A Framework Quantifying Trustworthiness of Supervised Machine and Deep Learning Models

Alberto Huertas Celdran[1,*], Jan Kreischer[1], Melike Demirci[1], Joel Leupp[1], Pedro M. Sanchez Sanchez[2], Muriel Figueredo Franco[1], Gérôme Bovet[3], Gregorio Martinez Perez[2] and Burkhard Stiller[1]

[1]Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH
[2]Department of Information and Communications Engineering, University of Murcia
[3]Cyber-Defence Campus, armasuisse Science & Technology

Abstract

Trusting Artificial Intelligence (AI) is controversial since models and predictions might not be fair, understandable by humans, robust against adversaries, or trained appropriately. Existing toolkits help data scientists to create fair, explainable, robust, and transparent Machine and Deep Learning (ML/DL) models. However, tools to quantify AI trustworthiness according to pillars and metrics relevant for heterogeneous scenarios are still missing. This work proposes a novel algorithm that quantifies the trustworthiness level of supervised ML/DL models according to their fairness, explainability, robustness, and accountability. The algorithm is deployed on a Web application to allow the general public to calculate the trustworthiness of their models. Finally, a validation scenario with models classifying cyberattacks demonstrates the applicability of the Web application and algorithm.

Keywords

Trust, Supervised Machine Learning, Trust Framework, Deep Learning Trust

1. Introduction

Artificial Intelligence (AI) made great strides over the last decade [1]. Some relevant examples are the victory of IBM Watson in the open-domain Q&A game of Jeopardy (2011), the first version of Tesla Autopilot, with lane control, breaking, and speed limit adjustment (2014), or when AlphaGo defeated world champion Lee Sedol in the Go game (2016). In parallel to these achievements, AI has also gained relevance as support to human decision-making, spanning from trivial to highly complex applications. The diagnosis and treatment of diseases, assessment of legal issues, or admission to credits are good examples of current tasks supported by AI systems. In these scenarios and many others, delegating partially or entirely the decision-making process to automated and intelligent systems generates a trust dependency. Thus, it is critical to have mechanisms to quantify the trustworthiness level of AI systems and their predictions.

Recently, the research community has agreed on the importance of fairness, explainability, robustness, and accountability as pillars for trusted AI [2]. Concerning fairness, bias is one of the main issues against trusting AI systems [3]. Bias can be introduced by human prejudice in the training dataset or assumptions made during the process of training Machine and Deep Learning (ML/DL) models. Furthermore, too small, incomplete, or not diverse enough training datasets might introduce bias as well. The literature has proposed different solutions to detect and avoid bias. For instance, IBM AI Fairness 360 [4] contains a set of metrics to detect bias in ML/DL models and datasets. Algorithms to mitigate bias during the preprocessing, in-processing, and post-processing stages are also available in the IBM toolkit. Another de-biasing approach is called Fairness GAN [5], which creates new and unbiased datasets from original biased ones. However, aspects dealing with the explainability, robustness, or accountability of models are not taken into account by these tools.

Explainability is another important pillar that consists of understanding how ML/DL models come to their conclusions. When AI is used in high-risk fields, all stakeholders should understand the main decision drivers. For human decision-making, trust can be gained by explaining the underlying rationale. However, for AI, aspects such as the algorithm class, features importance, or model complexity are some of the key aspects to explain predictions [6]. In this context, the IBM AI Explainability 360 toolkit [7] contains a pool of algorithms and methods to explain model details and predictions. This toolkit also includes two metrics to evaluate the goodness of explanations. In addition, the literature has created various libraries implementing local explanations methods like LIME or SHAPE [8]. Despite the contributions of these solutions they only focus on explainability and do not

SafeAI2023: The AAAI's Workshop on Artificial Intelligence Safety, February 13–14, 2023, Washington, D.C.
*Corresponding author.
huertas@ifi.uzh.ch (A. Huertas Celdran); jan.kreischer@uzh.ch (J. Kreischer); melike.cilogluu@gmail.com (M. Demirci); joel.leupp@uzh.ch (J. Leupp); pedromigel.sanchez@um.es (P. M. Sanchez Sanchez); franco@ifi.uzh.ch (M. F. Franco); gerome.bovet@armasuisse.ch (G. Bovet); gregorio@um.es (G. Martinez Perez); stiller@ifi.uzh.ch (B. Stiller)
0000-0001-7125-1710 (A. Huertas Celdran); 0000-0002-6444-2102 (P. M. Sanchez Sanchez); 0000-0002-0208-0521 (M. F. Franco); 0000-0002-4534-3483 (G. Bovet); 0000-0001-5532-6604 (G. Martinez Perez); 0000-0002-7461-7463 (B. Stiller)

consider other important aspects of trusted AI like the robustness of ML models.

In this sense, robustness is the third pillar of trusted AI and refers to the model ability to deal with adversarial samples. Even in black-box approaches, where model details are unknown to adversaries, it is possible to cause unexpected predictions using adversarial attacks. Therefore, to ensure trusted ML/DL models, their predictions must be stable and robust, even when adversaries are present [9]. In this context, [10] proposes a comprehensive taxonomy for adversarial robustness and discusses potential consequences of attacks considering data integrity, confidentiality, and privacy. From a different perspective, the IBM Adversarial Robustness Toolbox (ART) [11] offers a collection of adversarial attack/defense implementations, run-time attack detection methods, poisoning detection, and robustness metrics. These metrics are very relevant to calculate the trustworthiness level of AI models, but others dealing with the methodology followed to train models are missing.

To measure the quality of the ML pipeline or methodology used to train ML/DL models, the fourth main pillar of trusted AI proposes to use accountability and transparency aspects. More in detail, train/test splitting strategy, data pre-processing, normalization, or feature extraction and selection are some aspects providing valuable insights to trust AI systems. In this sense, [12] proposes the creation of FactSheets as a form of AI documentation. Additionally, [13] studies the documentation needed by stakeholders to trust AI. Finally, the IBM AI FactSheets 360 [4] provides a guide and tool for the manual preparations of factSheets.

Despite the valuable contributions of previous works, trusted AI is an emerging research field that needs more effort. In particular, the state-of-the-art lacks a comprehensive and unified collection of relevant metrics per pillar to identify and quantify trusted ML/DL models. Furthermore, existing solutions focus on detecting and mitigating different issues per pillar. However, there is no solution combining the pillars and computing a global trustworthiness level of ML/DL models. Last but not least, IBM toolkits are helping data scientists to create fair, explainable, robust, and transparent ML/DL models. Nevertheless, tools to quantify AI trustworthiness in heterogeneous and real-world scenarios are still missing.

To address the previous challenges, this article presents the following contributions:

- A taxonomy with the four main pillars of trusted AI (fairness, explainability, robustness, and accountability) and their most relevant metrics to quantify the trustworthiness level of supervised ML/DL models.
- An extensible, adaptive and parameterized algorithm (available in [14]) to quantify the trustworthiness level of supervised ML/DL models with tabular data according to the pillars and metrics of the proposed taxonomy.
- A Web application [15] that allows stakeholders to use the proposed algorithm to compute and compare the trustworthiness level of their supervised ML/DL models.
- A case study focused on models classifying malware affecting Internet of Things (IoT) devices to demonstrate the suitability of the developed Web application and proposed algorithm.

The remainder of this paper is structured as follows. Section 2 analyzes previous works in the area of trusted AI. Section 3 presents the pillars and metrics relevant for trusted AI. Section 4 introduces the design and implementation of the trusted AI algorithm. While Section 5 presents the functionality of the Web application, Section 6 validates the algorithm in a cybersecurity scenario. Finally, Section 7 draws some conclusions and future work.

2. Related Work

First, it is important to mention that trust in AI is an incipient field, and, to the best of our knowledge, there is no solution automatically assessing the trustworthiness level of ML/DL models in multiple dimensions. However, related work has proposed metrics and tools for individual dimensions or pillars impacting model trustworthiness. Therefore, this section reviews work dealing with AI trustworthiness in terms of fairness, explainability, robustness, and accountability.

Starting from fairness, the authors of [16] introduced the notions of group fairness and individual fairness and discussed how model developers could address the topic. In the same direction, [17] proposed an algorithm for fair classification that complied with the two previous notions of fairness. To mitigate unfairness in AI, [5] proposed a novel de-biasing approach, called Fairness GAN, capable of creating a new dataset that approximates a given original biased one. Reviewing existing tools and frameworks, IBM AI Fairness 360 toolkit [4] contains a comprehensive set of fairness metrics such as statistical parity, equal opportunity, or average odds that can be used to detect bias in ML models and datasets. Furthermore, the toolkit offers algorithms capable of mitigating bias during the pre-processing, in-processing, and post-processing stages. Facebook internal Fairness Flow toolkit [18] is another solution composed of a Python library that provides a simple API that requires a data set of predictions, labels, and group membership. As an output, the API offers informative metrics, statistical confidence, and how to interpret the results. Microsoft Fairlearn [19] is another open-source project that combines visualization capabilities with unfairness detection and mitigation algorithms.

Explainability is another important dimension for trusted AI. In this sense, IBM AI Explainability 360 toolkit [7] contains several algorithms and methods to create explanations for ML models or make explainable

models like decision trees more accurate. It also includes two proxy metrics to evaluate the goodness of explanations. Compared to the work at hand, this toolkit does not evaluate the level of model explainability but provides explanation methods for different types of models. Additionally, various python libraries implemented local explanations methods like LIME or SHAPE [8]. All those solutions aim to explain the decisions of models but not to classify models on their explainability level, as the paper at hand does.

Robustness against adversarial attacks is something well-studied in the literature. The authors of [10] presented a comprehensive taxonomy with explanations for existing attacks and defenses. They also discussed the potential consequences of adversarial attacks in terms of data integrity, confidentiality, and privacy. However, the authors did not analyze metrics measuring the models robustness. In terms of metrics, the literature focuses mainly on intrinsic and post-hoc explainability methods. In terms of existing tools, IBM ART [11] is an open-source Python library that collects adversarial robustness techniques on ML models. It consists of adversarial attack/defense implementations, run-time attack detection methods, poisoning detection, and robustness metrics. This toolbox implements multiple attack detection and defense techniques. Similar to the work at hand, ART also gathers possible robustness metrics.

Dealing with accountability, [12] proposed a methodology for creating factsheets. Factsheets provide information regarding ML models, such as training data, model type, or training methodology. All stakeholders involved in the AI lifecycle contribute to the Factsheet creation, and it covers the expertise gap between AI producers and consumers. The authors of [13] studied the necessities of developers and other stakeholders to understand what content to include in factsheets. They highlighted the importance of documenting how a model was structured, what training data was used, and how features were engineered. In terms of existing tools, IBM AI FactSheets 360 [4] provides a guide for preparing factsheets. Transparent and well-documented communication between model creator and model operator increases trust and enables more efficient integration of pre-trained models. One limitation of the previous works is that the creation of factsheets requires a lot of manual effort. The paper at hand automates this process as much as possible.

In conclusion, despite the contributions made by the previous works, there is no automatic tool or solution combining different pillars and metrics to compute the trustworthiness level of supervised ML models.

3. Main Pillars & Metrics of Trustworthy AI

This section describes the pillars and metrics identified by this work as the most relevant to quantify the trustworthiness of supervised ML/DL models (see Figure 1).

3.1. Fairness Pillar

ML/DL models can be classified as unfair due to different reasons such as *i)* biased training data, *ii)* unbalanced or lack of training data, or *iii)* discrimination of *protected groups*, among others. The concept of *protected group* refers to a set of samples sharing a common property or feature value. For instance, in a job application, women could be a protected group to avoid gender discrimination. Considering these reasons, this work proposes the following metrics to quantify the fairness level of supervised ML/DL models.

- *Underfitting*: detects if the model is unable to learn the relationship between inputs and outputs accurately. In this work, Eq (1) calculates underfitting as the difference between the model accuracy with train data and a baseline threshold (established according to the expected accuracy of the selected application scenario).

$$Underfitting = Acc_{train} - Acc_{baseline} \quad (1)$$

- *Overfitting*: measures the model generalization capabilities. In this work, Eq (2) calculates overfitting as the difference between the model accuracy with train and test datasets.

$$Overfitting = Acc_{train} - Acc_{test} \quad (2)$$

- *Class Balance*: measures the ratio of samples belonging to different classes in the training dataset. Class imbalance is a problem for predicting tasks of infrequent classes with few samples. In this work, the class balance is computed by using Eq. (3) and chi-square (X^2) distribution. It performs a statistical test that checks the deviation between a perfect samples distribution (E_i) for all classes (i) and the actual distribution (O_i).

$$X^2 = \Sigma_i \frac{(O_i - E_i)^2}{E_i} \quad (3)$$

- *Statistical Parity Difference*: computes the spread between the percentage of samples belonging to the majority group receiving a favorable outcome compared to a protected group [20]. This metric and the following three detect discrimination in predictions using samples of different groups. Eq. (4) measures the Statistical Parity Difference (SPD) between both groups. $\hat{Y} = 1$ represents a favorable prediction and $Pr(\hat{Y} = 1|P = p)$ is the probability of receiving a favorable prediction if the sample belongs to the protected minority ($P = 1$) or the unprotected majority ($P = 0$). If SPD is close to zero, the classifier has *statistical parity*.

$$SPD(\hat{Y}, Y, P) = |Pr(\hat{Y} = 1|P = 1) - Pr(\hat{Y} = 1|P = 0)| \quad (4)$$

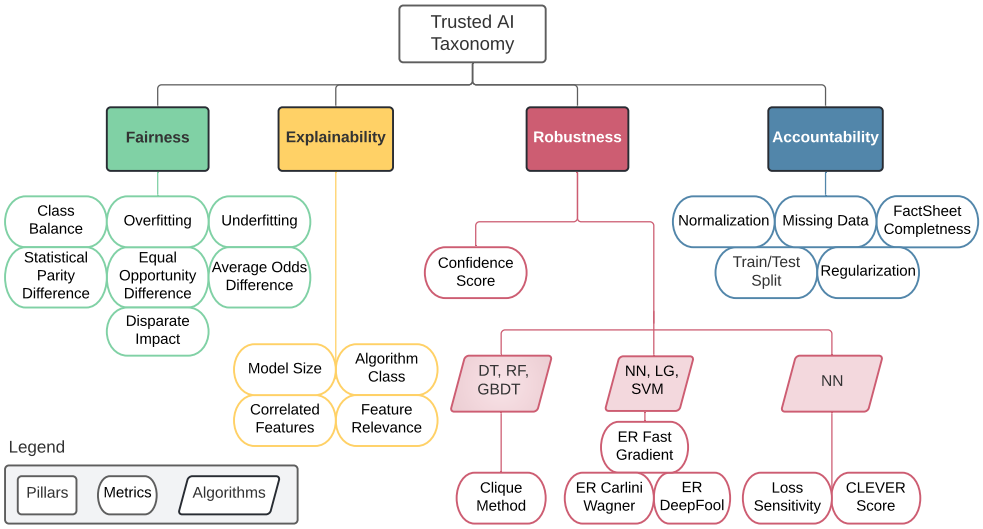

Figure 1: Taxonomy of Pillars and Metrics Involved in the Computation of Trustworthy AI

- *Equal Opportunity Difference*: measures the spread between true positive rate (TPR) and false positive rate (FPR) of protected and unprotected groups [21]. Eq. (5) calculates the Equal Opportunity Difference (EOD). $Pr\{\hat{Y} = 1|P = 0, Y = 0\}$ and $Pr\{\hat{Y} = 1|P = 1, Y = 0\}$ are the FPR of unprotected and protected groups, respectively. Additionally, $Pr\{\hat{Y} = 1|P = 0, Y = 1\}$ and $Pr\{\hat{Y} = 1|P = 1, Y = 1\}$ are the TPR of unprotected and protected groups.

$$EOD(\hat{Y}, P) = Pr\{\hat{Y} = 1|P = 1, Y = y\} - Pr\{\hat{Y} = 1|P = 0, Y = y\}, y \in 0, 1 \quad (5)$$

- *Average Odds Difference*: calculates the mean absolute difference in TPR and FPR between protected and unprotected groups [22]. Eq. (6) measures the Average Odds Difference (AOD) as $1/2[(FPR_{unprotected} - FPR_{protected}) + (TPR_{unprotected} - TPR_{protected})]$

$$AOD(\hat{Y}, Y, P) = \Sigma_{y \in \{0,1\}} |Pr\{\hat{Y} = 1|P = 1, Y = y\} - Pr\{\hat{Y} = 1|P = 0, Y = y\}| / 2 \quad (6)$$

- *Disparate Impact*: measures the ratio of a protected group receiving a favorable prediction divided by the ratio of an unprotected group receiving a favorable outcome. Eq (7) calculates the Disparate Impact (DI) and uses the previously defined variables.

$$DI(\hat{Y}, Y, P) = \frac{Pr\{\hat{Y} = 1|P = 1\}}{Pr\{\hat{Y} = 1|P = 0\}} \quad (7)$$

3.2. Explainability Pillar

Explainable artificial intelligence focuses on enabling a deeper understanding of the inner workings of ML/DL models. To quantify the explainability of models and their predictions, the following metrics (which assume tabular data format) are used.

- *Algorithm Class*: indicates the model explainability degree according to the algorithm type and its complexity. This work defines a predefined and configurable score for each algorithm, which is inspired by the literature.
- *Correlated Features*: measures the percentage of highly correlated features. High correlation among features might lead to biases in most explanation techniques. This work calculates the number of highly correlated features (>= 95%) and predefines configurable thresholds to compute the metric score.
- *Feature Relevance*: calculates the percentage of irrelevant features for a set of predictions. The lower irrelevant features, the better, as they would only make explanations more complex without being relevant [23]. As in the previous metric, this work calculates the number of irrelevant features (relevance <= 3%) for the model and predefines configurable thresholds to compute the metric score.
- *Model Size*: indicates the number of parameters used by models. In this work, this metric considers predefined thresholds that are used to evaluate the model comprehensibility, which inversely correlates with the model degrees of freedom determined by the number of parameters.

3.3. Robustness Pillar

To trust ML/DL models, their predictions should be stable and robust. If small changes in the input data cause significant deviations in the output, adversarial perturbations can be used to generate undesired outcomes. This work proposes using the following metrics to measure the robustness of supervised ML/DL models.

- *Confidence Score*: measures the probability of correctly predicting samples. It calculates the stability of predictions, the more stable the predictions, the more robust the model [24]. Eq (8) calculates the confidence score as the mean over all the precision scores for all thresholds. Where T are the labels, and $Thrs$ are the probabilities that one vector is classified correctly.

$$Confidence = \frac{1}{|Thrs|}\Sigma_T \frac{TP}{TP + FP + FN} \tag{8}$$

- *Loss Sensitivity*: calculates the largest variation of the output of a Neural Network (NN) under a small change in its input. Overall, it quantifies the smoothness of a model [24]. The smaller the variation in the output, the smoother the model. Eq (9) calculates Loss sensitivity (g), where $\mathcal{L}$ is the loss function.

$$g = \left\| \frac{\partial \mathcal{L}}{\partial x} \right\|_1 \tag{9}$$

- *Cross Lipschitz Extreme Value for Network Robustness (CLEVER) Score*: measures the minimal perturbation that is needed to change the classification outcome [25] using the local Lipschitz constant [11]. It is applicable to NN.
- *Clique Method*: finds the exact minimal adversarial perturbation or a guaranteed lower bound of it [26]. It is applicable to Decision Trees (DT), Random Forests (RF), and Gradient Boosted Decision Trees (BGDT).
- *Empirical Robustness*: measures the average minimal perturbation that needs to be introduced to change the model prediction [11]. Empirical robustness (ER) is applicable to NN, Logistic Regression (LG), and Support Vector Machine (SVM) algorithms. Eq. (10) calculates ER, where C is a trained classifier, ρ is an untargeted attack, X is the test data. First adversarial inputs $\rho(x_i)$ are crafted and the classifier is tested against them. In the equation only the adversarial inputs which successfully fooled the model are being considered. So only the indices $I \in 1, 2, n$ where $C(x_i) \neq C(\rho(x_i))$ must be taken. Selecting attacks is a challenging task, and this work considers the success ratio and calculation speed of each attack to select Fast Gradient [9], Carlini & Wagner [27], and DeepFool [28] attacks.

$$ER(C, \rho, X) = \frac{1}{|I|} \sum_{i \in I} \frac{||\rho(x_i) - x_i||}{||x_i||} \tag{10}$$

3.4. Accountability Pillar

Accountability fosters trust in AI by documenting, validating, and notifying the creation, evaluation, and maintenance of ML/DL models. This work identifies the following metrics as relevant to evaluating the accountability of supervised ML/DL models. The following metric scores are calculated using predefined configurable thresholds that can be found in [14].

- *Train/Test Split*: measures the ratio between the number of samples used for training and testing. If the training and testing dataset do not contain enough representative data, the model will not perform well, it will not be able to generalize, and the predictions will not be reliable.
- *Missing Data*: evaluates how missing values of features of the training dataset are handled. If the model is trained on a dataset containing many missing values, the model will not be reliable.
- *Normalization*: evaluates if some models have been trained with normalized or non-normalized data. It helps mitigating the effects of outliers and ensures that features values are in the same range [29].
- *Regularization*: measures if the ML/DL model used generalization techniques during training. It avoids model parameters taking extreme values, and it is necessary to avoid memorization during training NN with millions of parameters.
- *FactSheet Completeness*: measures if the FactSheet includes all necessary information that stakeholders need in order to trust the model and its predictions. A FactSheet summarizes important metadata regarding the steps followed to train a model, purpose, algorithm, and data, among others.

4. Algorithm Quantifying Trustworthy AI

This section presents a novel, extensible, and parametrized algorithm able to quantify the trustworthiness level of supervised ML/DL models. Figure 2 shows the algorithm schema and lifecycle.

First, for each pillar, the algorithm computes the metrics explained in the previous section. For that, each metric receives as an input the *i)* training and testing datasets, *ii)* trained ML/DL model, and *iii)* FactSheet with the metadata of the training methodology. Then, the algorithm evaluates if the inputs fulfill the conditions of each metric. If so, each metric is independently calculated according to its formula and input data. Table 1 shows for each metric, its inputs, conditions, and output.

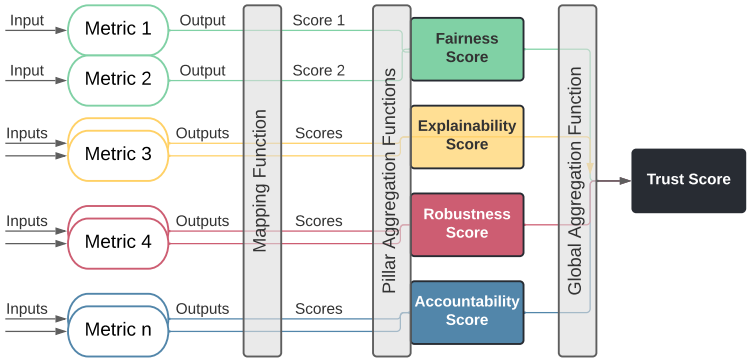

Figure 2: Design of the Trusted AI Algorithm

Table 1
Metrics Considered by Our Trusted AI algorithm

Pillar	Metric	Input	Condition	Output
Fairness	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 %
Explainab.	Algorithm 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	Testing Dataset, Model	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]

The metrics outputs cannot be interpreted as trust scores because they have different data types, scales, and meanings. Therefore, each metric output must be interpreted and translated into a standard trust score using a mapping function. The proposed trust score for all metrics ranges from one to five, where one corresponds to the worst score, and five represents the best score. The mappings from metrics outputs to trust scores are predefined according to good practices indicated in the literature. However, this process could involve some arbitrary decisions adding biases. To avoid it, the mapping function is parameterized and can be fine-tuned by stakeholders according to the data domain, metric, or scenario.

The next step consists of aggregating all the metrics scores of each pillar and calculating a score per pillar. The algorithm proposes a weighted approach where each metric has particular importance in the pillar score. It is up to discuss whether all metrics are equally important and how weighted they should be. Because of that, default weights for every metric are defined, but stakeholders can modify them according to the scenario characteristics.

Finally, the four pillar scores are aggregated into a global trust score, which is the return value of the algorithm. Computing the global trust score is done analog to calculating the pillars scores. Independent weights are assigned to each pillar, and the global trust score is the weighted average of each pillar. Since the importance of each pillar depends on the scenario, the predefined configuration of the algorithm (equal importance per pillar) can be modified by stakeholders. Algorithm 1 shows the pseudocode implementing the previous steps of the proposed Trusted AI algorithm.

Algorithm 1 Trusted AI Algorithm

function TRUSTED_AI($model, data_{train}, data_{test}, factsheet, config_{map}, config_{weights}$)
  $trust\_score \leftarrow 0$
  $pillars \leftarrow fairness, explainability, robustness, accountability$
  $pillars\_scores \leftarrow empty\ dictionary$
  for $p\ in\ pillars$ **do**
    $score_p \leftarrow 0$
    $metrics\_scores \leftarrow empty\ dictionary$
    $metrics \leftarrow get\_metrics(p)$
    for $m\ in\ metrics$ **do**
      $args \leftarrow config_{map}[m]$
      $metrics\_scores[m] \leftarrow get\_score_m(args)$
    for $(m \in keys, v \in values)\ in\ metrics\_scores$ **do**
      $w_m \leftarrow config_{weights}[m]$
      $score_p \leftarrow score_p + w_m * v$
    $pillars\_scores[p] \leftarrow score_p$
  for $(p \in keys, v \in values)\ in\ pillars\_scores$ **do**
    $w_p \leftarrow config_{weights}[p]$
    $trust\_score \leftarrow trust\_score + w_p * v$
  return $trust\_score$

5. Web Application

This section presents a Web-based application hosting the proposed algorithm and allowing stakeholders to calculate the trustworthiness level of supervised ML/DL models [15] in an intuitive way. To understand the application functionality, it is important to introduce the concepts of *scenario* and *solution*. Firstly, a scenario is the application context where supervised ML/DL models are created to solve a particular task. Classifying malware in IoT devices, or fraud in credit cards are two examples of scenarios. Secondly, a solution is a combination of *i)* the training & testing data, *ii)* a trained supervised ML/DL model, and *iii)* a FactSheet solving a given scenario task. Thus, the Web application allows stakeholders to:

- Create scenarios and upload solutions for existing scenarios.
- Calculate and graphically see the trustworthiness level of a solution.
- Re-calculate the trustworthiness level of a given solution according to customized parameters.
- Compare the trustworthiness levels of two solutions, explaining the meaning and results of each metric and pillar.

The Python framework *dash* 2.1.0 was used to implement the Web application backend and frontend components. The backend implements the algorithm and a database to store the input data needed by the algorithm. The frontend provides stakeholders with a graphical interface composed of the following four pages. The Appendix contains one screenshot of each page.

5.1. Web Application Functionality

The main functionality of the proposed Web application is organized in the following main pages: scenario, upload, analyze, and compare. Below, more details about them are given.

Scenario Page. This page allows stakeholders to create new scenarios with their descriptions.

Upload Page. Stakeholders use it to upload their solutions for a previously created scenario. In particular, the following aspects are needed:

- *Scenario & Solution.*
- *Description*: Brief description of the solution (optional).
- *Training & Testing data*: Two datasets (format: *csv* or *pickle*).
- *Protected Feature*: Protected features of the Training data (optional).
- *Protected values*: Protected values for the protected features (optional).
- *Target Column*: Column of the training dataset to be predicted.
- *ML/DL model*: Supervised ML/DL model of the solution (format: *pickle*).
- *FactSheet*: Methodological steps (format: *json*) with the following fields.
  - *Model Name* (optional);
  - *Purpose*: Supervised ML/DL model goal (optional).
  - *Domain*: Where the model is used (optional).
  - *Data*: Description of the data and the preprocessing techniques (optional).
  - *Model information*: Information about the model (optional).

- *Authors*: Model owner (optional).
- *Contact information* (optional).
- *Regularization*: Type of regularization technique (default: none).

Once the solution is uploaded, the previous information is stored in the backend database, and the Analyze Page is automatically loaded.

Analyze Page. Stakeholders use this page to quantify and visualize the trust score of their solutions. For that, a scenario and a solution must be selected, and the backend executes the proposed algorithm to compute the trusted AI score of that solution. Once the analysis is performed, an interactive report with the following two sections is displayed.

- *General information*: scenario, model information, performance metrics, and properties.
- *Trustworthiness score*: bar and spider chart with one score per pillar.
- *Interactive menu per pillar*: *i)* trust score for each pillar metric, *ii)* details about how each metric score is computed, and *iii)* reasons why specific metrics are not computed.

The configurations of mapping and weights are hidden and predefined. However, they can be shown and tuned by stakeholders. The information of both sections can be downloaded as a PDF report.

Compare Page. On this page, stakeholders can compare the trust scores of two selected solutions of a given scenario. They can also choose the configuration of the mapping functions and weights per metric and pillar applicable to the two solutions. The page displays a compact side-by-side view per solution with the following information: *i)* general information of the solution, *ii)* traditional performance metrics, *iii)* a bar chart with the trust score per pillar, and *iv)* one chart per pillar with the metrics scores.

6. Validation Scenario

To demonstrate the suitability of the trusted AI algorithm and Web application, this work considers a scenario focused on classifying cyberattacks affecting Internet-of-Things (IoT) devices. The proposed validation scenario aims to demonstrate the usefulness of having both traditional performance metrics and trustworthiness scores to select the best ML/DL solutions.

The scenario leverages a labeled dataset containing the normal behavior of some Raspberry Pi devices acting as radio spectrum sensors and the behavior of those devices under Denial-of-Service (DoS) and Data Leakage attacks. These device behaviors are modeled using the kernel events of the device using the *perf* Linux monitoring tool. In such scenario, two Support Vector Classifiers (SVCs) have been trained to detect and classify ongoing attacks. These models were trained using the same dataset but with different data normalization and splitting strategies. Then, the *IT Sec Incident Classification* scenario is created, and one solution per model (*Support Vector Classifier 01 (SVC_1)* and *Support Vector Classifier 02 (SVC_2)*) is uploaded to the Web application. Figure 3 shows part of the Analyze page for SVC_1.

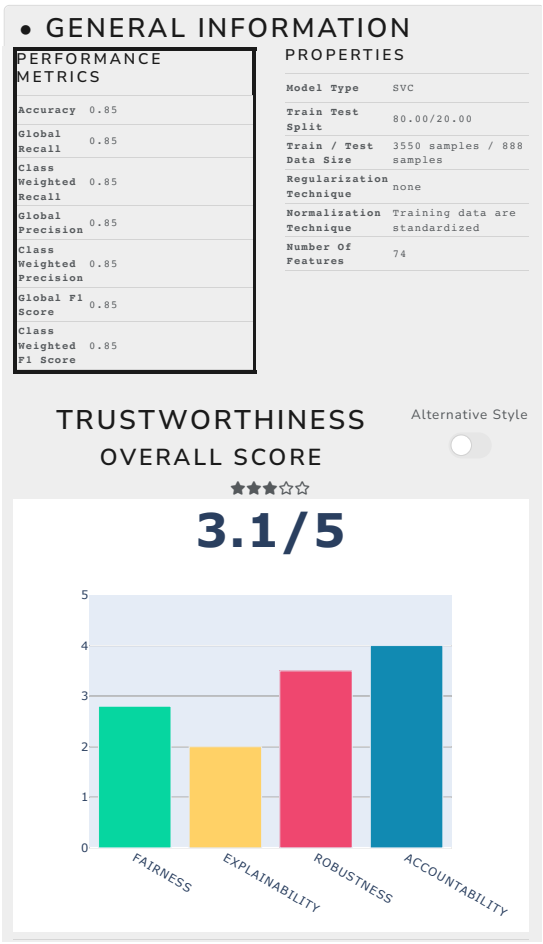

Figure 3: Analyze Page of the Web Application.

Figures included in the Appendix show that considering traditional performance metrics, SVC_2 is the best solution, achieving 0.87 global accuracy and F1-Score. However, the trustworthiness level also has to be analyzed to make a proper decision. In this sense, the trustworthiness scores show that the SVC_1 solution is the most trusted (3.1 vs. 2.5) since it is more robust and has a higher accountability score. Paying attention to the metrics scores of the accountability pillar, it can be seen that the scores for the train/test split and the normalization metrics are higher for SVC_1 (5 vs. 2 and 4 vs. 1, respectively). Indeed, the splitting strategy of SVC_1 is 80/20 (as suggested in the literature), while for SVC_2 is 90/10. In addition, SVC_1 solution standardized the training data, the opposite of the other solution. At this

stage, it is worth mentioning that the mapping function and weights per pillar and metric of the algorithm are not shown for the sake of simplicity and room. However, they can be found on the compare page of the Web application [15] once both solutions are selected.

7. Conclusions

This article introduced a novel, adaptive, and parameterized algorithm able to quantify the trustworthiness level of supervised ML/DL models with tabular data. The algorithm considers twenty-three metrics grouped into four pillars of trusted AI (fairness, explainability, robustness, and accountability). It combines the metrics outputs to compute a global trustworthiness score of supervised ML/DL models according to their training and testing data, model, and FactSheet. Also, the algorithm was deployed on a Web application where a cybersecurity scenario demonstrates the practical applicability of the algorithm. In conclusion, this work outlines the importance of considering not only traditional performance metrics but also the trustworthiness level of ML/DL models.

Future work plans to improve the limitations of the current solution with the support of unsupervised models and the inclusion of suggestions to improve the trustworthiness level of ML/DL models. Finally, the Web application will be redesigned to reduce computation time.

Acknowledgments

This work has been partially supported by *(a)* the Swiss Federal Office for Defense Procurement (armasuisse) with the CyberTracer and RESERVE (CYD-C-2020003) projects and *(b)* the University of Zürich UZH.

References

[1] ThinkML, The Growing Timeline of AI Milestones, https://achievements.ai, 2021.

[2] J. Wing, Trustworthy ai, Commun. ACM 64 (2021) 64–71.

[3] G. Saposnik, D. Redelmeier, C. Ruff, P. Tobler, Cognitive Biases Associated with Medical Decisions: A Systematic Review, BMC medical informatics and decision making 16 (2016) 1–14.

[4] R. Bellamy, et al., AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias, arXiv preprint arXiv:1810.01943 (2018).

[5] P. Sattigeri, S. Hoffman, V. Chenthamarakshan, K. Varshney, Fairness GAN: Generating Datasets with Fairness Properties Using a Generative Adversarial Network, IBM Journal of Research and Development 63 (2019) 3:1–3:9.

[6] S. Dash, O. Günlük, D. Wei, Boolean Decision Rules via Column Generation, in: 2018 International Conference on Neural Information Processing Systems, 2018, p. 4660–4670.

[7] A. Vijay, et al., The Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques, arXiv preprint arXiv:1909.03012 (2019).

[8] S. Lundberg, S. Lee, A Unified Approach to Interpreting Model Predictions, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 4765–4774.

[9] I. Goodfellow, J. Shlens, C. Szegedy, Explaining and Harnessing Adversarial Examples, arXiv preprint arXiv:1412.6572 (2014).

[10] E. Tabassi, K. Burns, M. Hadjimichael, A. Molina-Markham, J. Sexton, A Taxonomy and Terminology of Adversarial Machine Learning, NIST IR (2019) 1–29.

[11] M. Nicolae, et al., Adversarial Robustness Toolbox v1.0.0, arXiv preprint arXiv:1807.01069 (2019).

[12] J. Richards, D. Piorkowski, M. Hind, S. Houde, A. Mojsilovic, A Methodology for Creating AI FactSheets, arXiv preprint arXiv:2006.13796 (2020).

[13] M. Hind, et al., Experiences with Improving the Transparency of AI Models and Services, in: 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–8.

[14] J. Leupp, M. D. Demirci, J. Bauer, Trusted-AI Git Repository, https://github.com/JoelLeupp/Trusted-AI, 2022.

[15] J. Bauer, M. Demirci, J. Leupp, A. Huertas, Web Application to Quantify Trusted AI, https://www.trusted-ai.net, 2022. User: trusted-ai; pass: mp@csg2021.

[16] S. Feuerriegel, M. Dolata, G. Schwabe, Fair AI, Bus. Inf. Syst. Eng. 62 (2020) 379–384.

[17] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, C. Dwork, Learning fair representations, 30th International Conference on Machine Learning, ICML 2013 (2013) 1362–1370.

[18] I. Kloumann, J. Tannen, How we're using fairness flow to help build ai that works better for everyone, 2021. URL: https://ai.facebook.com/blog/how-were-using-fairness-flow-to-help-build-ai-\that-works-better-for-everyone.

[19] Fairlearn, Improve fairness of ai systems, 2021. URL: https://github.com/fairlearn/fairlearn.

[20] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, C. Dwork, Learning fair representations, in: International conference on machine learning, PMLR, 2013, pp. 325–333.

[21] M. Hardt, E. Price, N. Srebro, Equality of opportunity in supervised learning, Advances in neural information processing systems 29 (2016).

[22] D. Solans, B. Biggio, C. Castillo, Poisoning attacks on algorithmic fairness, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2020, pp. 162–177.

[23] M. Saarela, S. Jauhiainen, Comparison of feature importance measures as explanations for classifica-

tion models, SN Applied Sciences 3 (2021) 1–12.
[24] F. Y. et al, Interpreting and evaluating neural network robustness, in: 2019 International Joint Conferences on Artificial Intelligence, 2019, pp. 4199–4205. URL: https://www.ijcai.org/proceedings/2019/0583.pdf.
[25] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, L. Daniel, Evaluating the robustness of neural networks: An extreme value theory approach, arXiv preprint arXiv:1801.10578 (2018).
[26] H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, C.-J. Hsieh, Robustness verification of tree-based models, Advances in Neural Information Processing Systems 32 (2019).
[27] N. Carlini, D. Wagner, Towards Evaluating the Robustness of Neural Networks, in: 2017 IEEE symposium on security and privacy, IEEE, 2017, pp. 39–57.
[28] S. Moosavi-Dezfooli, A. Fawzi, P. Frossard, Deepfool: a simple and accurate method to fool deep neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.
[29] T. Jayalakshmi, A. Santhakumaran, Statistical normalization and back propagation for classification, International Journal of Computer Theory and Engineering 3 (2011) 1793–8201.

A. Images from web application

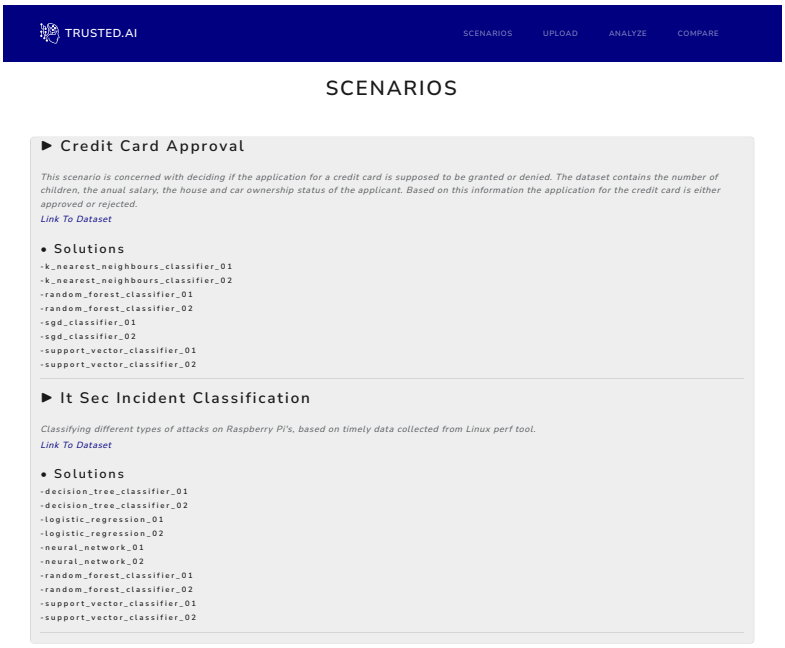

Figure 4: Scenario Page of the Web Application.

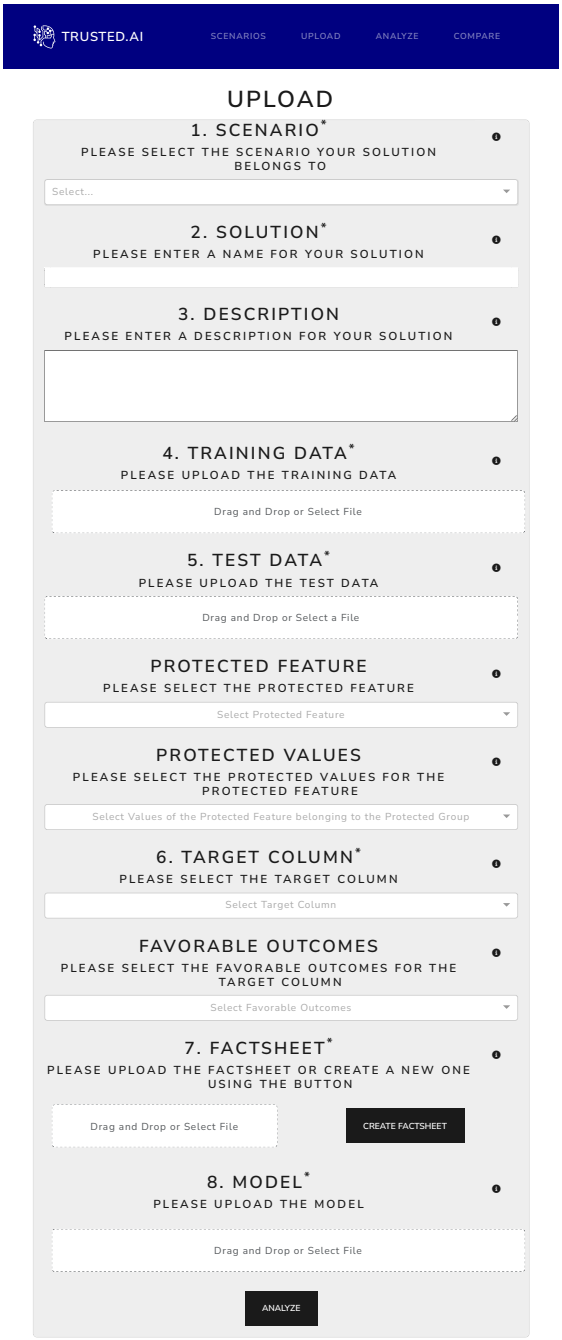

Figure 5: Upload Page of the Web Application.

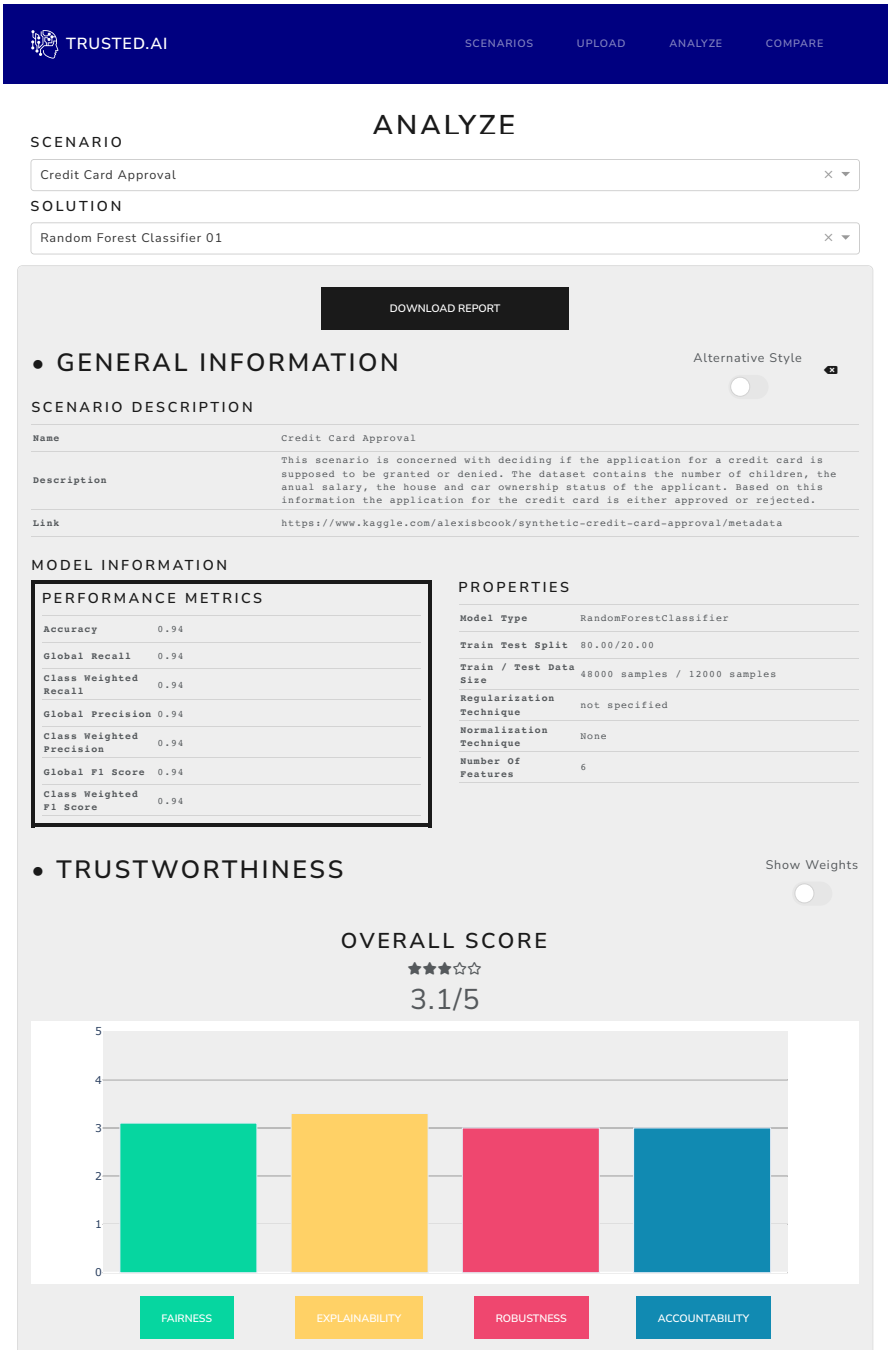

Figure 6: Analyze Page of the Web Application.

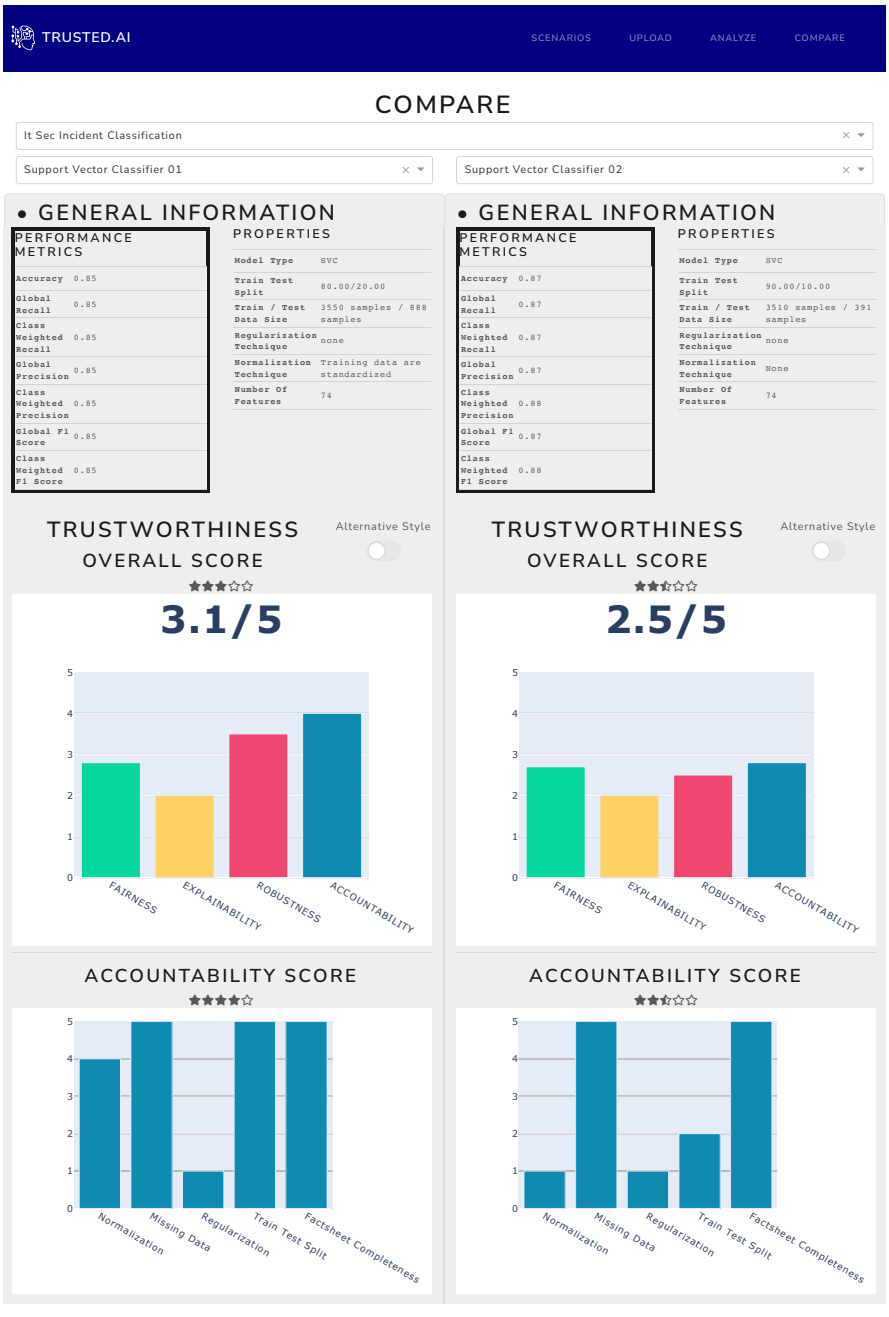

Figure 7: Compare Page with Performance Metrics and Trustworthiness Scores of Two Solutions.