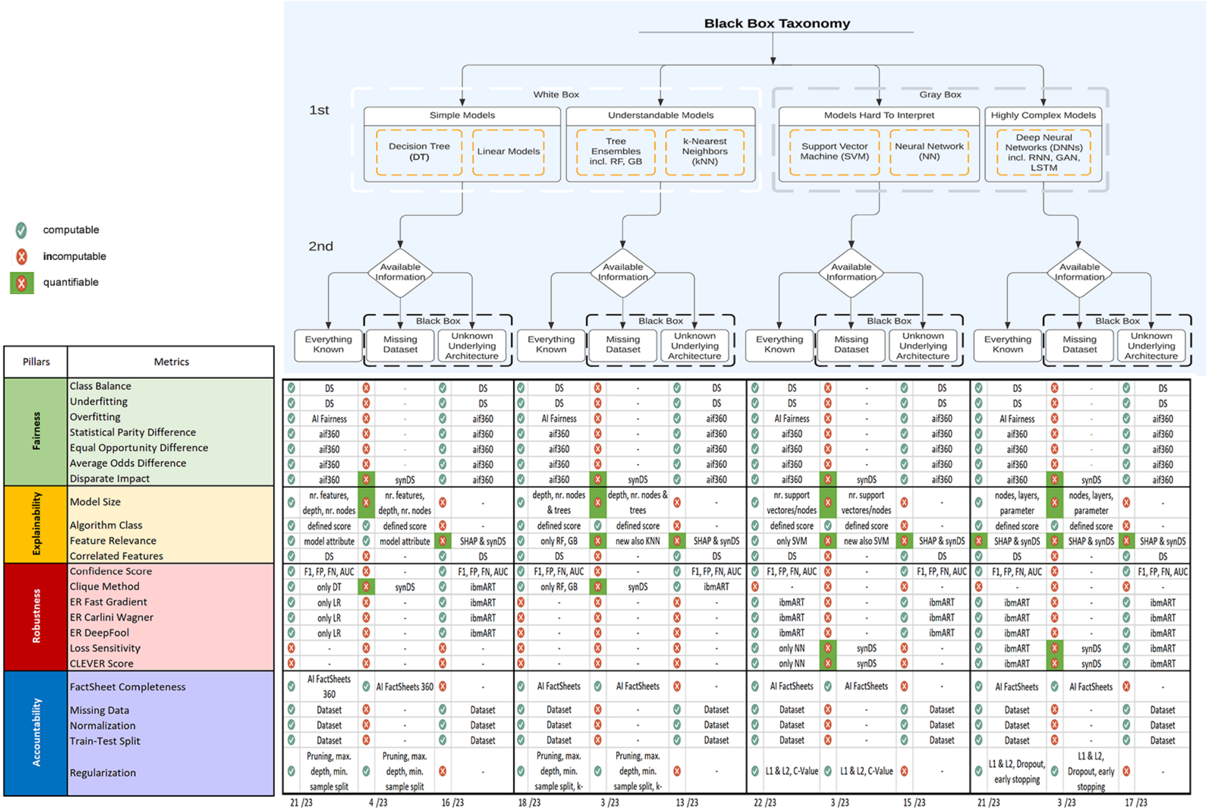
Pillar	**Metric**	**Input**	**Condition**	**Output**
Fariness	Underfitting	Training & Testing **Datasets**, Model	-	[0-1]
	Overfitting	Training & Testing **Datasets**, Model	-	[0-1]
	Statistical Parity Difference	Training **Dataset**, FactSheet	Applicable if a protected group and a favorable outcome are defined	[0-1]
	Equal Opportunity Difference	Testing **Dataset**, Model, FactSheet	Applicable if a protected group and a favorable outcome are defined	[0-1]
	Average Odds Difference	Testing **Dataset**, Model, FactSheet	Applicable if a protected group and a favorable outcome are defined	[0-1]
	Disparate Impact	Testing **Dataset**, Model, FactSheet	Applicable if a protected group and a favorable outcome are defined	[0-1]
	Class Balance	Training **Dataset**	-	Class %
Explainability	Algorithmic Class	Model	-	Name
	Correlated Features	Training & Testing **Datasets**	Applicable on features with non-missing values	[0-1]
	Feature Relevance	Model	Applicable for models providing features relevance scores	[0-1]
	Model Size	Training **Dataset**	-	Integer
Robustness	Confidence Score	Model, Testing **Dataset**	Applicable on models providing prediction probabilities	%
	Clique Method	Model	Applicable on DT, RF, and GBDT algorithms	Real
	Loss Sensitivity	Model	Applicable on NN algorithms	Real
	CLEVER Score	Model	Applicable on NN algorithms	Real
	ER Carlini Wagner	Model, Testing **Dataset**	Applicable on NN, LR, and SVM algorithms	%
	ER Fast Gradient	Model, Testing **Dataset**	Applicable on NN, LR, and SVM algorithms	%
	ER DeepFool	Model, Testing **Dataset**	Applicable on NN, LR, and SVM algorithms	%
Accountability	Normalization	Training & Testing **Dataset**	-	Name
	Missing Data	Training & Testing **Datasets**	-	Integer
	Regularization	FactSheet	Applicable if the regularization technique details are present in the FactSheet	Name
	Train-Test Split	Training & Testing **Datasets**	-	[0-1]
	FactSheet Completeness	FactSheet	-	[0-1]

4.2.2 Incomputable Metrics

This Subsection aims to make the impact of the missing dataset more tangible, through a visual representation of incomputable metrics. Figure 4.2 shows a matrix where all metrics are listed vertically on the left side, grouped by pillar. Each column corresponds to one scenario from the Black Box Taxonomy (see Figure 4.1). The first three columns of the matrix belong to the subgroup *Simple Models* and differentiate each other in the level of available information. Similarly, the next three columns belong to the next subgroup, and so on. For each cell, it is highlighted if the metric is computable for the respective scenario (green checkmark), or not (red cross). Additionally, helpful information is provided beside the indication about the computability, such as the source of the tool which has already implemented the computation of the metric and other information. Below each column, the ratio of computable metrics is displayed. For *Simple Models* where everything is known, 21/23 metrics could be computed. This ratio is high, and the metrics are computed from all pillars. The second column shows computable metrics for the scenario, where the dataset is not available. A significant drop in computable metrics to 4/23 is identifiable. Especially the pillars *fairness* and *robustness* have to be highlighted, where without exception all metrics get incomputable, since they rely on the uploaded dataset. The third column shows computable metrics for the scenario, where information about the underlying architecture of the model is unknown. Here, the ratio of computable metrics of 16/23 is not alarming, however, the *explainability* pillar is most affected by the lack of information about the underlying architecture of the model. Likewise, for the second subgroup *Understandable Models* similar behavior of incomputable metrics can be observed, except for the metric *Feature Relevance*, since this metric is implemented to be model dependent (only RF&GB) and the three Empirical Robustness (ER) metrics (see *Condition* column in Table 4.2). For subgroups *Models Hard To Interpret* and *Highly Complex Models*, the ratio of computable metrics is comparable to the subgroup *Simple Models*, except for the feature *Clique Method*, which is only applicable for tree-based models. Further, the following *Robustness* metrics can be applied when all information is available: *Loss Sensitivity* and *CLEVER Score* (see *Condition* column in Table 4.2).

4.2.3 Quantifiablility of Incomputable Metrics

As depicted in Figure 4.2, certain incomputable metrics (red cross) are highlighted with a green background. These metrics become quantifiable using the synthetic dataset generator (see Section 4.3) to replace the original test dataset with the synthetic dataset (labeled by the Black Box). Let us derive at these final quantifiable metrics, by evaluating all available metrics individually. It is not possible to replicate the original dataset that was used for the training of the ML model. Therefore, incomputable metrics that do not rely on the *Training Dataset* as *Input*, form a subset of possibly quantifiable metrics. For each metric part of this subset, the exact implementation [12] is further studied in detail:



✓ computable
✗ incomputable
☒ quantifiable

Pillars	Metrics	Everything Known	Missing Dataset	Unknown Underlying Architecture	Everything Known	Missing Dataset	Unknown Underlying Architecture	Everything Known	Missing Dataset	Unknown Underlying Architecture	Everything Known	Missing Dataset	Unknown Underlying Architecture
Fairness	Class Balance	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS
	Underfitting	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS
	Overfitting	✓ AI Fairness	✗ -	✓ aif360	✓ AI Fairness	✗ -	✓ aif360	✓ AI Fairness	✗ -	✓ aif360	✓ AI Fairness	✗ -	✓ aif360
	Statistical Parity Difference	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360
	Equal Opportunity Difference	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360
	Average Odds Difference	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360	✓ aif360	✗ -	✓ aif360
	Disparate Impact	✓ aif360	☒ synDS	✓ aif360	✓ aif360	☒ synDS	✓ aif360	✓ aif360	☒ synDS	✓ aif360	✓ aif360	☒ synDS	✓ aif360
Explainability	Model Size	✓ nr. features, depth, nr. nodes	☒ nr. features, depth, nr. nodes	✗ -	✓ depth, nr. nodes & trees	☒ depth, nr. nodes & trees	✗ -	✓ nr. support vectores/nodes	☒ nr. support vectores/nodes	✗ -	✓ nodes, layers, parameter	☒ nodes, layers, parameter	✗ -
	Algorithm Class	✓ defined score	✓ defined score	✗ -	✓ defined score	✓ defined score	✗ -	✓ defined score	✓ defined score	✗ -	✓ defined score	✓ defined score	✗ -
	Feature Relevance	✓ model attribute	✓ model attribute	☒ SHAP & synDS	✓ only RF, GB	☒ new also KNN	☒ SHAP & synDS	✓ only SVM	☒ new also SVM	☒ SHAP & synDS	☒ SHAP & synDS	☒ SHAP & synDS	☒ SHAP & synDS
	Correlated Features	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS	✓ DS	✗ -	✓ DS
Robustness	Confidence Score	✓ F1, FP, FN, AUC	✗ -	✓ F1, FP, FN, AUC	✓ F1, FP, FN, AUC	✗ -	✓ F1, FP, FN, AUC	✓ F1, FP, FN, AUC	✗ -	✓ F1, FP, FN, AUC	✓ F1, FP, FN, AUC	✗ -	✓ F1, FP, FN, AUC
	Clique Method	✓ only DT	☒ synDS	✓ ibmART	✓ only RF, GB	☒ synDS	✓ ibmART	✗ -	✗ -	✗ -	✗ -	✗ -	✗ -
	ER Fast Gradient	✓ only LR	✗ -	✓ ibmART	✗ -	✗ -	✗ -	✓ ibmART	✗ -	✓ ibmART	✓ ibmART	✗ -	✓ ibmART
	ER Carlini Wagner	✓ only LR	✗ -	✓ ibmART	✗ -	✗ -	✗ -	✓ ibmART	✗ -	✓ ibmART	✓ ibmART	✗ -	✓ ibmART
	ER DeepFool	✓ only LR	✗ -	✓ ibmART	✗ -	✗ -	✗ -	✓ ibmART	✗ -	✓ ibmART	✓ ibmART	✗ -	✓ ibmART
	Loss Sensitivity	✗ -	✗ -	✗ -	✗ -	✗ -	✗ -	✓ only NN	☒ synDS	✗ -	✓ ibmART	☒ synDS	✓ ibmART
	CLEVER Score	✗ -	✗ -	✗ -	✗ -	✗ -	✗ -	✓ only NN	☒ synDS	✗ -	✓ ibmART	☒ synDS	✓ ibmART
Accountability	FactSheet Completeness	✓ AI FactSheets 360	✓ AI FactSheets 360	✗ -	✓ AI FactSheets	✓ AI FactSheets	✗ -	✓ AI FactSheets	✓ AI FactSheets	✗ -	✓ AI FactSheets	✓ AI FactSheets	✗ -
	Missing Data	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset
	Normalization	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset
	Train-Test Split	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset	✓ Dataset	✗ -	✓ Dataset
	Regularization	✓ Pruning, max. depth, min. sample split	✓ Pruning, max. depth, min. sample split	✗ -	✓ Pruning, max. depth, min. sample split, k-	✓ Pruning, max. depth, min. sample split, k-	✗ -	✓ L1 & L2, C-Value	✓ L1 & L2, C-Value	✗ -	✓ L1 & L2, Dropout, early stopping	✓ L1 & L2, Dropout, early stopping	✗ -
		21 /23	4 /23	16 /23	18 /23	3 /23	13 /23	22 /23	3 /23	15 /23	21 /23	3 /23	17 /23

Figure 4.2: Overview Of Computable Metrics Regarding The Black Box Taxonomy

- Equal Opportunity Difference:
  The *Equal Opportunity Difference* (EOD) is a metric from the fairness pillar and is only applicable when for the dataset a protected group and a favorable outcome are defined, e.g. credit risk assessment where the ethnicity of the applicant is a feature used for predicting the likeliness of repaying a credit. In this example, the protected group takes the value of a certain ethnicity and the favorable outcome takes the value describing the likelihood of repaying the credit. This metric captures potential performance issues, where one group is disproportionally benefiting of a higher opportunity being predicted the favorable outcome. Because discrimination towards certain ethnicities is considered unacceptable in a society, the feature should have as little influence as possible on the prediction. Equation (4.3) shows the formal computation of the EOD, where $\widetilde{Y}$ stands for the predicted label of the model, Y stands for the true label, and P stands for the protected group membership.

  $$EOD(\widetilde{Y}, Y, P) = Pr(\widetilde{Y} = 1|P = 1, Y = y) - Pr(\widetilde{Y} = 1|P = 0, Y = y), y \in 0, 1 \tag{4.3}$$

  As $y \in 0, 1$, the computation is measuring the difference between the false positive rates (FPR) and the difference between the true positive rates (TPR) of the unprotected and the protected group. This however, is only possible when the true labels Y of the testing dataset are available. Since no dataset is provided, and the synthetic dataset only allows generating the predicted labels $\widetilde{Y}$, it is not possible to make this metric quantifiable.

- Average Odds Difference:
  Similar to the previous metric, the *Average Odds Difference* is also a metric from the fairness pillar and is only applicable when a protected group and favorable outcome are defined. The aim is to ensure that the model performs equally well for both groups and the effectiveness is similar. It is measured as the mean absolute difference between making accurate predictions (TPR) and misclassifying negative cases as positive (FPR). Again, these metrics can not be computed since the true labels Y are not available, regardless of the synthetic dataset's accessibility. Therefore, this metric does also not become quantifiable.

- Disparate Impact (DI):
  This is the last metric that belongs to the fairness pillar. Again, this metric is only applicable if a protected group and favorable outcome are specified. It measures if protected attributes are relevantly influencing the outcomes of a model, such that the subgroups are treated differently. It does this by dividing the probability of receiving a favorable outcome for the protected group by the probability of receiving a favorable outcome for the unprotected group, as depicted in Equation (4.4).

  $$DI(\widetilde{Y}, P) = \frac{Pr(\widetilde{Y} = 1|P = 1)}{Pr(\widetilde{Y} = 1|P = 0)} \tag{4.4}$$

  As the Equation shows, the computation does not rely on the true labels of the original dataset Y, but only on the predictions of the model $\widetilde{Y}$. Therefore, this metric becomes quantifiable, using the synthetically generated dataset labeled by the model (y_pred).

- Feature Relevance:
  This metric belongs to the explainability pillar since it supports the understanding of the model's behavior and provides insights into the influences of the decision-making process. The current implementation allows the computation of feature importance only for models providing feature relevance scores. From there, the percentage of irrelevant features is computed. As discussed in Section 4.1, especially rule-based systems such as DTs and ensembles of rule-based systems (e.g. RF and GB) provide this possibility, together with SVMs. For more complex Gray Box models, like kNN and DNN, no such functionality is available. Therefore, existing explanator tools like SHAP (described in Subsection 3.2) can be applied to calculate Shapley values and approximate the importance of each feature (code in Appendix A). Therefore, this metric becomes quantifiable even for the subset of *Highly Complex Models* from the Black Box Taxonomy.

- Confidence Score:
  The *Confidence Score* is a metric that belongs to the robustness pillar and is well known to ML developers. In many cases, ML models provide not only the predicted class label but also a probability score for each class. It measures the level of certainty or reliability of a prediction made by an ML model. Considering the implementation, the average certainty over all true predictions is computed, which should be high for robust models. As the labels of the original dataset are not available, the confidence over all *true* predictions can not be computed. Therefore, this metric remains incomputable.

- Clique Method:
  The *Clique Method* is a metric that measures the robustness of tree-based models. Since DTs are using learned rules, instead of a continuous step function, traditional gradient-based evaluation metrics are not applicable. For this reason, the Clique Method (originating from graph theory) can be used to find the largest set of data points that share similar characteristics or fall into the same regions of the DT. For these data points identified by the clique, the tree makes consistent predictions. Identifying this *max clique*, helps in assessing the robustness of the DT model and understanding the stability of its predictions across different data points. Since the true label of the original dataset is not needed, this metric becomes quantifiable using the synthetic dataset. In the case that the maximum clique is small, it can be derived that even small changes to the output of the model result in changes of the prediction and that the tree based classifier is not robust.

- Loss Sensitivity:
  The *Loss Sensitivity* metric uses the loss function of NNs to compute the maximal possible deviation of the output, with respect to small changes of the input. It measures the change of the loss, which is related to the model's decision surface. For loss functions with local extremas and large differences for the gradients, the decision confidence may quickly drop to negative areas [37]. Since for models trained on the Keras framework, the loss function is bound to the model, the current implementation is retrieving it and using the IBM ART library for the metric computations. For this metric, the synthetic dataset can be applied.

- CLEVER Score:
  The *Cross Lipschitz Extreme Value for Network Robustness* (CLEVER) Score is developed for NNs and provides an estimation of the maximum amount by which the output can change in response to small alterations in the input. For this, the product of the Lipschitz constant and the amount of perturbation is used to estimate an upper and lower bound for changes in the output. The current implementation uses the *clever_u* function from the IBM ART library which takes the model and a predicted sample as input and provides the bound limiting the number of changes to the class output. Therefore, this method is also applicable to the scenario using a synthetic dataset.

- Empirical Robustness:
  It is a metric that evaluates the susceptibility of ML models with respect to different types of attacks and belongs to the robustness pillar. The accuracy score of the model is compared to the accuracy when presented with adversarial samples. In the current implementation, three different types of attacks (Carlini Wagner, Fast Gradient, ER DeepFool) are implemented for SVMs and Logistic Regression models. As previously mentioned, the computation of this metric requires the comparison of the accuracy score, which again relies on the true labels (Y) of the original dataset. However, since this information is missing, the metric does not become quantifiable.

- Model Size:
  This metric belongs to the explainability pillar and is an important indicator of the model's interpretability. Large models are harder to understand, because more

parameters influence the prediction of the model. Because models have different architectures, there also exist different approaches how to compute the size of a model. However, a good indication is the number of features used for training the model because in most models they do positively correlate to the number of weights/parameters of the model. This corresponds exactly to how the current implementation measures the size of the model. As this implementation depends on the dataset, it becomes incomputable when the dataset is not available. Applying the synthetic dataset however, this metric becomes quantifiable again. Further, Figure 4.2 provides additional information next to the checkbox, showing other approaches how the model size can be calculated. The code can be found in the Appendix A.

Concluding, the following metrics (highlighted with green background in Figure 4.2) become quantifiable using the synthetically generated dataset: Disparate Impact (DI), Feature Relevance, Clique Method, Loss Sensitivity, CLEVER Score.

4.3 Synthetic Dataset Generation

As discussed in *Related Work* (Section 3.4), existing data augmentation and generation methods do not form applicable solutions for the aim of the presented work, as they require access to the original dataset. The following Subsections describe the goal, methodology, and algorithmic implementation of the synthetic dataset generation. Two scenarios of the implementation are presented, which use information about the model's features and statistical properties in order to compute a synthetic dataset from scratch. The variations differ in the amount of input provided to create an artificial dataset - the more statistical properties are provided, the better the generator can mimic the characteristics and patterns of real data. This provides the user the possibility to decide how much information he/she feels comfortable sharing with the platform, addressing constraints such as data privacy.

4.3.1 Goal and Benefit

The availability of a synthetic dataset provides the possibility to systematically investigate the behavior of the model and to gain insights into its characteristics and behavior. For instance, by strategically examining the model's predictions across different demographic groups or protected attributes, potential biases and discrimination can be detected in the *fairness* pillar. The incomputable metric *Disparate Impact* becomes quantifiable through this approach. Further, a synthetic dataset also allows identifying the importance of different features (computation of the *Feature Relevance* metric) which improves the assessment of the *explainability* pillar. Additionally, for the computation of the *Model Size* metric, the number of features can also be extracted from the synthetic dataset. Likewise, *robustness* properties can be analyzed, such as the sensitivity to perturbations. By introducing different types and magnitudes of perturbations, the model's sensitivity and its ability to handle perturbed inputs can be measured. Depending on the fact, to which subgroup the ML model belongs, the following metrics get quantifiable: *Loss Sensitivity, CLEVER Score, and Clique Method*. Regardless of the unavailability of the original dataset used for training and testing the ML model, using the synthetic dataset for statistical probing may discover important characteristics and improve the assessment of the model's global trustworthiness by making incomputable metrics described in Section 4.2.1 quantifiable.

4.3.2 Process Design

Figure 4.3 shows an illustration of the process design and the flow of operations toward the computation of an advanced trust score. In the first step, the synthetic dataset is generated. The following Subsection 4.3.3 provides more details about what information must/may be provided to the algorithm and what respective advantages are. Subsection 4.4.1 gives insights into the algorithmic computation. In the second step, the Black Box model is used as a labeling oracle to provide labels for the generated dataset. Through this labeling process, model intrinsic characteristics are propagated onto the dataset. In a third step, this can be exploited in the metric calculation to provide a more complete and realistic picture of the *fairness, explainability* and *robustness* pillar scores. Lastly, the pillar scores are aggregated to compute the advanced trust score.

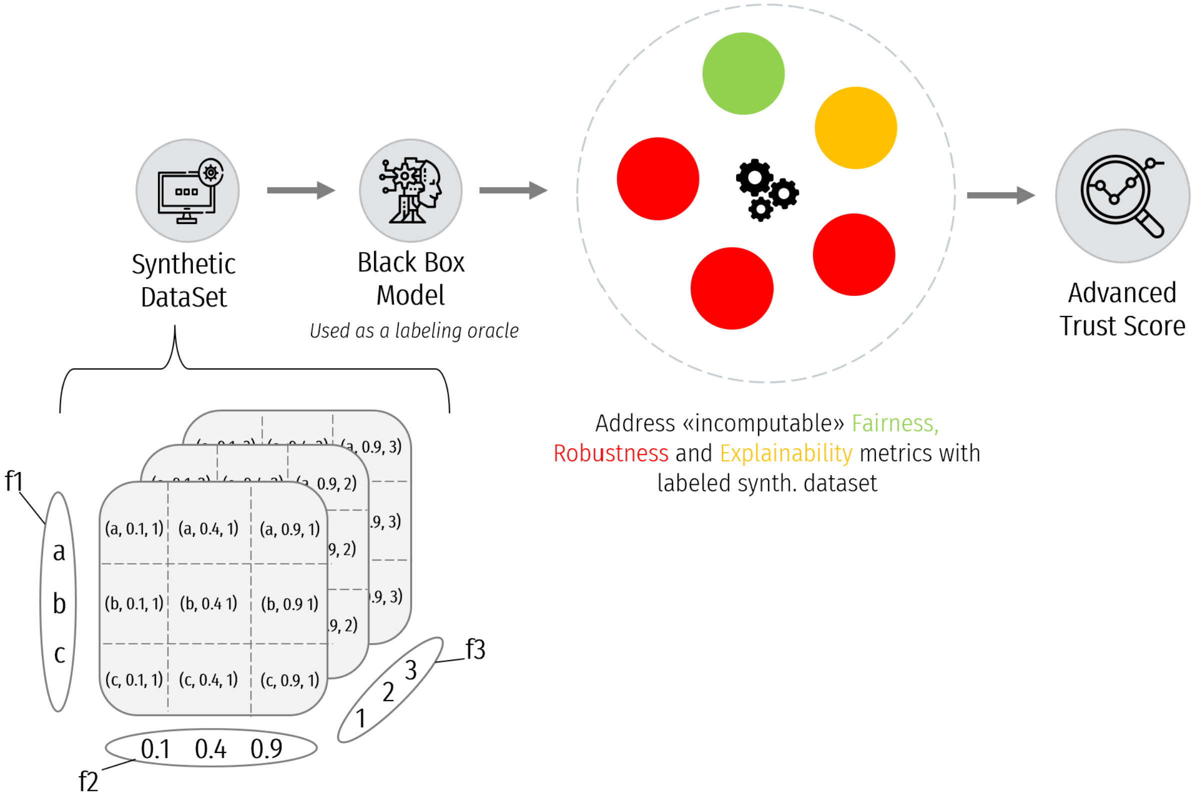


Figure 4.3: Process Design From Data Creation To Score Computation

4.3.3 Scenarios

In order to systematically engineer a synthetic tabular dataset, some information needs to be provided by the user, to prevent creating a fully arbitrary dataset. Further, the ML model requires maintaining a consistent input format, since the model is designed to handle a fixed number of features. This work presents two variations for dataset creation. Table 4.3 vertically lists statistical properties required as input for each scenario. The user of the system, who is interested in evaluating the trustworthiness of an ML model under the constraint of not sharing the original dataset, has to decide how much information describing the data is going to be shared with the platform. For the MUST scenario, mandatory information include the *Feature Name*, *Data Type*, *Minimum Value, Maximum Value*, and *Unique Values*. In the MAY scenario, the user benefits from a more accurate trust score, since the resulting dataset used for the computation of the metrics is more similar to the original dataset compared to the MUST scenario. Additionally, in the MAY scenario the statistical properties *Mean*, *Variance*, and *skew* are considered. This provides information about the shape of the distribution and its modality (multi/single modal, long tail, Non-Gaussian). The fourth central movement (*Kurtosis*) is not considered in the current implementation, but could be considered for future work.

Table 4.3: Two Variations Of The Synthetic Dataset Generator And The Required Input

	MUST (High Privacy)	MAY (High Similarity)
Feature Name	Yes	Yes
Data Type	Yes	Yes
Minimum Value	Yes	Yes
Maximum Value	Yes	Yes
Unique Values	Yes	Yes
Mean	No	Yes
Variance	No	Yes
Skew	No	Yes
Kurtosis	No	No

The challenge lies in using as few exact specifications as possible, while at the same time preserving the statistical properties of the original dataset. A clear trade-off between privacy (MUST) and similarity (MAY) can be identified. The more similar the synthetic dataset is to the original dataset, the more accurate the assessment of trustworthiness can be. However, this also allows for different types of privacy attacks, such as *inference attacks*, where information leakage from the model is exploited and sensitive information can be inferred about individual data points. By querying the model with the entire synthetic dataset, an adversary could carefully select instances to reveal information about sensitive data points. However, the success of this attack is mainly dependent on the ML model's application of privacy-enhancing techniques to mitigate such attacks (e.g. differential privacy) and less on the synthetic dataset itself. The synthetic dataset rather serves as a pool of possible instances, from which the attacker could carefully select instances. It is also notable mentioning, that the dataset creation is model-independent, and works the same for all subgroups from the Black Box Taxonomy (see Section 4.1).

4.4 Implementation

4.4.1 Algorithmic Design

The algorithmic design of the synthetic dataset generator consists of two main components: (1) the specification of ranges and (2) the generation of the dataset. Both components are implemented in a Python class called Generator. The first component is responsible for specifying the ranges of possible values for each feature. This is the systematic part of the implementation, and the decisions taken influence the properties of the synthetic dataset. The second component is the executive part, which is responsible for the actual generation of the dataset.

(1) Specification of Ranges

This component uses information provided by the user, to specify ranges for each feature, that contain the statistical properties and will be used by the second component. For this, the data type of the individual features and the specified scenario (MUST or MAY) play a crucial role. The implementation differentiates between three different data types:

- *Categorical*: This data type includes nominal values which can only be categorized as for instance ”a”, ”b”, and ”c” without the possibility to order values based on a rank. This also includes instances, when a category is described with a number, as for instance group ”1”, ”2”, and ”3”. In this example, group ”1” is not considered better or worse than group ”2”.
- *Discrete*: This data type includes natural numbers that can be categorized and ranked, as for instance the numbers on a scale from one to ten, $0 < a \leqslant 10 \mid a \in \mathbb{N}$. This often applies to counts or scores.
- *Continuous*: This data type includes real numbers that may take any possible numeric value and can be meaningfully split into smaller parts, such as decimal values and fraction. Examples are 0.25 and -412.5.

The implementation of the *categorical* feature is the simplest, regardless of the specified scenario. Because the synthetic dataset is a representation of the original dataset, each category should be present at least once. Therefore, the ranges for categorical features simply contain every unique value of the categorical feature (see line 19 in Code Listing depicting the range specification function for the MAY scenario 4.2). For the data types *discrete* and *continuous*, the implementation for specifying the range is different between the MUST and the MAY scenario. Figure 4.4 illustrates the method of range specification for features with *discrete* data types on an example. The same methodology however, can be applied also for features with *continuous* variables. The box at the top represents a histogram plot, showing the occurrences of each value from the *discrete* feature in the original dataset, together with its probability distribution function (orange line). The value span reaches from the minimum value 1 to the maximum value 15. Other statistical properties are also visible in the Figure, such as the mean with the value of 11 and the *variance* of value 1. Further, the distribution is left-skewed, since the left tail (smaller values) is much longer than the right tail (larger values). This is captured in the statistical property of a negative skew.

```python
# discrete (finite options within a defined range)
if d_type[idx].__eq__("discrete"):
    self.ranges.append(range(int(min_[idx]), int(max_[idx]) + 1, np.maximum(1, int(max_[idx] - min_[idx]) // 10)))
```

Code Listing 4.1: Specification Of Range For Discrete Features In MUST Scenario

For the specification of the range in the MUST scenario, only the minimum and maximum values are available. The Code Listing 4.1 shows the implementation, where the step size is defined as the tenth fraction of the absolute difference between the minimum and maximum value (integer division for features with *discrete* values, regular division for *continuous* features). This ensures, that the range always contains at most ten equally spaced values. This approach is visualized in the middle box of the Figure, where the step size equals to a value of 2. The final range therefore, consists of the following values: [1,3,5,7,9,11,13,15]. However, the difference in the statistical properties can be observed for the specified range of the MUST scenario, with a mean of value 8 and skew of value 0 (no skew). This is due to the methodology and the absence of this the properties in the specification of the range. For the MAY scenario, these statistical properties are used to generate a range, which better reflects the original distribution. The final range for the MAY scenario consists of the following values: [1,3,4,6,9,11,12,13,14,15]. Here, values from the upper half (8 to 15) of the total range are more often represented than values from the lower half (1 to 8). For this reason, the mean of the range for the MAY scenario is higher (equal to the mean of the original dataset) than the mean of the range for the MUST scenario. This similarity of statistical properties compared to the original dataset is also shown for the variance and skew.

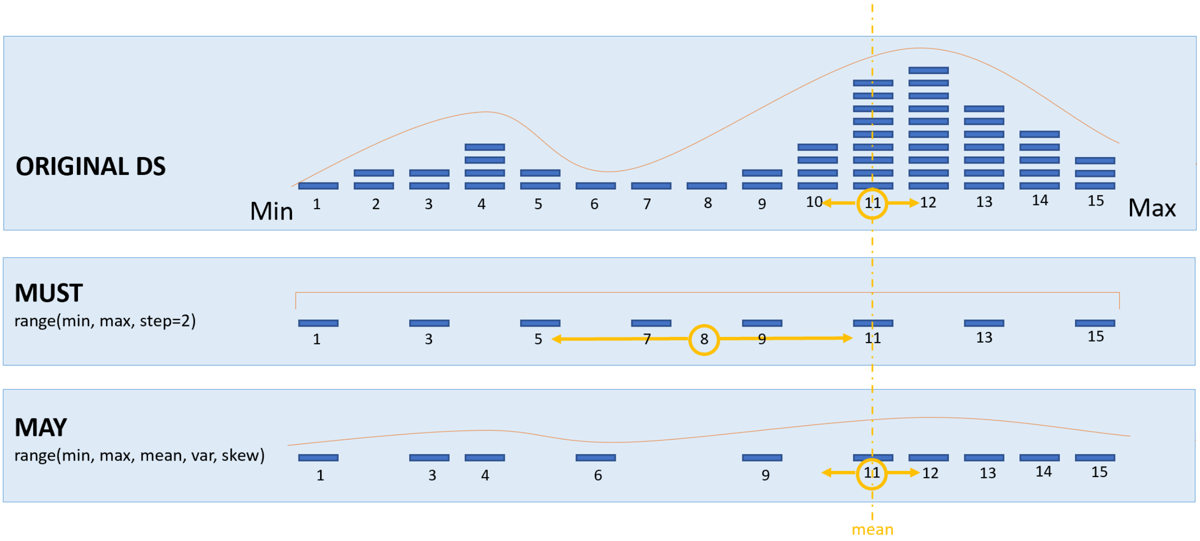


Figure 4.4: Illustration Of Range Specification For Features Of Discrete Data Type

As shown above, the ranges computed for the MAY scenario are more accurately representing the distribution of the original dataset. This raises the question, how the algorithm can decide, what discrete value should be selected for the specification of the range, and how is the distance between the values is computed, considering the fact, that the spacing is not defined by a fixed constant? The following paragraph provides an answer to this question.

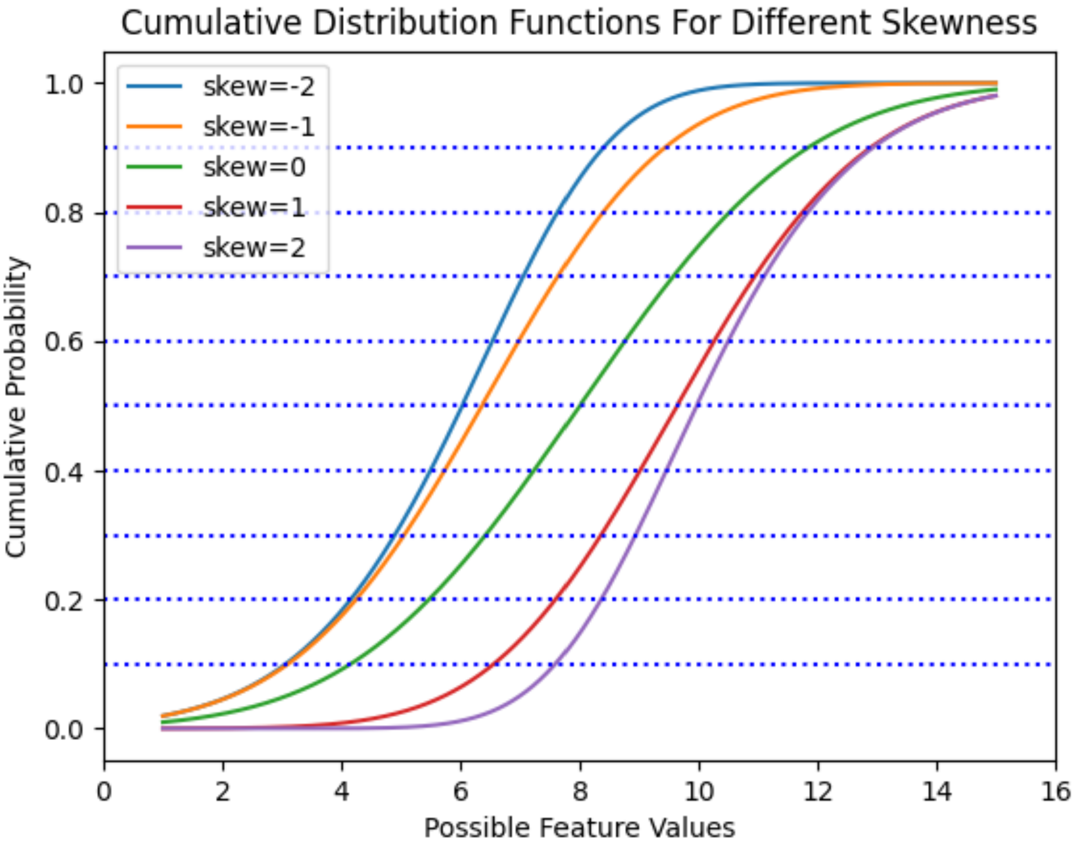


Figure 4.5: CDF Plot

The code from line 22 to 38 in the Code Listing 4.2 exemplifies the value specification on the concrete implementation. The algorithm draws a large enough number of samples from the skewed normal distribution with the provided statistical properties as arguments. These samples approximate the values from the original dataset (blue bars in top box of Figure 4.4, since they share the same statistical properties. However, it is important to mention, that it is not possible to derive at the exact values, since many possible distributions may share the same properties. Because samples may result in higher/lower values than the maximum/minimum value provided as a statistical property, the samples out of bound are replaced with the respective extrema (line 25/26 in the Code Listing). Then, the samples are sorted in ascending order which could result in an array as the following: [1,2,2,3,3,4,4,4,4,...,15,15,15]. Since each sample has the probability of $\frac{1}{\#_{samples}}$, the cumulative probability represents the accumulation of the individual probabilities. As the value of the sample in the sorted array increases, so does the value of the cumulative distribution. The summed probability over all samples equals to one. Figure 4.5 depicts different Cumulative Distribution Functions (CDFs) for different skewness values. As it can be seen, the skewness value influences, how *early* the slope of the CDF begins to increase. Therefore, the values for the cumulative probability can be used as thresholds - if reached, the value of the sample is included into the range. Because the length of the specified range should not exceed more than ten values, the thresholds are specified in an interval of 0.1, as the dashed, horizontal blue lines in the Figure indicate (line 31 in the Code Listing). Assuming the provided skew in the example equals -1, the first value of the range takes the value 3 since this is the x-value of the graph, where the y-value crosses the first horizontal line. The second value is at 4.5, which is casted to an integer value in the case of being faced with discrete values (not casted for features with continuous variables). This could result in the following range, when each unique value is only once: [3,5,6,7,8,9,10]. In this example, the range of possible values is rather small. In situations where the minimum and maximum value are wider separated, the variation of the distances in the specified range gets bigger.

```
def _specify_ranges_may(self, d_type, min_, max_, unique, mean, std, skew):
    """ Iterates over the number of features (n) and
    specifies a [List] for each feature containing
    possible values for the later dataset generation.

    Args:
        :param d_type:  [List:Strings] of size n
        :param min_, max_, unique:  [List] of size n
        :param mean, std, skew:  [List:float] of size n
                        NaN for categorical features

    Sets:
        self.ranges:  [NestedList] attribute of the
        Generator object with n Lists of different size
    """
for idx, _ in enumerate(d_type):
    if d_type[idx].__eq__("categorical"):
        # ordinal or nominal
        self.ranges.append(unique[idx])
        continue

    # draw from distribution with statistical properties
    X = skewnorm(skew[idx], loc=mean[idx], scale=std[idx]).rvs(10000)
    # replace drawn samples out of bound
    X[X < min_[idx]] = min_[idx]
    X[X > max_[idx]] = max_[idx]
    X.sort()
    c_prop = np.arange(1, len(X) + 1) / len(X)
    r = []

    for threshold in np.arange(0, 1.1, 1 / 9):
        i = (np.abs(c_prop - threshold)).argmin()
        # discrete (finite options)
        if d_type[idx].__eq__("discrete"):
            r.append(int(X[i]))
        # continuous (infinite options)
        elif d_type[idx].__eq__("continuous"):
            r.append(X[i])
        else:
            raise AttributeError("Unsupported data type provided.")

    r = list(set(r))
    r.sort()
    if int(min_[idx]) not in r: r.append(int(min_[idx]))
    if int(max_[idx]) not in r: r.append(int(max_[idx]))
    self.ranges.append(r)
```

Code Listing 4.2: Code Dataset Generation

(2) Generation of the Dataset

This component is responsible for the execution of the data generation. It uses the specified ranges from the first component, based on which a dataset is generated and saved to a CSV file. The implementation is independent of the specified scenario (MUST or MAY) and uses the Python module called *Itertools* [74]. It provides the usage of fast, memory-efficient tools for the creation of iterators. Line 27 in Code Listing 4.3 calls the *product()* function, to generate a Cartesian product of input iterables. In mathematics, the term Cartesian product is defined as a mathematical operation that combines elements from two or more sets to create a new set. The operation is denoted by the ”×” symbol. The new set of a Cartesian product between set A and set B contains ”all possible ordered pairs whose first component comes from (set) A, and whose second component comes from (set) B” [75, p. 1]. This is illustrated on a short example below:

Let $A = \{1, 2\}$ and $B = \{a, b\}$. Then the Cartesian product of A and B is written as $A \times B$ and results in a new set $A \times B = \{(1, a), (1, b), (2, a), (2, b)\}$. Note that $|A \times B| = 4 = |A| \times |B|$.

```
 1 def _generate_dataset(self, path, f_names, must):
 2     """ Generates a dataset and saves it to a CSV
 3         file at the provided path destination.
 4         This function uses the ranges saved in the
 5         self.ranges attribute.
 6 
 7     Requires:
 8         Before calling this function, the ranges
 9         have to be specified by calling the
10         _specify_ranges() function.
11 
12     Args:
13         :param path:    [String] pa
14         :param f_names: [List:Strings] names of the
15                         features in a list for
16         :param must:    [bool] used for file naming
17 
18     Returns:
19         :return         [DataFrame] synthetic dataset
20     """
21     mustormay = "must" if must is True else "may"
22     full_filename = f"{path}/synDS_{mustormay}.csv"
23     with open(full_filename, "w", newline="") as f:
24         writer = csv.writer(f)
25         writer.writerow(f_names)
26         # Write all possible combinations to the CSV file
27         for element in itertools.product(*self.ranges):
28             writer.writerow(element)
29     return pd.read_csv(full_filename)
```

Code Listing 4.3: Code Dataset Generation

For the implementation of the synthetic dataset generation, the individual sets correspond to the ranges for every individual feature, provided by the first component. Assume, that an original dataset (which is not accessible for the trustworthiness analysis) contains three features that are listed and described below, together with the specified ranges from the first component of the synthetic dataset generator:

- $f1$: The first feature is a categorical feature with the values ”a”, ”b”, and ”c”. The specified range from the first component of the implementation is simply containing each unique value: [”a”, ”b”, ”c”]
- $f2$: The second feature consists of continuous values ranging from zero to one. The specified range for the MAY scenario could look like this: [0.1, 0.4, 0.9]
- $f3$: The third feature contains discrete values from one to three. The specified range is the following: [1, 2, 3]

The ranges specified for each feature form a set, since the values in the range are only appearing once. For the generation of a synthetic dataset, the Cartesian product of all three feature ranges can be computed. This example is illustrated in Figure 4.3 in the bottom left. The formal description looks as follows:

$$F_1 \times F_2 \times F_3 = \{(f_1, f_2, f_3) \mid f_1 \in F_1,\ f_2 \in F_2,\ f_3 \in F_3\} \tag{4.5}$$

The result is a dataset containing nine samples. This value is calculated as the product of the total number of values per feature ($|A \times B \times C| = |A| \times |B| \times |B| = 27 = 3 \times 3 \times 3$). Figure 4.3 as well shows, that the newly created dataset contains for example the samples (b, 0.4, 1) and (a, 0.9, 3), which might also appear in the original dataset as such. Further, this example also nicely illustrates, that the synthetic dataset may also contain samples that are not part of the original dataset. One such example may be the sample (c, 0.1, 1) under the assumption that the second (continuous) feature of the original dataset only contained values *0, 0.2, and larger* but not the value 0.1. However, the more statistical properties are considered for the range specification, the better will the synthetic dataset approximate the original one.

4.4.2 Time Complexity Analysis

To estimate the time needed for a dataset creation and to evaluate the time complexity of the implementation, an experiment has been conducted. It is important to highlight, that real world datasets have different properties, such that it is not feasible to test every scenario. Therefore, the aim of this time complexity analysis is to provide a realistic upper bound.

As indicated in the previous Section, the size of the synthetic dataset depends on two components: first, on the *number of features* and second, on the *Range Length* of each feature. The former relies on the original dataset and can not be influenced by the design of the implementation. The latter depends on the first part of the Algorithmic Design (see Subsection 4.4.1) and can vary between different data types. Gentle reminder: the *Range Length* for categorical features equals the number of unique independent values. In the special cases of binary categorical features, the *Range Length* has a value of two. For discrete and continuous features, the *Range Length* is bounded by a maximum length of ten. The specification of MUST or MAY influences the spacing between the individual values, but not primarily the length of the range itself. Therefore, it is assumed that the average *Range Length* will lie between five and eight values.

One possible example is depicted in Table 4.4 containing four features. As shown in the last column, the *Range Length* is different and mainly depends on the data type of the feature. In this example, the synthetically generated dataset would contain 720 rows ($|F1| \times |F2| \times |F3| \times |F4| = 10 \times 2 \times 6 \times 6 = 720$).

Table 4.4: Possible Result Of The Range Specification

Feature	Data Type	Specified Range	Range Length
F1	discrete	[0,2,5,7,9,12,15,18,24,31]	10
F2	binary categorical	[0, 1]	2
F3	continuous	[1.2, 2.1, 4.4, 5.9, 8.2, 8.8]	6
F4	multinomial categorical	[a, b, c, d, e, f]	6

To provide an upper bound for the time calculation, the experiment assumes that all features have the same specified *Range Length*. Figure 4.6 depicts the Time Complexity Plot, where the x-axis shows the independent variable (*number of features*) and the y-axis shows the dependent variable (time in seconds). The colored lines illustrate the difference in the specified *Range Length* from five to eight. Regardless of the *Range Length*, the time needed to generate a synthetic dataset with seven features or fewer is tolerably low (less than five seconds). However, in all cases an exponential time increase can be identified for eight features or more. For a dataset with eight features, each having a specified *Range Length* of eight, the final dataset contains $8^8 = 16'777'216$ samples and the generation takes slightly longer than 80 seconds.

Again, this computation assumes that all features have the same *Range Length*, and serves as an upper bound. Real-world datasets however, often have features of different data types, which result in distinct *Range Lengths*. Therefore, the experiment suggests that this approach is practical for datasets with a feature size around ten.

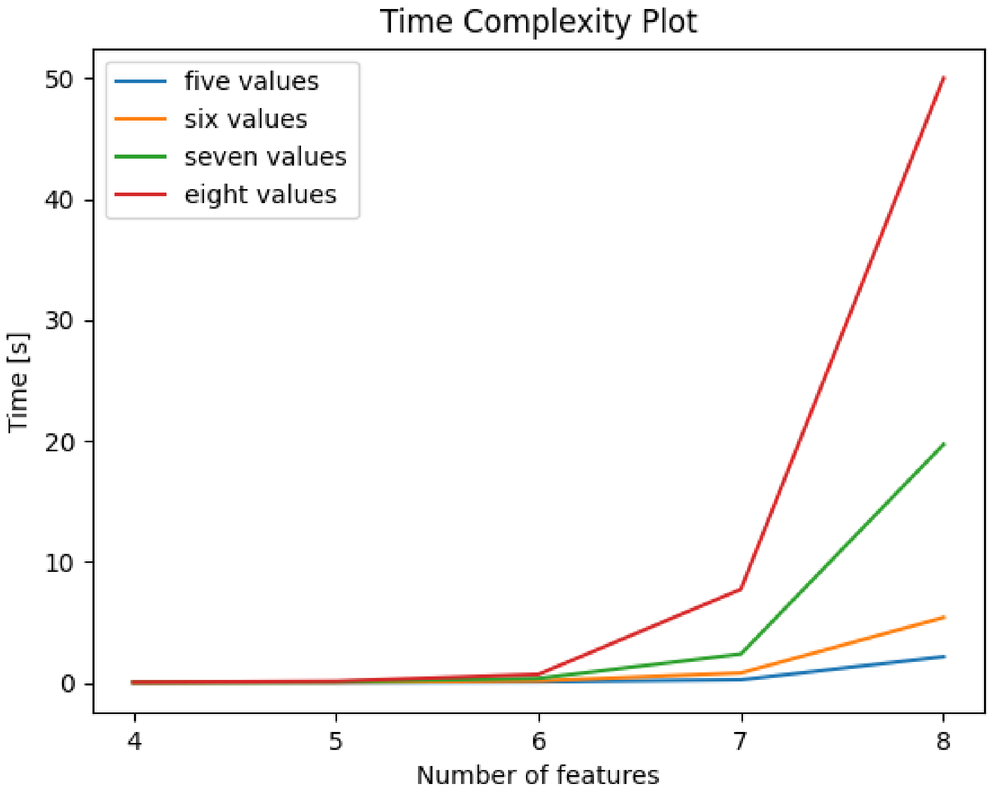


Figure 4.6: Experiment To Evaluate The Time Needed For The Dataset Generation

4.4.3 Challenges and Limitations

The creation of synthetic data is not a trivial process, and many different approaches exist that can be pursued. For the context of the presented work, access to the original dataset is not granted. This clearly limits the number of applicable methods, as for instance no existing data augmentation methods can be applied, such as the usage of GANs. As the use-case of a synthetic dataset generation from *zero* is rather specific, no existing solutions for this challenge could be identified. Therefore, the methodology of the presented implementation provides novelty as it is capable of creating a tabular dataset for two scenarios (MUST and MAY). This Subsection aims to address potential limitations.

As mentioned above, the methodology of the implementation is concentrated on generating tabular data. The methodology uses the Cartesian product to generate the dataset from different feature ranges. One property of the Cartesian product is, that the cardinality of the output equals the product of the cardinalities of all the input sets. This causes an issue for datasets with a large number of features because it would result in blowing up the size of the dataset exponentially. The Time Complexity Analysis in Subsection 4.4.2 has shown, that the dataset generation with up to eight features is relatively feasible. For datasets with more than eight features, the practicability depends on the individual data types.

Further, the size of the specified range for categorical features corresponds to the number of unique values in the original dataset. This causes problems if the dataset contains unique identifiers and a feature is used as a primary key. The number of unique values equals the size of the original dataset, which again influences the size of the synthetic dataset. However, the unique identifier does not provide useful information that can be used for the ML model and is anyway considered an irrelevant feature. Furthermore, the removal of direct identifiers avoids the possibility of re-identification of individuals and improves the privacy aspect. Additionally, the generation of exponentially large files is not only impractical regarding the aspect of file size, but also the time it takes to produce the dataset. Another limitation by design is, that statistical properties do not contain information about the relation between features. Additional input such as the feature correlation could be used to filter out *impossible* samples from the synthetic dataset, e.g. a two-year-old baby with a body weight of 65 kg. The dataset generation is tested and works properly for normalized datasets, but not for one-hot-encoded features because the information is spread across different columns. Similarly, the dataset generation works for spatial features, e.g. longitude and latitude values because they are in a decimal degree format. But for features, where the basis is not decimal as for instance time-series data, the generator is missing this information (one day has 24 hours). A solution could be to convert the time value into a universal Unix timestamp before the sample generation and convert them back again. Similar to the conversion from the time to a running total number of seconds, other feature values could be converted into the decimal system for the dataset generation.

4.4.4 Possible Integration Into Trusted AI Platform 2.0

The implementation of the synthetic dataset generator aims to address the limitation of the Trusted AI [12] platform, when faced with a scenario for which the original dataset is not uploaded to the tool. Because a new version of the platform is being developed simultaneously in independent work, the final integration of the dataset generator into the current version of the platform is not part of this thesis. However, the dataset generator is built in a modular way and can be considered an encapsulated module independent of the current implementation. The code is also written in Python, which allows simple integration into the application without the need for an API. The architecture of the application including the synthetic dataset generator could look as depicted in Figure 4.7.

The *Upload Module* in the front end could provide the user an alternative interface if the dataset is not allowed to be shared with third parties out of privacy concerns. Instead of a mandatory dataset field, the user is asked to provide descriptive information about the features of the dataset. Hereby, information from the MUST scenario is marked as mandatory fields, and additional statistical information from the MAY scenario is marked as optional (see for reference Table 4.3).

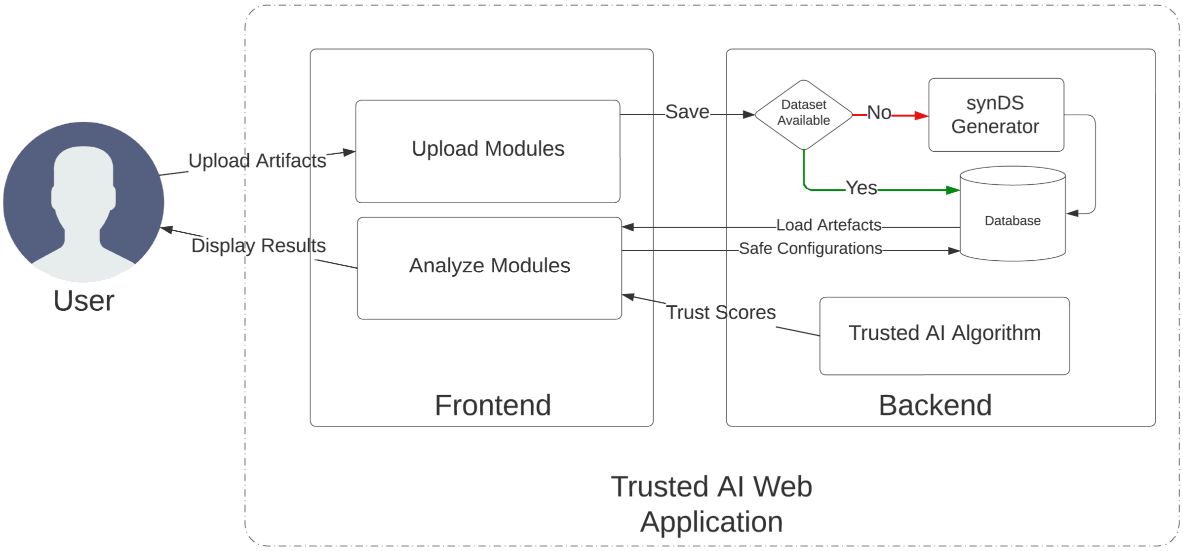


Figure 4.7: App Architecture Including The SynDS Generator, Based On [12]

Algorithm 1 shows the algorithmic pseudocode and illustrates the change in the current implementation needed, to produce a successful integration. Lines 10 to 13 show, that the current implementation does not need to be changed and the regular trust score computation can be executed. However, the algorithm first will have to check if a dataset is provided by the *Upload Module* and in if this is not the case, then in lines 3 to 8 the instantiation of the Generator object is executed, and the respective method called depending on if the user provided statistical properties for the MUST or MAY scenario.

Algorithm 1 Extended Trusted AI Algorithm

```
 1: function TrustedAI(model, $data_{train}$, $data_{test}$, factsheet, $config_{map}$, $config_{weights}$)
 2:     if $data_{train} = 0 \mid\mid data_{test} = 0$ then
 3:         generator ← Generator()
 4:         if MUST then
 5:             $data_{test}$ ← generator.generate_dataset_must($MUST_{args}$)
 6:         else
 7:             $data_{test}$ ← generator.generate_dataset_may($MAY_{args}$)
 8:         end if
 9:     end if
10:     compute metrics                          ▷ Regular Trust Score Computation
11:     apply mapping
12:     compute pillar scores
13:     compute weighted trust score
14:     return trust score
15: end function
```

4.4.5 Code and Installation

The implemented code can be found in the GitHub repository [76], together with the *ReadMe.md* file including additional instructions, specified requirements, and execution examples.

The following steps describe the installation process:

Clone the GitHub Repository

```
git clone https://github.com/dgagul/trustworthyBlackbox
cd trustworthyBlackbox
```

Create a virtual environment and install the dependencies

```
python -m venv BBenv
BBenv\Scripts\activate
pip install -r requirements.txt
```

Launch the Jupyter Notebook and execute desired scenario

```
jupyter notebook
```

Chapter 5

Evaluation

The primary objective of this Chapter is to evaluate the robustness of the trust score for trustworthiness assessment. To achieve this, this Chapter evaluates the computation of the trustworthiness level of ML and DL models in four different ways: True Score, Limited Score, Advanced Score MUST, and Advanced Score MAY. The last three ways address the assessment of trustworthiness under the restriction of having no access to the original dataset (Black Box), whereas the first score is used as a baseline for comparison. The goal is to measure the effectiveness of the implemented synthetic dataset generation and its appliance to measure a more accurate trustworthiness score under the restriction of having limited information present. The implementation is evaluated on two real-world scenarios: Diabetes Prediction using Random Forest (RF) in Section 5.1 and Home Credit Default Risk using Deep Neural Networks (DNN) in Section 5.2. Both models solve a classification problem, whereas the same model could be applied to a regression task. The selection of these two model types is based on the differences in the respective properties. As defined in the Black Box Taxonomy in Section 4.1, RF is placed in the subgroup called *Understandable Models* whereas DNN belongs to the subgroup called *Highly Complex Models*. Evaluating models from different subgroups respects the fact that some metrics are only applicable to a specific ML/DL algorithm (see Table 4.2). For each of the two scenarios, a detailed problem description is included and insights into the dataset and architecture of the used model are provided. Then, the four trustworthiness scores are computed. Finally, each scenario ends with a comparison of the four trustworthiness scores.

5.1 Scenario: Diabetes Prediction - RF

5.1.1 Problem Description

Especially in the last years, the fast-evolving technology of AI has made it useful in various aspects of health care. ML models are used for instance in analyzing medical images, drug discovery as well as in medical diagnostics and treatment [77]. However, the usage of AI in healthcare has risen ethical considerations and expressed the need for interpretable models [28]. One particular example is the diagnosis and the probability prediction of suffering from diabetes based on the patient's symptoms, medical history, and lab results. Ensuring that the diabetes prediction model generalizes well to new patient data from different populations is crucial for its responsible and trustworthy use. Therefore, this scenario is well suited for the trustworthiness assessment using the methodology of the Trusted AI platform [12]. One important aspect however, is the patient's privacy and data security. As the patient's health records contain personal and sensitive data, stakeholders are not allowed to share the information with third parties including the online tool, used for trustworthiness analysis. For these reasons, the example of diabetes prediction forms a well-suited evaluation scenario.

5.1.2 Dataset and Model

For the first scenario, the diabetes prediction dataset from Kaggle was used [78] to train an RF predictor. Based on a patient's medical history and demographics, the diabetes prediction dataset can be used to build an ML model to predict diabetes in that patient. This helps doctors develop individualized treatment programs and identify individuals at risk of developing diabetes. Researchers can also use the dataset to examine associations between different demographics, medical characteristics, and risk of developing diabetes. The presented dataset comes pre-labeled and contains patient's medical and demographic information. Table 5.1 depicts the features of the Kaggle dataset [78] that are used for the training of the RF classifier. The columns *dType, min, max, mean, std,* and *skew* describe statistical properties of the feature values and are used for the generation of the synthetic dataset.

The trained model is an RF Classifier from the scikit-learn library [79], with the parameter *n_estimators* set to 1000 and *random_state* set to 42. The model was trained on categorical features, where strings were encoded using integer dummy variables. The data was not normalized before training. The dataset was split into an 80/20 ratio for training and testing, respectively. After training, the model achieved an accuracy of 97.0%, indicating its high performance in making accurate predictions on the test data.

Table 5.1: Features Used By The RF Classifier To Predict Diabetes

Feature	Description	dType	min	max	mean	std	skew
Age	plays a significant role since older persons are more likely to be diagnosed with diabetes.	discrete	0	80	41.8	22.5	-0.05
Gender	biological sex of a person is referred to as their gender, and this can affect how susceptible they are to developing diabetes. The feature is divided into three categories: male, female, and other.	multinomial categorical	-	-	-	-	-
Blood glucose level	describes how much glucose is present in the blood at any one moment. This feature is closely linked with diabetes, as diabetes patients are often affected by a high blood glucose level.	discrete	80	300	138	40.7	0.82
Hypertension	measures if the arterial blood pressure is persistently high. Since this feature is binary, a value of 0 means that there is no hypertension, whereas a value of 1 means that the person suffers from hypertension.	binary categorical	0	1	0.07	0.26	3.23
Body mass index (BMI)	frequently used measure to determine whether the person has a healthy weight in relation to their height. According to this, a BMI under 18.5 means, that the person is underweighted, while a BMI range between 18.5 and 24.9 is defined as normal. Overweight persons have a BMI between 25 and 29.9, obese persons over 30.	continuous	10.0	95.6	27.3	6.63	1.04
HbA1c level	indicator of a person's average blood glucose level for the previous two to three months. A higher risk of getting diabetes is indicated by a higher Hemoglobin value.	continuous	3.5	9.0	5.52	1.07	-0.06
Heart disease	is also linked to a higher chance of getting diabetes. A value of 0 signifies that the person does not suffer from heart disease, a value of 1 indicates that the person has heart disease.	binary categorical	0	1	0.03	0.19	4.73
Smoking history	a known risk factor for developing diabetes. Categories include: not current, former, no info, current, never and ever.	multinomial categorical	-	-	-	-	-
Diabetes	is used as the target class for the RF prediction, labels each sample if the patient was diagnosed with diabetes.	binary	0	1	0.08	0.27	2.97

5.1.3 Trustworthiness Scores

True Score

The true score is calculated based on the original dataset and the trained RF classifier, according to the methodology presented by the work of Leupp et al. as depicted in Figure 3.5. It serves as a ground truth and is assumed to reflect the true trustworthiness score of the RF classifier. The results of the trustworthiness analysis are depicted in Figure 5.1, showing the scores of the individual metrics on top of the respective bar together with the aggregated pillar score as a ratio of the maximum (five points) at the top. In total, 18 individual metrics were computed, of which seven come from the fairness pillar, four from the explainability pillar, two from the robustness pillar, and five from the methodology pillar. The pillar scores are computed according to Equation 3.1 with the default configuration weights depicted in Figure 3.6. The fairness score of the RF classifier measures a value of 4.0/5. The explainability score shows a value of 4.4/5, robustness a value of 4.0/5, and methodology a value of 2.4/5.

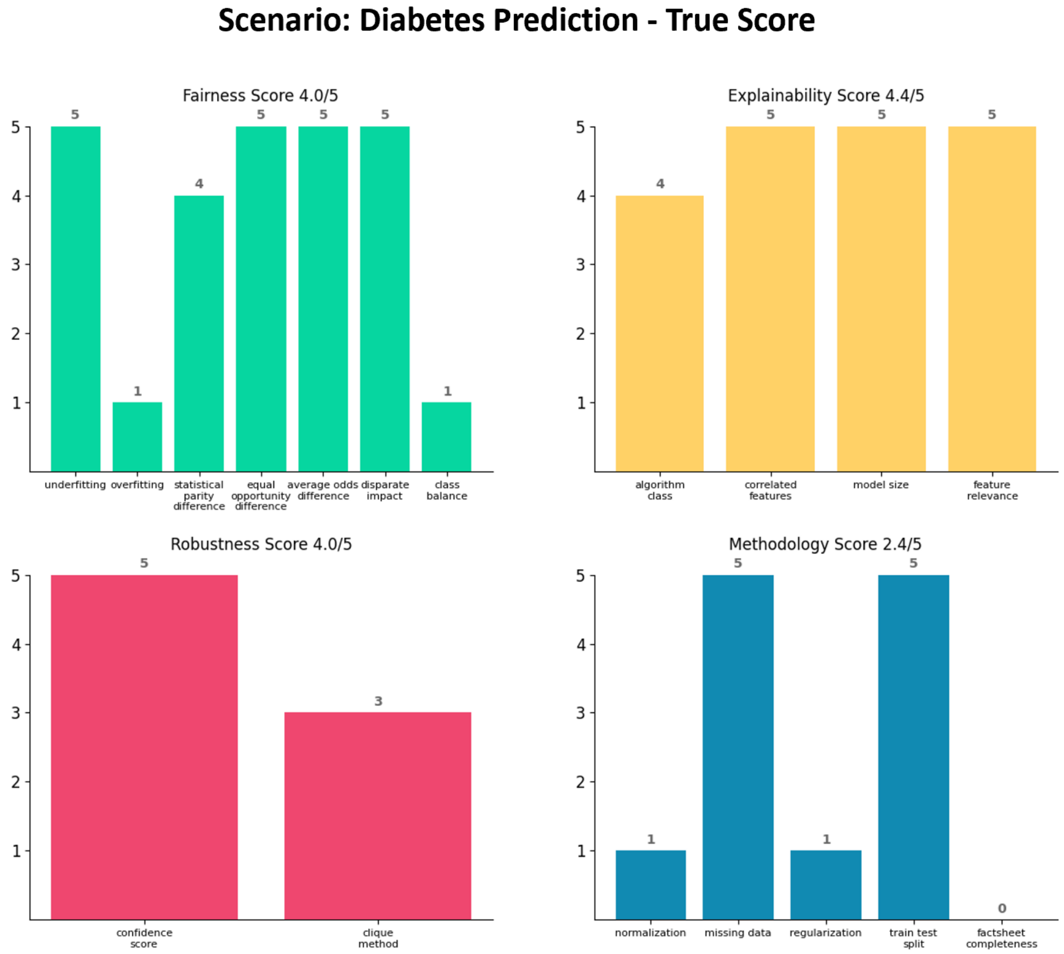


Figure 5.1: RF True Score: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

The aggregated score of the methodology pillar shows a rather low score and has a high variance regarding the individual scores. While the feature *Missing Data* and *Train Test Split* have a value of five, the metrics *Normalization, Regularization* and *FactSheet Completeness* show a low score of zero to one. This correctly reflects the attributes of the model, since for evaluation purposes the original dataset on which the RF classifier was trained, is not normalized and the model does not use any regularization method as described in the previous Subsection 5.1.2.

Limited Score

The limited score reflects the assessable trustworthiness, under the restriction that the original dataset is not uploaded by the user of the tool. The score is calculated using a limited subset of computable metrics for this scenario. In Figure 5.2, one big difference compared to the previous example can be quickly identified.

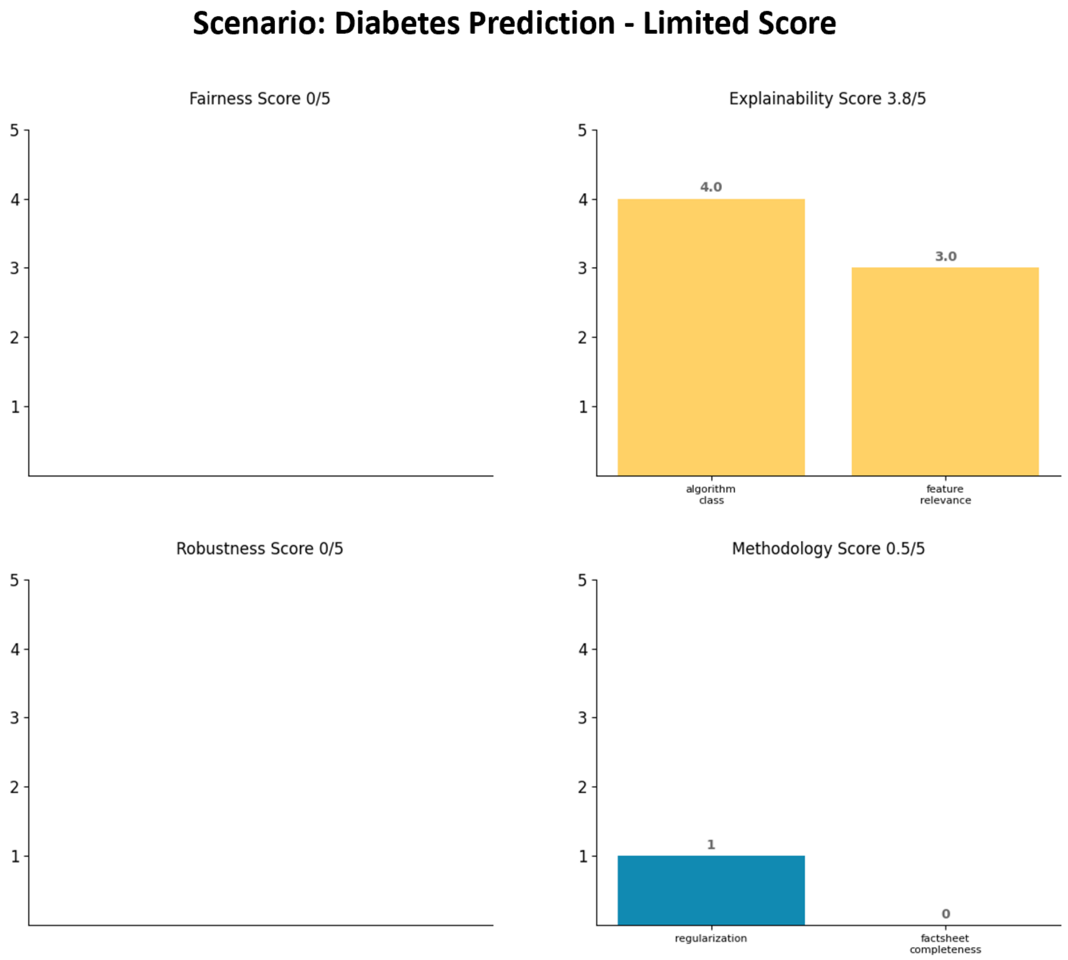


Figure 5.2: RF Limited Score: Fairness Scores (Top Left), Explainability Scores (Yellow), Robustness Scores (Bottom Left), Methodology Scores (Blue)

Both, the fairness as well as the robustness score are showing a value of zero. This is due to the fact that for these pillars, all available metrics rely on the dataset and therefore get incomputable. Also, the number of computable metrics for the explainability pillar dropped from four (in the case of the true score) to two computable metrics. Similarly, in the methodology pillar, only two metrics remain computable (compared to five in the previous example). The unavailability of computable metrics for any pillar is alarming, as the low pillar score with a value of zero is not reflecting the true pillar score and can be misleading. Further, the unavailability of metrics per pillar complicates the computation of a global trust score. The comparison of the global trust scores is provided in a separate paragraph after presenting the two advanced trust scores. The explainability score of the RF classifier measures a value of 3.8/5. Comparing it with the true score, the metrics *Model Size* and *Correlated Features* become incomputable. The *Feature Relevance* metric can not be computed using the SHAP approximation, because of the unavailability of the dataset. Therefore, the original implementation (relevance estimate by the RF model) is used, which results in a metric score of value three. The methodology score shows a value of 0.5/5. This is because the metric calculation mainly depends on the methodology applied for training the model (takes training dataset as input).

Advanced Score MUST

The advanced score MUST is calculated using the synthetic dataset generated by the module presented in Section 4.4 with only the statistical properties of the MUST scenario (see Table 4.3) provided as input. The aim is to provide a more accurate representation of the true score and improve the total number of computable metrics. The results of the trustworthiness analysis are depicted in Figure 5.3, showing the scores of the individual metrics. In total seven metrics were computed. Compared to Figure 5.2, there exists no pillar with a score of zero, due to the absence of computable metrics. Considering the fairness pillar, the metric *Disparate Impact* became quantifiable using the synthetic dataset for the metric calculation and measures a metric score of value five. Since only one metric is available, this results in a pillar score of value five according to the described methodology (see Equation 3.1). For the robustness pillar, the *Clique Method* became quantifiable and measures a value of two. Similarly to the fairness pillar, this results in a pillar score of value two for the robustness pillar. The explainability pillar consists of the metrics *Algorithm Class, Model Size* and *Feature Relevance*. Compared to the explainability pillar of the limited score, the *Model Size* metric additionally became quantifiable using the synthetic dataset. Further, the *Feature Relevance* metric, shows a higher score with a value of five, compared to the limited score (value of three), but the same value as in the True Score. Because of the availability of the synthetic dataset, this metric is also computed with the SHAP explanator. The methodology pillar shows equal results as in the limited score (both pillar score of 0.5).

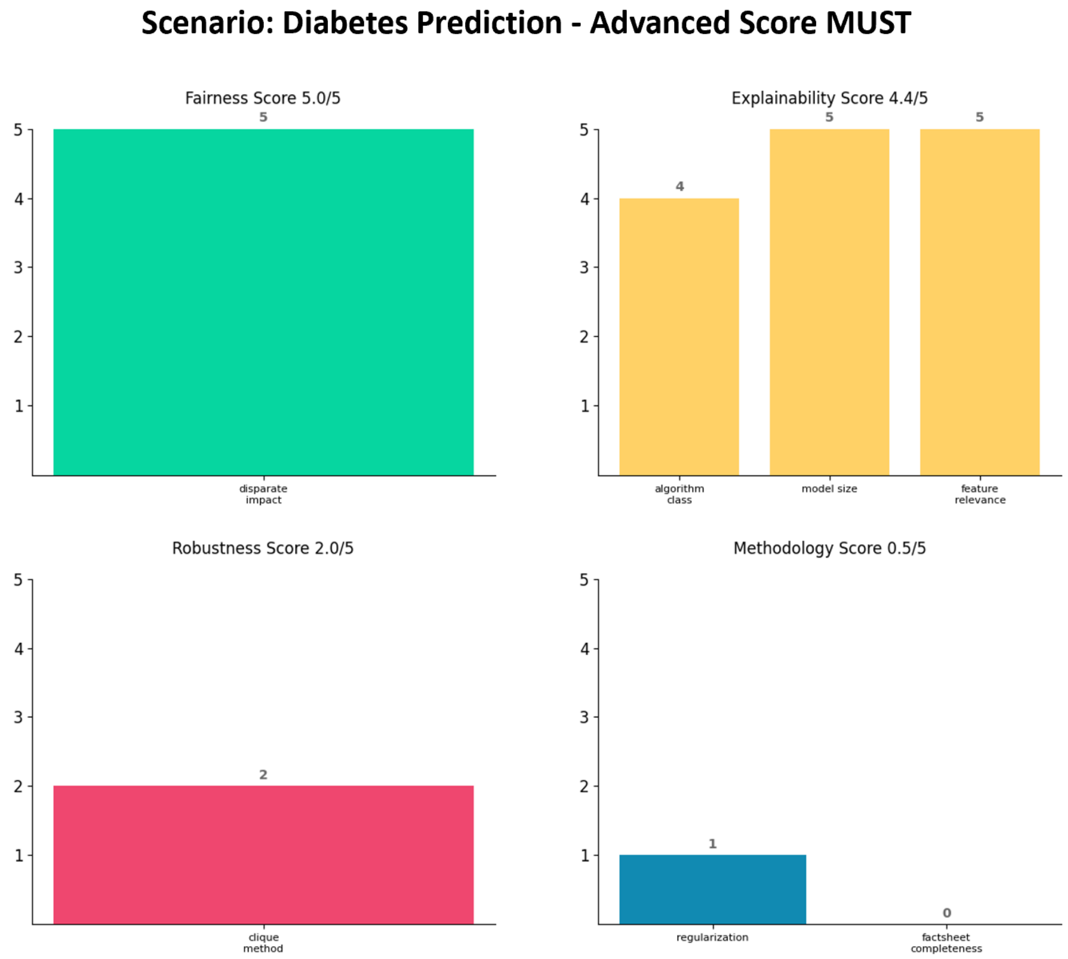


Figure 5.3: RF Advanced Scores MUST: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

Advanced Score MAY

The advanced score MAY is following the same approach of using the synthetic dataset, with the difference of using additional statistical properties of the MAY scenario (see Table 4.3) provided as input. Since this dataset more accurately reflects the original dataset, more representative metric/pillar scores are expected. The results of the trustworthiness analysis are depicted in Figure 5.4, showing the scores of the individual metrics. Similarly to the previous example (Advanced Score MUST), the same number of metrics were computed with a value of seven. This was expectable since the methodology for calculating the metrics is the same (see Figure 4.3). The only difference compared to the Advanced Score MUST is a higher robustness pillar score of value 3/5 (compared to 2/5). This reflects that the more accurate dataset leads to an improved metric score for the *Clique Method*.

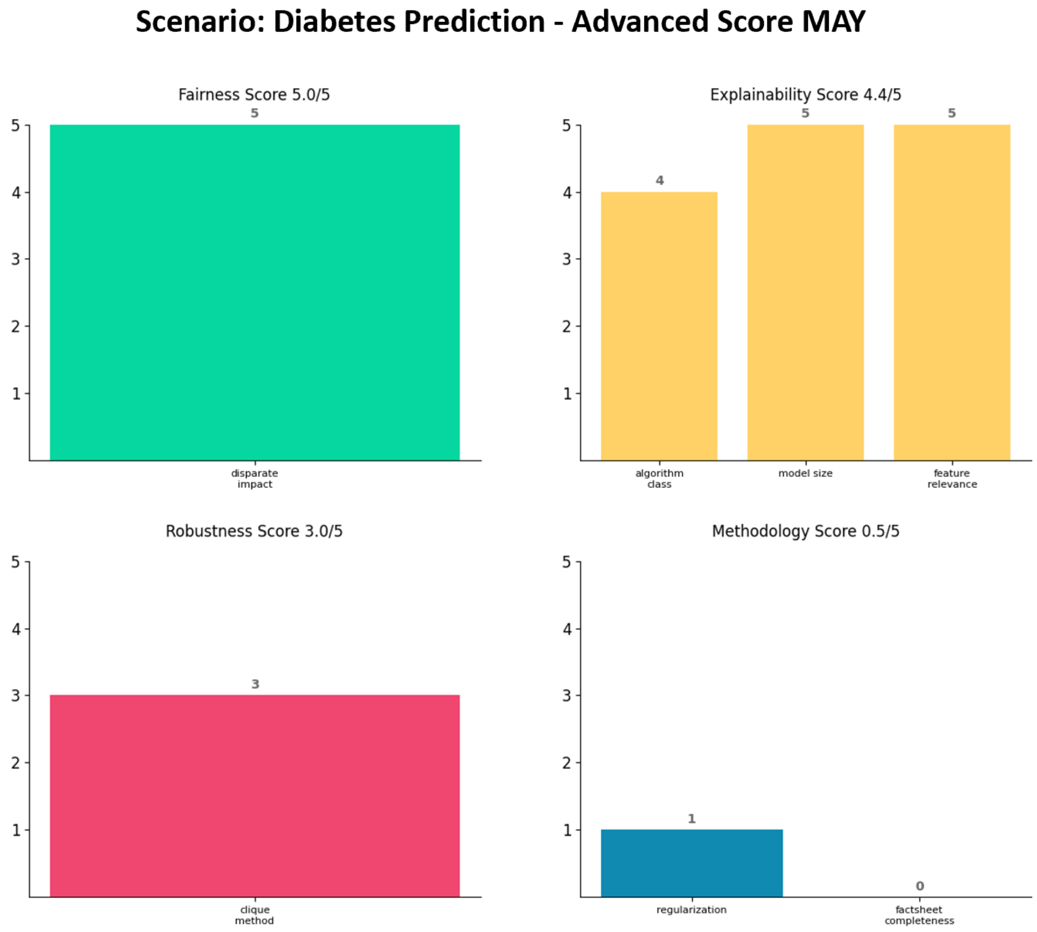


Figure 5.4: RF Advanced Scores MAY: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

Comparison

This Subsection provides a side-by-side comparison of the overall trust scores for the *True Score, Limited Score, Advanced Score MUST*, and *Advanced Score MAY*. Figure 5.5 shows a summary of the previously mentioned paragraphs. Each subplot is showing the aggregated overall score as a ratio of the maximum (five points) at the top. The individual pillar scores can be seen on top of the respective bar charts.

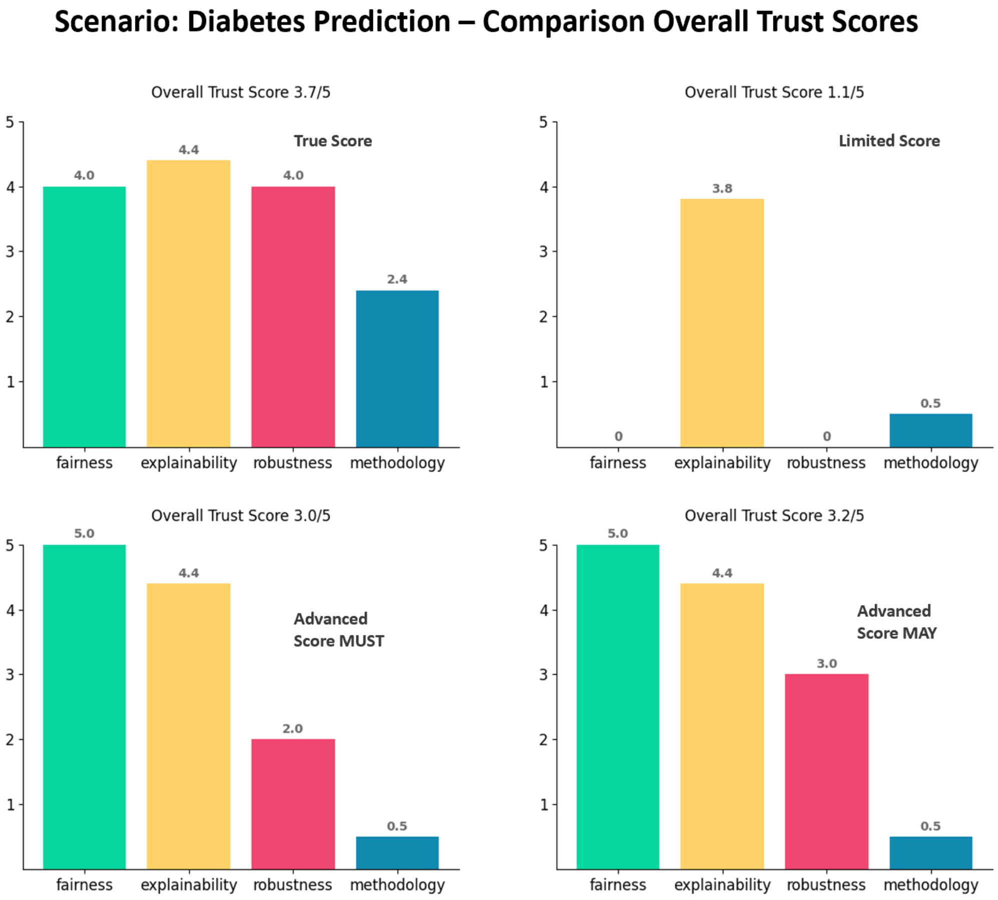
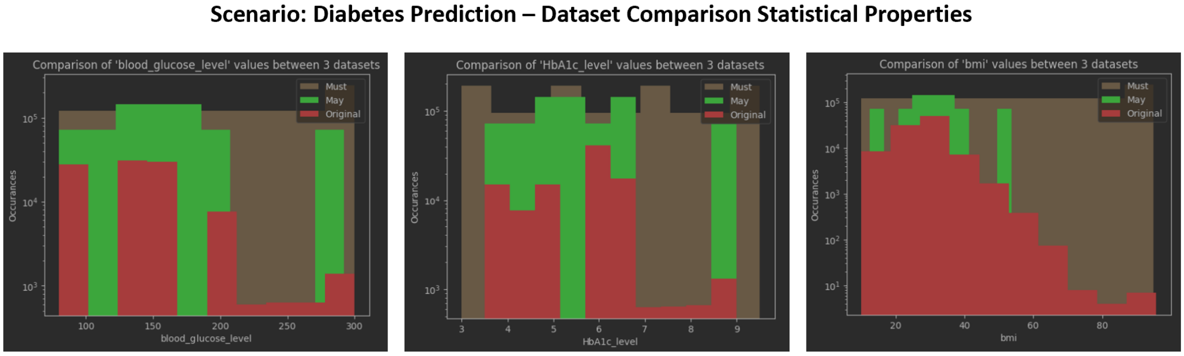


Figure 5.5: RF Overall Trust Scores Comparison: True Score (Top Left), Limited Score (Top Right), Advanced Score MUST (Bottom Left), Advanced Score MAY (Bottom Right)

The true overall score is depicted in the top left and was calculated using the original dataset. It shows a value of 3.7/5 and is assumed to reflect the true trustworthiness score of the RF classifier. In the top right, the overall score shows a value of 1.1/5 and results from the aggregation of a few computable metrics that do not rely on the dataset (as it is unavailable). While the pillar score for the explainability pillar slightly reduced (from 4.4 to 3.8), the pillar scores of the fairness and robustness pillar have dramatically dropped to a value of zero. This is due to the unavailability of any computable metric. Also, the score of the methodology pillar dropped to a value of 0.5/5. It can be concluded, that the limited score is misleading and in no way reflects the true score.

The two subplots at the bottom show the advanced scores, calculated using the generated synthetic dataset. Comparing the overall trust scores, it becomes recognizable that the availability of the synthetic dataset improves the trustworthiness assessment and provides more accurate results. The explainability pillars exactly reflect the true value of 4.4/5. The robustness score (bottom left) measures a value of two compared to the limited score (value of zero). Even better, the robustness value approximates the true score in the Advanced MAY example (value of three). This is an indication that the usage of the synthetic dataset may help to derive a more accurate assessment of global trustworthiness level. The methodology pillar looks quite different and has the same score of 0.5/5 as the limited score. This is because the metrics from the methodology pillar mostly depend on the original dataset used for training the model. The last pillar (fairness) shows interesting behavior when comparing the true score to both advanced scores. The advanced score MUST and MAY show a pillar score of value 5/5, which is higher than the pillar score in the ground truth. However, one big difference in the number of metrics used for calculating the pillar score can be identified. While the true fairness score is calculated using seven metrics, the pillar score for both of the advanced scores is computed using only one single metric. Since this single metric (*Disparate Impact*) has a value of five, the whole pillar score ends up having a high value.


Scenario: Diabetes Prediction – Dataset Comparison Statistical Properties

Comparison of 'blood_glucose_level' values between 3 datasets

Comparison of 'HbA1c_level' values between 3 datasets

Comparison of 'bmi' values between 3 datasets

Figure 5.6: Diabetes Prediction Dataset Comparison For Features: Blood Glucose Level (Left), HbA1c_level (Middle), BMI (Right)

Figure 5.6 shows the comparison of the original dataset in red, the synthetically generated dataset with the MUST scenario in brown, and the dataset with the MAY scenario in green. Three histogram plots are visible, showing the distribution of possible values together with their occurrences in the dataset. The left histogram shows the discrete feature *Blood Glucose Level*, the middle plot the continuous feature *HbA1c_level*, and the last plot on the right the continuous feature *BMI*. In all three histogram plots, the feature values of the MUST dataset are uniformly distributed, where each possible value has the same number of appearances. Comparing this to the original dataset in red, it becomes obvious that this is not the case for real-world datasets. The BMI plot on the right indicates, that the true distribution is right-skewed and most people have a lower value, whereas the number of people with a very high value is relatively small. These statistical properties are well captured by the distribution of the MAY dataset, which better follows the trend of the original dataset. Despite the aim is not to perfectly match the values of the original dataset, the Figure clearly illustrates the difference between the results.

5.2 Scenario: Home Credit Default Risk - DNN

5.2.1 Problem Description

Not only in the field of health care is the application of AI technology vastly improving standardized procedures. Also in financial services, the usage of AI has been shown to gain more attention. The technology is applied for example in fraud detection systems or to improve anti-money laundering systems [80]. Another interesting use case is the task of risk assessment. As the risk in many scenarios can be quantified, accurate risk assessment is critical to avoid potential consequences. AI models are used to assess credit risk by analyzing the historical data of a person to predict the likelihood of successful repayment of the loan. However, many people struggle to obtain credit because their loan histories are either nonexistent or not good enough. In applying ML models, the objective is to ensure that creditworthy clients are not denied access to loans [81]. The aim is to train a model, that helps to approve loans with appropriate credit amounts. Equal treatment in credit lending is crucial, even for individuals without an extensive credit history. Equal treatment is also important, considering historical gender-based discrimination. Measuring the trustworthiness of such a model helps to identify potential biases and generates awareness. Similar to the previous evaluation scenario, the privacy of applicants and their data security is very important in the financial sector. Credit lenders may not be willing to share sensitive financial records containing sensitive data. For these reasons, the example of diabetes prediction forms a well-suited evaluation scenario.

5.2.2 Dataset and Model

For the second scenario, the Home Credit Default Risk dataset from Kaggle was used [81]. The raw dataset is a dataset used for Kaggle competitions, with 122 features joined from various sources. As the joined table contains missing values and not all features are considered equally important, nine features (including the target) were used for the training of the classifier in this evaluation scenario. A description of the features together with the most important statistical properties can be seen in Table 5.2. Categorical features consisting of string values were encoded using integer dummy variables. No data normalization was applied before training. The train-test split measures a ratio of 80/20.

The trained model is a DNN Classifier from the PyTorch framework. The architecture of the model consists of one input layer with eight nodes (number of features excluding the label), one hidden layer with ten nodes, and the output layer with two nodes as the task is a binary classification problem. A cross-entropy loss function was used together with the Adam optimizer and a learning rate of 0.001. The model was trained in ten epochs with a batch size of 100. The final model achieved an accuracy of 96%. As the algorithm used for the metric calculation in the Trusted AI platform is implemented mainly for DNN models from the Keras framework, the model parameters were saved and loaded into a Keras model with the same architecture.

Table 5.2: Features Used By The DNN Classifier To Predict Credit Risk

Feature	Description	dType	min	max	mean	std	skew
Amt credit	the total credit amount that is asked for by the applicant.	continuous	4.5e+4	4.0e+6	5.9e+5	4.0e+5	1.23
Amt income total	indicates the applicant's yearly income and plays an important role for the financial solvency, strongly right-skewed.	continuous	2.5e+4	1.1e+8	1.6e+5	2.3e+5	3.9e+2
Family members	describes the number of family members of the applicant. This feature directly impacts the financial responsibilities and expenses or financial obligations.	discrete	1	20	2.1	0.91	0.98
Own car	flag indicator that marks the ownership of a car.	binary categorical	0	1	0.34	0.47	0.67
Own realty	flag indicator that marks the ownership of a realty. This feature reduces the applicant's risk since they can sell the property to recover the debt (a form of collateral).	binary categorical	0	1	0.3	0.46	0.84
Gender	indicates the gender of the applicant.	multinomial categorical	-	-	-	-	-
Education type	is linked to a higher chance of individual employment opportunities and earning potential. This feature differentiates between five categories ranging from secondary to academic degree.	multinomial categorical	-	-	-	-	-
Region rating client	a score based on the geographical location of the residence. Measures the overall economic health of the region. The feature contains a score from one to three.	discrete	0	3	2.05	0.5	0.08
Target	is used as the target class for the DNN prediction, labels each sample if the applicant credit request was granted.	binary categorical	0	1	0.08	0.27	3.07

5.2.3 Trustworthiness Scores

True Score

Using the original dataset and the DNN classifier, applicable metrics were computed for each pillar. Figure 5.7 shows the representing values, with a fairness score of 4.1/5, an explainability score of 2.2/5, a robustness score of 3.0/5, and a methodology score of 2.4/5. Compared to the previous evaluation scenario (diabetes prediction using RF), the individual methodology metrics show the same values. This is expected, as the methodology for the training of the model is consciously kept the same. In total, 19 individual metrics were computed, of which seven come from the fairness pillar, four from the explainability pillar, three from the robustness pillar, and five from the methodology pillar.

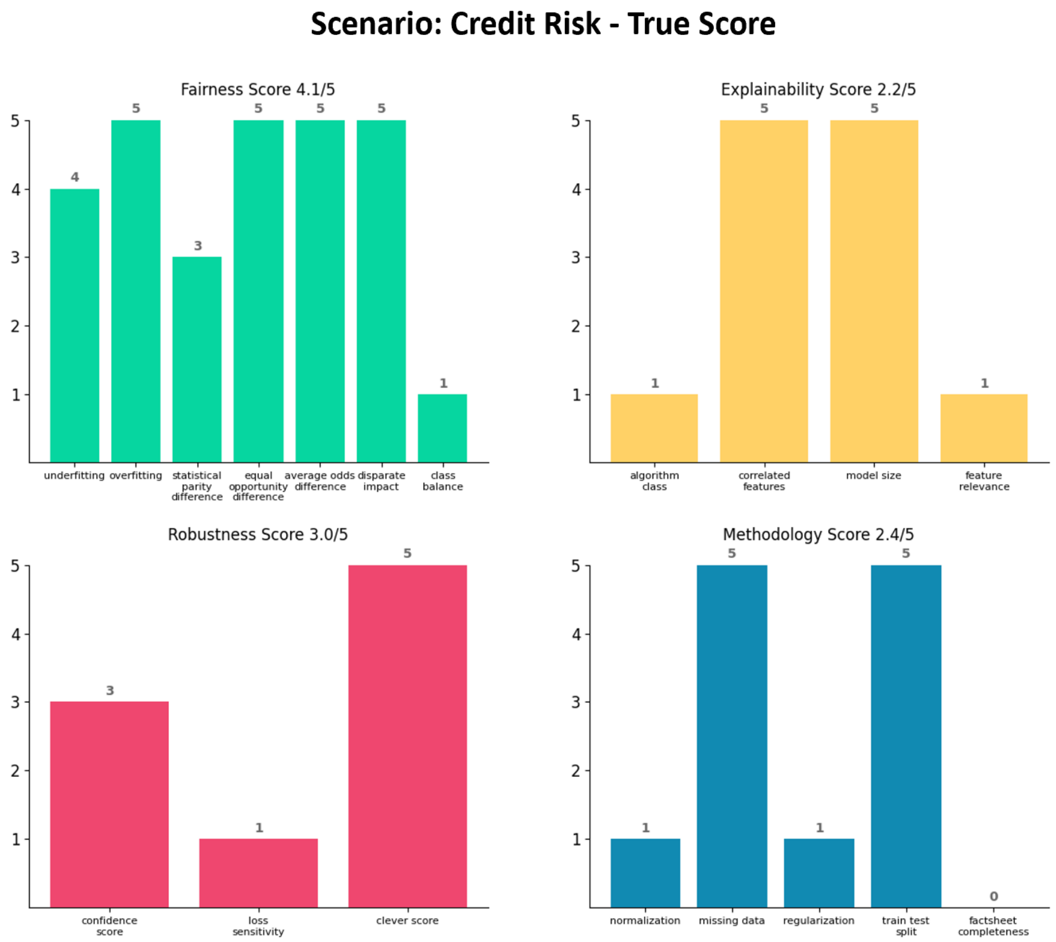


Figure 5.7: DNN True Score: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

Looking at the explainability pillar, the *Feature Relevance* metric is computed using the original dataset and the SHAP explanator presented in Subsection 3.2.3, as DNNs do not provide the possibility to extract the values for feature importance. The *Algorithm Class* metric of the explainability pillar indicates, that the model used in this evaluation scenario (DNN) is considered less interpretable compared to the previous evaluation scenario (RF) and therefore has a lower metric score with a value of one. Further, two additional metrics from the robustness pillar are applicable to this scenario. The *Loss Sensitivity* measures a score of value one, and the *CLEVER Score* a value of five. Again, the scores of the next paragraphs will be compared to the described true scores from this paragraph.

Limited Score

Regardless of the applied model, Figure 5.8 again indicates that numerous metrics get incomputable, as most rely on either the training or testing dataset. Once more, the fairness and robustness pillars have a value of zero as for these pillars, not a single metric can be computed. Comparing the pillar scores to the values from the true score, without exception, all values have dropped significantly. The number of computable metrics in the explainability pillar dropped from four to only a single metric. In general, this example demonstrates the flawed overall impression of the model's trustworthiness in the limited example.

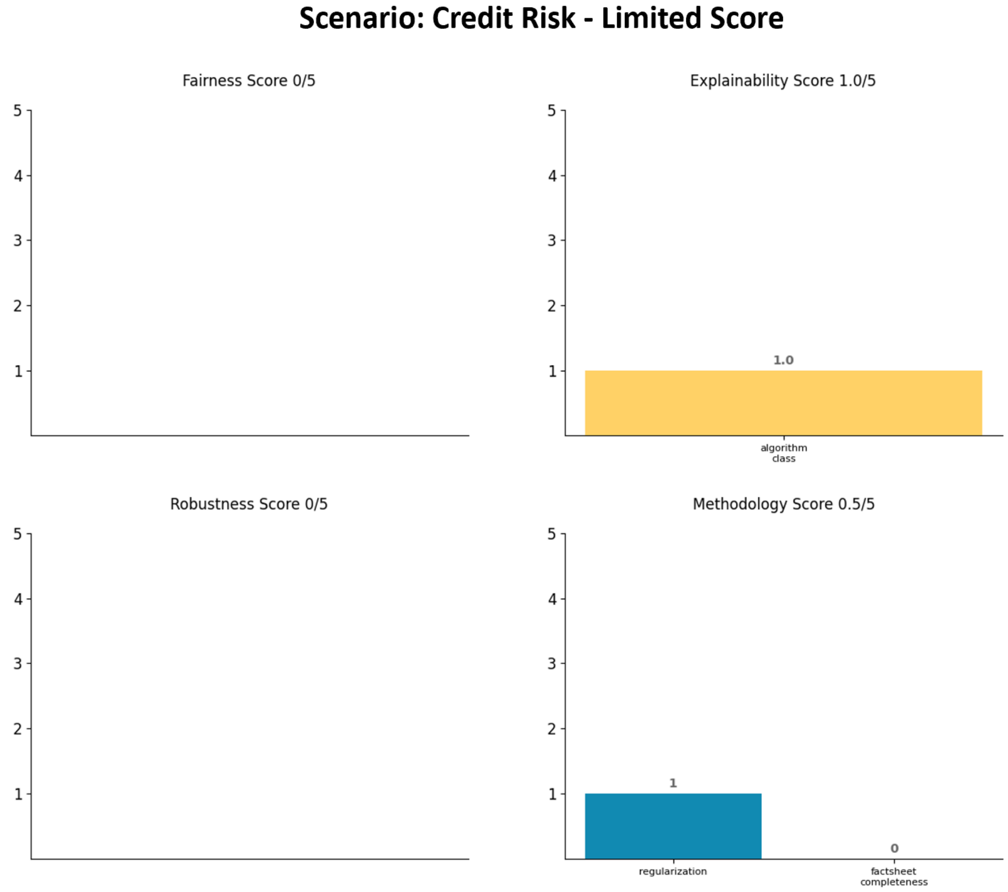


Figure 5.8: DNN Limited Score: Fairness Scores (Top Left), Explainability Scores (Yellow), Robustness Scores (Bottom Left), Methodology Scores (Blue)

Advanced Score MUST

Figure 5.9 shows the improved trustworthiness assessment, compared to the limited scenario. The synthetically generated dataset was created with statistical properties from the MUST column in Table 4.3. This way, five additional metrics become quantifiable, which results in a total number of eight computable metrics. Also, for every single pillar, at least one metric can be calculated. The value of the *Disparate Impact* metric from the fairness pillar shows the same value as in the true score. The same is true for the metrics *Loss Sensitivity*, *CLEVER Score*, *Model Size*, and *Feature Relevance*. This results in more accurate pillar scores, with a value of 5/5 for the fairness pillar, 1.7/5 for the explainability pillar, 3.0/5 for the robustness pillar, and 0.5 for the methodology pillar.

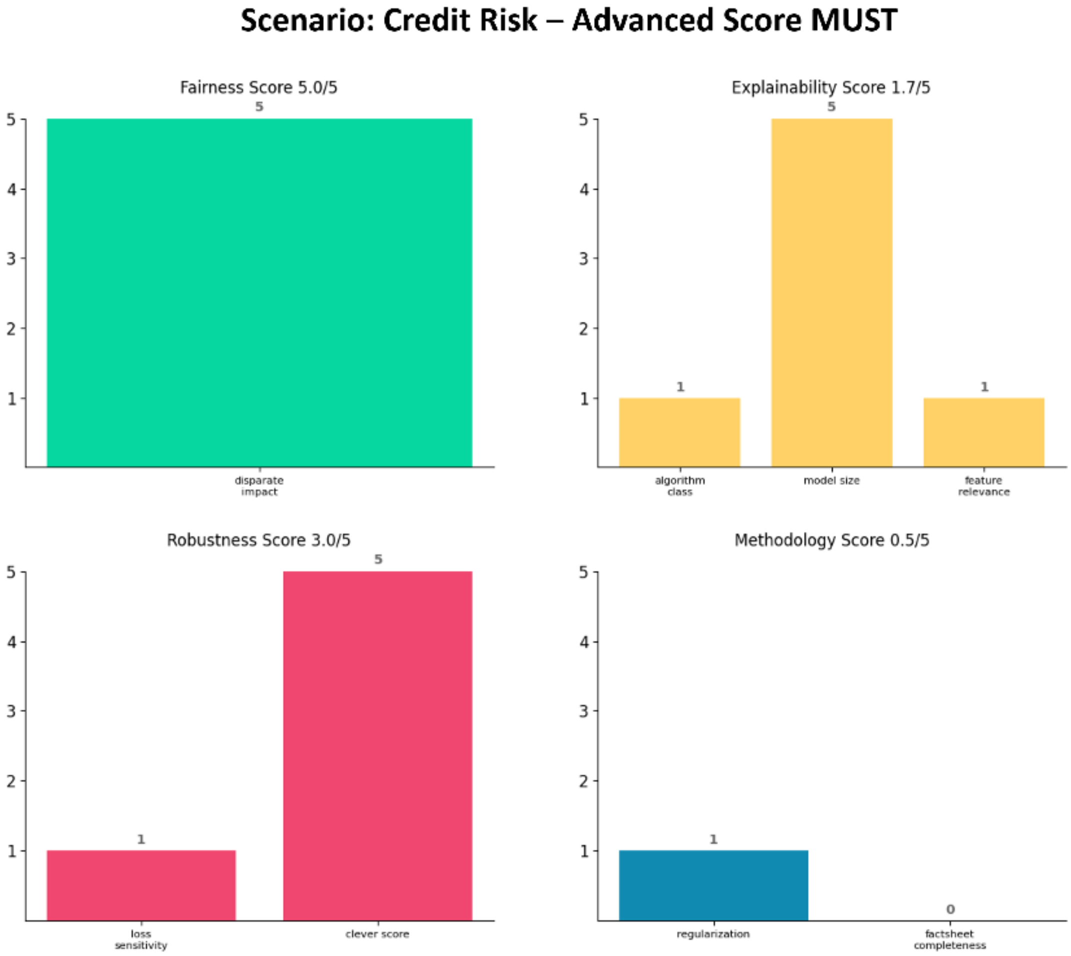


Figure 5.9: DNN Advanced Scores MUST: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

Interestingly, the fairness pillar score takes higher values compared to the true scores. Again, this is influenced by the methodology how the pillar scores are computed (see Equation 3.1). The weighted sum of a single (high) metric in the fairness pillar results in the same (high) value for the pillar score. For the robustness pillar, the *Confidence Score* metric is not computable for the Advanced Score MUST. Because the value of this metric in the True Score equals the average pillar score, it does not influence the weighted sum when becoming incomputable.

Advanced Score MAY

Figure 5.10 shows the results of the trustworthiness analysis using the synthetic dataset generated with properties from the MAY column in Table 4.3. All metric scores show the same value as in the Advanced Score MUST. A reason for this may be that the methodology of the dataset generation did not influence the computation of the individual metrics. However, the MUST dataset was sufficiently accurate to calculate the additionally quantifiable metrics, because they all show the same values as the true scores *Disparate Impact, Loss Sensitivity, CLEVER Score, Model Size* and *Feature Relevance*. Therefore, the MAY dataset could not have possibly improved the result. The next paragraph is providing a detailed comparison of the overall trust scores and the statistical properties of the corresponding datasets.

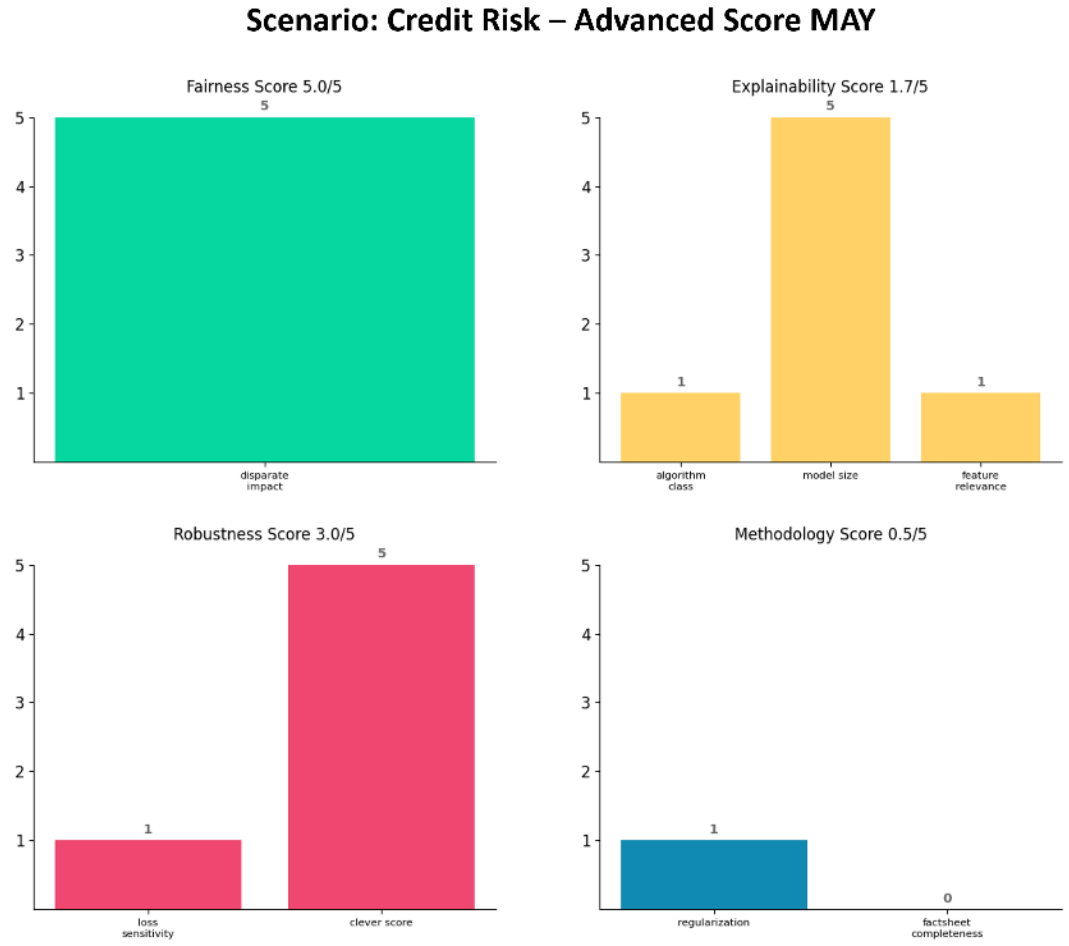


Figure 5.10: DNN Advanced Scores MAY: Fairness Scores (Green), Explainability Scores (Yellow), Robustness Scores (Red), Methodology Scores (Blue)

Comparison

Figure 5.11 shows a side-by-side comparison of the scores from the previous paragraphs. The true overall score is assumed to reflect the true trustworthiness level of the DNN classifier, with a value of 2.9/5. Similar to the previous evaluation scenario, the limited overall score (top right graph) is far off with a value of 0.4/5 and not representing the true score at all. Again, the advanced scores (bottom graphs) do more accurately reflect the true trust score (top left graph) with the same overall trust score with a value of 2.6/5. Different from the previous evaluation scenario, all scores show the same values between the Advanced Score MUST (bottom left) and Advanced Score MAY (bottom right). The reason for this result is, that the metrics in the MUST scenario already measure the same values as compared to the ground truth. Therefore, the synthetic dataset of the MUST scenario was good enough and the MAY dataset could not have improved the values further.

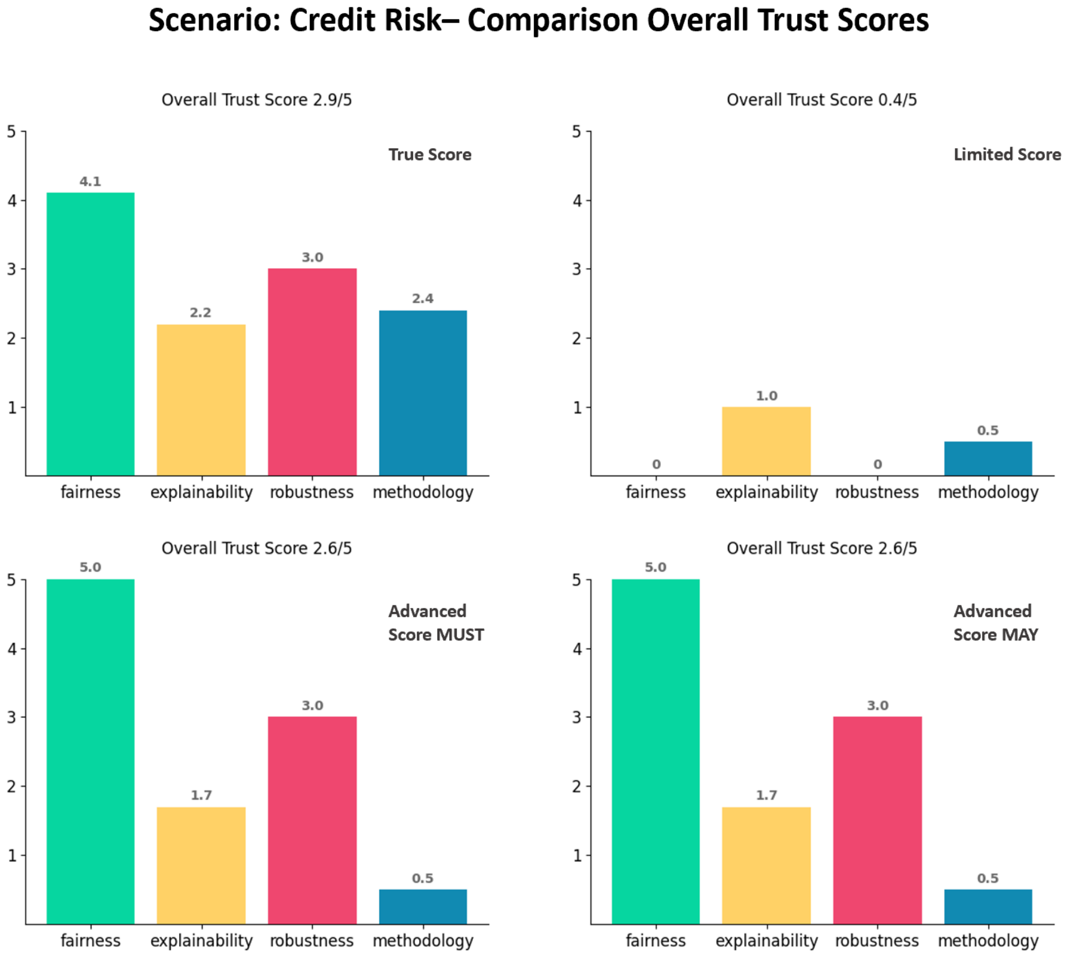


Figure 5.11: DNN Overall Trust Scores Comparison: True Score (Top Left), Limited Score (Top Right), Advanced Score MUST (Bottom Left), Advanced Score MAY (Bottom Right)

Figure 5.12 depicts six histogram plots showing statistical properties of features from the original dataset (red), the synthetic datasets MUST (brown), and MAY (green). The top row shows two continuous features (left and middle plots) and one discrete feature (right). The bottom row shows three categorical features, with each a different amount of possible categories (two, three, and five). All three features in the top row show a right-skewed distribution, which can also be seen in the last column of Table 5.2, where *Amt income total* (top middle plot) has the highest skew with a value of 390. Here also the difference between the MUST and MAY datasets is visible. The MAY dataset nicely captures this and other statistical properties, such as the mean and standard deviation, whereas the frequency of the MUST dataset always shows a uniform distribution. On the contrary, the difference is not visible for categorical features in the bottom plots. Here, all possible values within a dataset show the same number of occurrences. As the red bars in the *Name Education Type* plot on the right show, this is not true for the original dataset, as the value of zero (lower secondary) occurs the most, whereas the value of five (academic degree) occurs the fewest. This is also mentioned as a limitation of the methodology of the dataset generation in Section 3.3.

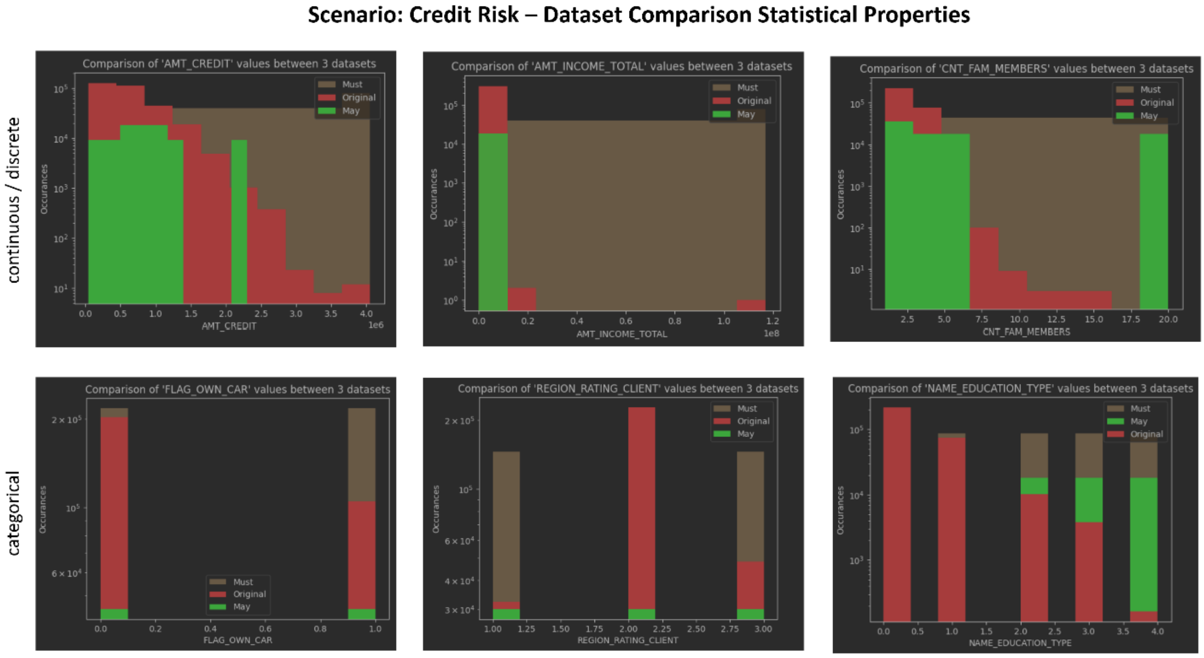


Figure 5.12: Credit Risk Dataset Comparison For Features: Blood Glucose Level (Left), HbA1c Level (Middle), BMI (Right)

5.3 Discussion

The two evaluation scenarios show that the use of a synthetically generated dataset improves the quality of assessing the model's trustworthiness with existing tools, under the limitation of having missing information available (no access to the original dataset). In both evaluation scenarios, the number of quantifiable metrics increased using the synthetic dataset (advanced scores). Further, for every pillar at least one metric became quantifiable in the advanced scores, which avoids a limited pillar score of value zero due to the lack of computable metrics. However, there are still metrics that can not be computed despite using the synthetic dataset. These are metrics that either rely on the original training dataset or the availability of true target labels (used for calculation of TPR and FPR). However, it has been shown that the approach can be used to gather information about the model by using it as a labeling oracle for the synthetic dataset and analyzing the predictions. This approach clearly improves the trustworthiness assessment and provides more accurate scores when compared to the limited scenario.

Another interesting aspect to discuss is the trade-off between privacy and accuracy, which requires careful consideration. Both dataset-generation approaches are beneficial when dealing with sensitive data or complying with privacy regulations. On one side, the synthetic dataset generator in the MUST scenario takes fewer input properties and offers higher privacy protection since it discloses limited information about the original dataset. On the other side, the dataset generation with more statistical properties in the MAY scenario results in a closer representation of the original dataset, which allows for a more reliable assessment of the ML model's trustworthiness as shown in the first evaluation scenario (True Score: 3.7, Limited Score: 1.1, Advanced Score MUST: 3.0, Advanced Score MAY: 3.2). However, because the MAY dataset is a more accurate approximation of the original dataset, it contains more sensitive information (statistical properties) that can be leaked and becomes vulnerable to potential inference attacks. These are privacy attacks in which an adversary tries to infer sensitive information about individuals in the original dataset by analyzing the predictions of the synthetic dataset [82]. Therefore, it is important to keep the dataset safe and only accessible to authorized users of the tool and apply mechanisms when storing the dataset, such as for example data encryption.

The comparison between the different datasets (original, MUST, and MAY) confirmed a successful implementation of the dataset generator and pointed out their differences with respect to the statistical properties. Further, the comparison reflects the strengths and limitations of the used methodology and supports the reasoning and interpretation of the individual trust scores. On one side, the generator proved to be capable of creating MUST and MAY datasets, that showed clear differences for discrete and continuous features. On the other side, the comparison has shown that the statistical properties between the MUST and MAY datasets look the same for categorical features. Regardless of the specified scenario, every possible category of the feature shows the same number of occurrences within a dataset. For instance, in the second evaluation scenario, the MUST dataset has the same number of fictive people that own a car, as people that do not own a car. The same is true for the MAY dataset. For features with more than two categories, the total number of samples is equally divided amongst each category. In future work, the distribution of categorical features could be integrated in the MAY scenario to improve the synthetic dataset's representation of the original dataset. One possible statistical property could be the distribution in the form of a *ratio*, which indicates the fraction of people that belong to a specific category divided by the total number of people in the original dataset. This information could be used to generate a synthetic dataset that reflects the category distribution of the original dataset. Another possible improvement of the current implementation could be the inclusion of statistical properties such as correlation (Pearson, Kendall rank, Spearman, Point-Biserial) between features, to prevent generating unrealistic samples. As an example, the diabetes dataset shows a low positive Spearman correlation between the feature *Age* and *BMI* with a value of 0.351. Having this information, *unrealistic* samples could be excluded from the synthetic dataset, e.g. an 80-year-old man (maximum value) with a BMI of 10.01 (minimum value). Considering this additional information into the MAY dataset generation would enhance its similarity to the original dataset and likely improve the accuracy of the metric calculation. This and other possible enhancements could be a valuable step to take in potential future work.

Further, a shortcoming in the representation of the pillar scores was identified, as the current implementation does not take into account how many metrics were computed. This not only affects the comparison of the True Score to the Limited Score, but also the comparison of ML models that belong to different subgroups in the Black Box Taxonomy, e.g. the comparison of a DT and a DNN classifier. In this example, the number of available metrics may be different because f.i. DT provides a form of feature relevance (higher nodes in the tree) but DNN does not. In both evaluation scenarios, the fairness pillar showed a higher value in the Advanced Scores, than in the True Score, ignoring the fact that the number of quantifiable metrics decreased from seven to one metric. This forms an issue, as the trust score of a pillar computed by only one metric is not as reliable/confident as a trust score computed by multiple metrics. Incorporating multiple metrics provides a much completer view on the pillar than one metric is able to provide. Thus, the absence of scores (incomputable metrics) decreases the confidence, as they could have impacted the aggregated score in either direction (positive or negative). One possible solution for the comparison could be, to provide a confidence interval (CI) for the model with fewer computable metrics. The range of the CI depends on the difference in computable metrics when comparing two models. Equation 5.1 shows the formal notation of such a CI:

$$\begin{gathered}
CI_{lower} = \sum_{i=1}^{n} \frac{w_{i,pillar} * score_{i,pillar}}{\sum_{i=1}^{n} w_{i,pillar}} * \frac{\#_{computable}}{\#_{total}} \\
CI_{upper} = \sum_{i=1}^{n} \frac{w_{i,pillar} * score_{i,pillar}}{\sum_{i=1}^{n} w_{i,pillar}} * \frac{\#_{computable}}{\#_{total}} \quad + \quad 5 * \frac{\#_{incomputable}}{\#_{total}}
\end{gathered} \tag{5.1}$$

The lower bound of the CI assumes that in the worst-case scenario, all incomputable metrics would have shown the minimum value of zero. It is computed by multiplying the pillar score with a Punishment Factor (PF). The PF is a fraction of the number of computable metrics divided by the total number of all possible metrics. The upper bound makes the assumption, that in the best-case scenario, all incomputable metrics would have received the maximum score of five. Therefore, the computation of the upper bound adds a product to the lower bound. The product is the multiplication of the value five (highest score possible) and the fraction of incomputable metrics divided by the total number of all metrics. From this definition results, the higher the ratio of computed metrics, the more confident and narrower the CI. Because one single value is needed for the computation of the overall trust score, the mean of the lower and upper bound would be a representative value. Returning a value above the mean would represent a rather optimistic value of trustworthiness, which is not desired since it could build false trust in a system and could lead to worse consequences. Table 5.3 shows an overview of the pillar scores before and after introducing the CI, to illustrate the proposed solution.

Table 5.3: Comparison Of The Pillar Score Before And After Introducing The Confidence Interval (CI)

			Fariness	**Explainability**	**Robustness**	**Accountability**	**Overall Score**
Diabetes Prediction	**True**	before	4.0	4.4	4.0	2.4	**3.7**
		PF	7/7	4/4	2/2	5/5	
		CI	4.0 - 4.0	4.4 - 4.4	4.0 - 4.0	2.4 - 2.4	
		after	4.0	4.4	4.0	2.4	3.7
	Limited	before	0.0	3.8	0.0	0.5	**1.1**
		PF	0/7	2/4	0/2	2/5	
		CI	0.0 - 5.0	1.9 - 4.4	0.0 - 5.0	0.2 - 3.2	
		after	2.5	3.2	2.5	1.7	2.5
	MUST	before	5.0	4.4	2.0	0.5	**3.0**
		PF	1/7	3/4	1/2	2/5	
		CI	0.7 - 5.0	3.3 - 4.6	1.0 - 3.5	0.2 - 3.2	
		after	2.9	4.0	2.3	1.7	2.7
	MAY	before	5.0	4.4	3.0	0.5	**3.2**
		PF	1/7	3/4	1/2	2/5	
		CI	0.7 - 5.0	3.3 - 4.6	1.5 - 4.0	0.2 - 3.2	
		after	2.9	4.0	2.8	1.7	2.9

The fairness pillar score in the Advanced Score MUST shows a higher value (five) than in the fairness pillar of the True Score (four) *before* the introduction of the CI. However, the ratio of computable metrics in the MUST is lower (1/7) compared to the True Score (7/7). Therefore, the computed metric score in the MUST does more influence the final pillar score than the same metric score does in the true scenario. Applying Equation 5.1, this is represented in low confidence and wider CI, ranging from 0.7 to 5.0. Computing the average from the CI boundaries results in a more representative pillar score with a value of 2.9.

The pillar score of the Advanced Score MUST and the True Score both show the same value of 4.4 in the explainability pillar. Again, fewer metrics were computed in the MUST scenario (3/4). Compared to the fairness pillar, the difference of computable metrics is smaller in the robustness pillar. Because only one single metric becomes incomputable, there are fewer metrics that could have impacted the pillar score (positively or negatively), and the CI is more narrow ranging from 3.3 to 4.6. The average results in a more representative pillar score of 4.0. The just described methodology is only one possibility to improve the comparison of aggregated scores with a different amount of computable metrics. Other approaches can be explored in future work.

Chapter 6

Final Considerations

6.1 Summary

Black Box Machine and Deep Learning systems are increasingly used and applied in various applications. There exists an essential need to understand their trustworthiness, especially when applied in critical decision-making processes. Thus, within this master's thesis, the computation of the trustworthiness level of Black Box models has been studied. In the beginning, a comprehensive background on ML and the distinction between Black Box and White Box models was introduced. It also covered essential concepts of trustworthiness in the context of ML/DL, introducing existing pillars for assessing trustworthiness with a particular emphasis on interpretability, transparency, and explainable AI (XAI). The *Related Work* Chapter surveyed scientific studies dedicated to enhancing the trustworthiness of Black Box ML models and portrayed existing tools, highlighting their contributions and limitations. Early work has proposed a Trusted AI Platform, capable of computing a trust score for White Box and Gray Box ML models. During the literature review, two common definitions of the term Black Box have been identified in the research community. Most related work has focused on the first definition, where the term Black Box is describing complex systems that are not interpretable, even for domain experts. However, the second definition, describing systems for which only limited information is available, to my best knowledge, has not yet been researched with respect to trustworthy AI so far.

In the *Design and Implementation* Chapter, a Black Box Taxonomy was presented, taking both definitions into consideration by grouping ML models into four subgroups based on their interpretability and adding another dimension for the distinction between the level of available information. From the taxonomy, a limitation in the existing trustworthiness analysis for ML models with limited information (no access to the original dataset) was identified, as certain metrics become incomputable. Therefore, the concept of a synthetic dataset generator has been proposed, providing a novel solution to make incomputable metrics quantifiable and improve the trustworthiness assessment for models with limited information. The developed solution provides two ways of generating a synthetic dataset

(MUST and MAY), varying in the number of statistical properties about the original dataset it takes as input, addressing the trade-off between privacy and accuracy. The process design and potential integration into the Trusted AI Platform 2.0 have been discussed. Finally, the dataset generation and trustworthiness assessment have been tested and evaluated in two different scenarios. For each evaluation scenario, the results of four different trust scores (True Score, Limited Score, Advanced Score MUST, and Advanced Score MAY) have been compared, analyzed, and discussed. The advantages and limitations of the implemented solution have been outlined and possible suggestions for improvement have been proposed.

6.2 Conclusion

Within the scope of this thesis, the feasibility of improving the trustworthiness assessment of ML models with limited information using a synthetic dataset can be confirmed. The unavailability of the original dataset consequentially leads to a decreased number of computable metrics, as by definition, most existing metrics rely on either the original training dataset or the true target labels for its computability. The methodology of using a synthetic dataset enabled certain incomputable metrics to become quantifiable and contributed to a more expressive trust score. Further, the evaluation process has uncovered some learnings and findings when comparing different solutions (also if all information is available), for which a different number of metrics are computed. Thus, this work suggested providing the score as a confidence interval (CI) for the solution with fewer computed metrics, taking into consideration the level of uncertainty.

The comparison between the MUST and MAY datasets showed the differences in discrete and continuous features. While the MUST dataset preserves higher privacy protection, the MAY dataset results in a more realistic representation of the original dataset. The comparison of categorical features between the datasets has identified a limitation that can be improved by considering the proposed statistical properties, such as the correlation or unique count (ratio) to filter out *impossible* samples and improve the distribution over the individual categories. It can be concluded, that the existence of both approaches provides a flexible framework to balance privacy concerns and the need for more accurate trustworthiness assessments. Organizations can choose the level of privacy they are comfortable with, while still obtaining reasonably accurate insights into their models' capabilities. The synthetic dataset generator is built in a modular way, which allows simple integration into the existing Trusted AI Platform 2.0.

Concluding, the presented work contributes to the research community by providing a Black Box Taxonomy that takes into account both common definitions of the term *Black Box* known by the community. Further, a novel approach was implemented, that allows the systematic generation of a tabular dataset from scratch, without having access to the original dataset. Lastly, it proposed a privacy-preserving way of computing the Black Box model's trustworthiness with the use of the synthetic dataset, resulting in a more accurate trust score.

6.3 Limitations and Future Work

Although the presented solution offers a tested and documented implementation, it is crucial to recognize the limitations that emerged throughout the research and implementation process, such that they can be addressed and improved, for instance in future work. As discussed throughout the presented work, the statistical properties for categorical features in the MAY scenario may be enhanced to more accurately reflect the properties of the original dataset. Further, additional metrics could be defined or added that do not require the original dataset or true labels, to further increase the number of computable metrics. This would increase the confidence of the computed pillar scores and decrease the range of the proposed CI. Also, the current implementation only supports the creation of tabular data. As complex Gray Box models such as DNNs have proven to work well in the field of computer vision and natural language processing, it could be interesting to consider generating synthetic data in the form of images or text. Finally, the implementation of the synthetic dataset generator into an existing platform would generate more awareness for privacy-preserving trustworthiness assessments and lower the barrier for potential users.

Declaration of Independence for written work

Selbständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel (inklusive generativer KI wie z.B. ChatGPT) angefertigt habe. Mir ist bekannt, dass ich die volle Verantwortung für die Wissenschaftlichkeit des vorgelegten Textes selbst übernehme, auch wenn (nach schriftlicher Absprache mit der betreuenden Professorin resp. dem betreuenden Professor) KI-Hilfsmittel eingesetzt und deklariert wurden. Alle Stellen, die wörtlich oder sinngemäss aus veröffentlichten oder nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Kriens, den August 7, 2023 D. Gagulić ……………………

Statement of authorship

I hereby declare that I have composed this work independently and without the use of any aids other than those declared (including generative AI such as ChatGPT). I am aware that I take full responsibility for the scientific character of the submitted text myself, even if AI aids were used and declared (after written confirmation by the supervising professor). All passages taken verbatim or in sense from published or unpublished writings are identified as such. The work has not yet been submitted in the same or similar form or in excerpts as part of another examination.

Kriens, August 7, 2023 D. Gagulić ……………………

Bibliography

[1] E. Uhlemann, "Connected-vehicles applications are emerging [connected vehicles]," *IEEE Vehicular Technology Magazine*, vol. 11, no. 1, pp. 25–96, 2016.

[2] H. Kang and C. Lou, "Ai agency vs. human agency: understanding human–ai interactions on tiktok and their implications for user engagement," *Journal of Computer-Mediated Communication*, vol. 27, no. 5, p. zmac014, 2022.

[3] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[5] J. Deng and Y. Lin, "The benefits and challenges of chatgpt: An overview," *Frontiers in Computing and Intelligent Systems*, vol. 2, no. 2, pp. 81–83, 2022.

[6] I. Research, "Ibm 360 trustworthy ai," https://research.ibm.com/topics/trustworthy-ai, accessed: 29.05.2023.

[7] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[8] I. D. Raji, T. Gebru, M. Mitchell, J. Buolamwini, J. Lee, and E. Denton, "Saving face: Investigating the ethical concerns of facial recognition auditing," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 145–151.

[9] J. Dastin, "Amazon scraps secret ai recruiting tool that showed bias against women," in *Ethics of data and analytics.* Auerbach Publications, 2022, pp. 296–299.

[10] P. H. O'Neill, "Mit technology review," https://www.technologyreview.com/2020/02/19/868188/hackers-can-trick-a-tesla-into-accelerating-by-50-miles-per-hour/, accessed: 02.08.2023.

[11] A. H. Celdran, J. Kreischer, M. Demirci, J. Leupp, P. M. Sanchez, M. F. Franco, G. Bovet, G. M. Perez, and B. Stiller, "A framework quantifying trustworthiness of supervised machine and deep learning models," in *SafeAI2023: The AAAI's Workshop on Artificial Intelligence Safety*, 2023, pp. 2938–2948.

[12] J. Leupp, M. Demirici, and J. Bauer, "Quantifying the trustworthiness level of artificial intelligence models and decisions," 2022.

[13] P. C. Jackson, *Introduction to artificial intelligence.* Courier Dover Publications, 2019.

[14] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning.* Springer, 2006, vol. 4, no. 4.

[15] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[16] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[17] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous uav navigation using reinforcement learning," *arXiv preprint arXiv:1801.05086*, 2018.

[18] J. von der Assen, A. H. Celdrán, J. Luechinger, P. M. S. Sánchez, G. Bovet, G. M. Pérez, and B. Stiller, "Ransomai: Ai-powered ransomware for stealthy encryption," *arXiv preprint arXiv:2306.15559*, 2023.

[19] A. H. Celdrán, P. M. S. Sánchez, J. von der Assen, T. Schenk, G. Bovet, G. M. Pérez, and B. Stiller, "Rl and fingerprinting to select moving target defense mechanisms for zero-day attacks in iot," *arXiv preprint arXiv:2212.14647*, 2022.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[21] R. M. Cichy and D. Kaiser, "Deep neural networks as scientific models," *Trends in cognitive sciences*, vol. 23, no. 4, pp. 305–317, 2019.

[22] A. Jeerige, D. Bein, and A. Verma, "Comparison of deep reinforcement learning approaches for intelligent game playing," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC).* IEEE, 2019, pp. 0366–0371.

[23] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *arXiv preprint arXiv:1704.02532*, 2017.

[24] D. Castelvecchi, "Can we open the black box of ai?" *Nature News*, vol. 538, no. 7623, p. 20, 2016.

[25] S. Supriyono, "Software testing with the approach of blackbox testing on the academic information system," *IJISTECH (International Journal of Information System and Technology)*, vol. 3, no. 2, pp. 227–233, 2020.

[26] S. R. Jan, S. T. U. Shah, Z. U. Johar, Y. Shah, and F. Khan, "An innovative approach to investigate various software testing techniques and strategies," *International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Print ISSN*, vol. 23951990, 2016.

[27] M. A. Ahmad, C. Eckert, and A. Teredesai, “Interpretable machine learning in healthcare,” in *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, 2018, pp. 559–560.

[28] K. Rasheed, A. Qayyum, M. Ghaly, A. Al-Fuqaha, A. Razi, and J. Qadir, “Explainable, trustworthy, and ethical machine learning for healthcare: A survey,” *Computers in Biology and Medicine*, p. 106043, 2022.

[29] M. Kentour and J. Lu, “Analysis of trustworthiness in machine learning and deep learning,” *The Eleventh International Conference on Advanced Communications and Computation*, 2021.

[30] O. Loyola-Gonzalez, “Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view,” *IEEE access*, vol. 7, pp. 154 096–154 113, 2019.

[31] T. Mahoney, K. Varshney, and M. Hind, *AI Fairness.* O’Reilly Media, Incorporated, 2020.

[32] G. V. Travaini, F. Pacchioni, S. Bellumore, M. Bosia, and F. De Micco, “Machine learning and criminal justice: a systematic review of advanced methodology for recidivism risk prediction,” *International journal of environmental research and public health*, vol. 19, no. 17, p. 10594, 2022.

[33] I. Research, “Ai explainability 360,” https://aix360.mybluemix.net/, accessed: 16.05.2023.

[34] R. Hamon, H. Junklewitz, and I. Sanchez, “Robustness and explainability of artificial intelligence,” *Publications Office of the European Union*, vol. 207, 2020.

[35] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[36] I. Research, “Art: Adversarial robustness toolbox,” https://github.com/Trusted-AI/adversarial-robustness-toolbox, accessed: 16.05.2023.

[37] F. Yu, Z. Qin, C. Liu, L. Zhao, Y. Wang, and X. Chen, “Interpreting and evaluating neural network robustness,” *arXiv preprint arXiv:1905.04270*, 2019.

[38] M.-I. Nicolae, M. Sinn, M. N. Tran, B. Buesser, A. Rawat, M. Wistuba, V. Zantedeschi, N. Baracaldo, B. Chen, H. Ludwig *et al.*, “Adversarial robustness toolbox v1. 0.0,” *arXiv preprint arXiv:1807.01069*, 2018.

[39] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, “Evaluating the robustness of neural networks: An extreme value theory approach,” *arXiv preprint arXiv:1801.10578*, 2018.

[40] H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, and C.-J. Hsieh, “Robustness verification of tree-based models,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[41] D. Piorkowski, D. González, J. Richards, and S. Houde, "Towards evaluating and eliciting high-quality documentation for intelligent systems," *arXiv preprint arXiv:2011.08774*, 2020.

[42] I. Research, "Ai factsheets 360," https://aifs360.mybluemix.net/introduction, accessed: 26.05.2023.

[43] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[44] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, no. 1, 2017, pp. 8–13.

[45] Dictionary.com, "Definition interpret," https://www.dictionary.com/browse/interpret, accessed: 14.05.2023.

[46] A. Ferrario and M. Loi, "How explainability contributes to trust in ai," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 1457–1466.

[47] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[48] G. Stiglic, P. Kocbek, N. Fijacko, M. Zitnik, K. Verbert, and L. Cilar, "Interpretability of machine learning-based prediction models in healthcare," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, no. 5, p. e1379, 2020.

[49] OpenAI, "Introducing chatgpt," https://openai.com/blog/chatgpt, accessed: 13.05.2023.

[50] M. Craven and J. Shavlik, "Extracting tree-structured representations of trained networks," *Advances in neural information processing systems*, vol. 8, 1995.

[51] C. Rudin, "Please stop explaining black box models for high stakes decisions," *stat*, vol. 1050, p. 26, 2018.

[52] ——, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[53] C. B. Azodi, J. Tang, and S.-H. Shiu, "Opening the black box: interpretable machine learning for geneticists," *Trends in genetics*, vol. 36, no. 6, pp. 442–455, 2020.

[54] C. Chandler, P. W. Foltz, and B. Elvevåg, "Using machine learning in psychiatry: the need to establish a framework that nurtures trustworthiness," *Schizophrenia bulletin*, vol. 46, no. 1, pp. 11–14, 2020.

[55] M. T. Ribeiro, S. Singh, and C. Guestrin, "" 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, pp. 1135–1144.

[56] S. Lundberg, "Shap github page," https://github.com/slundberg/shap, accessed: 28.05.2023.

[57] M. Korobov and K. Lopuhin, "Eli5 documentation page," https://eli5.readthedocs.io/en/latest/overview.html, accessed: 29.05.2023.

[58] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[59] P. Besse, C. Castets-Renard, A. Garivier, and J.-M. Loubes, "Can everyday ai be ethical? machine learning algorithm fairness," *Machine Learning Algorithm Fairness (May 20, 2018). Statistiques et Société*, vol. 6, no. 3, 2019.

[60] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh, "Vqa: Visual question answering," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2425–2433.

[61] A. Das, H. Agrawal, L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?" *Computer Vision and Image Understanding*, vol. 163, pp. 90–100, 2017.

[62] Microsoft, "Microsoft fairlearn," https://fairlearn.org/, accessed: 29.05.2023.

[63] B. Zhang, S. Gu, B. Zhang, J. Bao, D. Chen, F. Wen, Y. Wang, and B. Guo, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 304–11 314.

[64] W. H. Pinaya, P.-D. Tudosiu, J. Dafflon, P. F. Da Costa, V. Fernandez, P. Nachev, S. Ourselin, and M. J. Cardoso, "Brain imaging generation with latent diffusion models," in *MICCAI Workshop on Deep Generative Models.* Springer, 2022, pp. 117–126.

[65] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[66] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW).* IEEE, 2018, pp. 117–122.

[67] I. Ashrapov, "Tabular gans for uneven distribution," *arXiv preprint arXiv:2010.00638*, 2020.

[68] A. J. Myles, R. N. Feudale, Y. Liu, N. A. Woody, and S. D. Brown, "An introduction to decision tree modeling," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 18, no. 6, pp. 275–285, 2004.

[69] J. R. Quinlan, "Generating production rules from decision trees." in *ijcai*, vol. 87. Citeseer, 1987, pp. 304–307.

[70] X. Su, X. Yan, and C.-L. Tsai, "Linear regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012.

[71] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[72] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, pp. 1937–1967, 2021.

[73] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings.* Springer, 2003, pp. 986–996.

[74] P. S. Foundation, "Itertools module," https://docs.python.org/3/library/itertools.html#itertools.product, accessed: 24.07.2023.

[75] A. Doerr and K. Levasseur, "Applied discrete structures - 1: Set theory," https://shorturl.at/fiFK0, accessed: 24.07.2023.

[76] D. Gagulic, "Trustworthy black box repository," https://github.com/dgagul/trustworthyBlackbox, accessed: 06.08.2023.

[77] A. Bohr and K. Memarzadeh, *Artificial intelligence in healthcare.* Academic Press, 2020.

[78] Kaggle, "Diabetes prediction dataset," https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset?resource=download, accessed: 25.07.2023.

[79] S. learn Developers, "sklearn.ensemble.randomforestclassifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, accessed: 25.07.2023.

[80] J. Han, Y. Huang, S. Liu, and K. Towey, "Artificial intelligence for anti-money laundering: a review and extension," *Digital Finance*, vol. 2, no. 3-4, pp. 211–239, 2020.

[81] Kaggle, "Home credit default risk," https://www.kaggle.com/competitions/home-credit-default-risk/data?select=application_train.csv, accessed: 26.07.2023.

[82] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE symposium on security and privacy (SP).* IEEE, 2017, pp. 3–18.

Abbreviations

AC	Algorithm Class
AI	Artificial Intelligence
AM	Activation Maximization
amt	amount
API	Application Programming Interface
ART	Adversarial Robustness Toolbox
CDF	Cumulative Distribution Function
CEM	Contrastive Explanations Method
CF	Correlated Features
CI	Confidence Interval
CLEVER	Cross Lipschitz Extreme Value for Network Robustness
CNN	Convolutional Neural Network
DI	Disparate Impact
DL	Deep Learning
DNN	Deep Neural Network
DR	Decision Rule
DS	Dataset
DT	Decision Tree
dType	Data Type
e.g.	exempli gratia
EOD	Equal Opportunity Difference
ER	Empirical Robustness
et al.	et alii/et aliae/et alia
etc.	et cetera
f.i.	for instance
FI	Feature Importance
FPR	False Positive Rate
FR	Feature Relevance
FYP	For You Page
GAN	Generative Adversarial Network
GB	Gradient Boost
Grad-CAM	Gradient-weighted Class Activation Mapping
i.e.	id est
kNN	k-Nearest Neighbors
LIME	Local Interpretable Model-Agnostic Explanations
LSTM	Long Short-Term Model

max	maximum
min	minimum
ML	Machine Learning
MS	Model Size
NLM	Non-Linear Model
NLP	Natural Language Processing
NN	Neural Network
p.	page
PDP	Partial Dependence Plot
PF	Punishment Factor
PS	Prototype Selection
RF	Random Forest
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SA	Sensitivity Analysis
SHAP	Shapley Additive exPlanations
SM	Saliency Mask
std	standard deviation
SVM	Support Vector Machine
TE	Tree Ensemble
TPR	True Positive Rate
VAE	Variational Autoencoder
VQA	Visual Question Answering
vs.	versus
XAI	Explainable Artificial Intelligence

List of Figures

List of Tables

Code Listings

List of Algorithms

Appendix A

Additional Code Excerpt

```python
# Compute Feature Importance using SHAP Explainer
nr_random_samples = 20
rand_idxs = np.random.randint(0, test_data.shape[0], nr_random_samples)
sample_data = test_data.iloc[rand_idxs]
bg = sample_data.drop(target_column, axis=1)

# Fits the explainer
explainer = shap.Explainer(clf.predict, bg)
shap_values = explainer(bg)

# Calculates the feature importance (mean absolute shap value) for each feature
importance = []
for i in range(shap_values.values.shape[1]):
    importance.append(np.mean(np.abs(shap_values.values[:, i])))
importance = scipy.special.softmax(np.asarray(importance))
```

Code Listing A.1: Extantion Of The feature_relevance_score Function In The Explainability Pillar

```python
# New Suggestions to improve the Model Size Metric
if isinstance(clf, sklearn.ensemble.RandomForestClassifier):
    # Get the overall maximum depth
    overall_max_depth = max([estimator.tree_.max_depth for estimator in clf.estimators_])
    print(overall_max_depth)

    # Get the number of trees
    nr_trees = clf.n_estimators
    print(nr_trees)

    # Calculate the total number of nodes
    total_nodes = sum([dt.tree_.node_count for dt in clf.estimators_])
    print(total_nodes)

elif isinstance(clf, sklearn.neighbors.KNeighborsClassifier):
    # Get the "k" value
    k_value = clf.n_neighbors
    print(k_value)

elif isinstance(clf, sklearn.svm.SVC):
    # Get the number of support vectors
    total_support_vectors = len(clf.support_vectors_)
    print(total_support_vectors)

elif isinstance(clf, torch.nn.Module):
    # Calculate the number of parameters of a PyTorch DNN
    total_params = sum(p.numel() for p in clf.parameters() if p.requires_grad)
    print(total_params)
    # Get the number of layers in the PyTorch model

    num_layers = len(list(clf.children()))
    print(num_layers)

elif isinstance(clf, tf.keras.models.Sequential) or isinstance(clf, tf.keras.models.Model):
    # Calculate the number of parameters of a Keras/Tensorflow DNN
    total_params = clf.count_params()
    print(total_params)

    # number of layers
    num_layers = len(clf.layers)
    print(num_layers)
```

Code Listing A.2: Extantion Of The model_size_score Function In The Explainability Pillar