**Survey of post-hoc explainable machine learning on tabular data**

M.S. Veldhuis

TU Delft

[m.s.veldhuis@student.tudelft.nl](mailto:m.s.veldhuis@student.tudelft.nl)

T. Abeel (supervisor)

TU Delft

[t.abeel@tudelft.nl](mailto:t.abeel@tudelft.nl)

**Abstract**

As black-box machine learning models continue to be applied to domain-specific problems, scientists will continue to search for explanations to uncover why these models came to such conclusions. These explanations might serve to improve trust in the model for stakeholders, to uncover novel information about the field, or to identify any mistakes the model made. Although recent reviews have focused on defining the field of explainable machine learning, providing it with taxonomies, or putting explanations of visual data at the forefront, we identify a lack of attention for tabular data and a more practical survey that details the available explanation methods. Therefore, this survey highlights the current techniques for obtaining post-hoc explanations of machine learning models on tabular data. We provide two overview tables that can serve as a starting point for researchers to select suitable methods. Furthermore, details about how the explanations are obtained, key results, and practical applications and evaluations of the methods are discussed.

**Keywords**

XAI, Explainability, Interpretability, Machine learning, Artificial Intelligence, Post-hoc, Black-box, Tabular data

# Introduction

*Context*

The field of Artificial Intelligence (AI) broadly captures an array of algorithms that execute tasks related to human cognitive ability such as problem solving and learning [1]. Beneath this umbrella term, Machine Learning (ML) is a technique of autonomous learning from experience, to predict outcomes on unseen data. Nowadays, ML methods are applied to a wide variety of problems, and have achieved high accuracy in various scientific domains such as medicine [2, 3], biology [4, 5], and finance [6]. One particular ML technique has especially gained popularity over the past years, which is Deep Learning (DL) [7]. Typical domains in which DL flourishes are computer vision [8], natural language processing [9], and speech recognition [10]. Some applications of deep learning have even surpassed human expert abilities [11, 12].

Despite the significant results of these ML- and DL models, one of their main flaws is that humans cannot easily discern exactly how the model functions and why it comes to its conclusions. Due to this “black-box”-nature of these algorithms, a demand for explanations has emerged [13-15]. Scientists want to know *why* the model made a certain decision, in addition to knowing the outcome. This is especially important when these models are used to influence real-life decisions, and actively have an impact on people’s lives [14-16]. For instance, no model should use an individual’s race to determine whether or not they receive a loan. Similarly, we would not want a DL model for detecting cancerous moles, to base its decision on noise present in the image.

*Contribution*

With this review, we intend to highlight techniques that have been developed for providing explanations of machine learning applications. The literature survey was limited to the past ten years, with a focus on actively maintained or well-established implementations. We aim our efforts at *post-hoc* explanations, which are explanations that have been generated after the underlying ML model has been optimized. This stems from the workflow where data scientists or domain experts first design a model with the aim of achieving good accuracy on solving a certain problem, and an explanation is sought afterwards [17], e.g. “*How can we explain why the model has achieved these results?*”. From the recent literature, there seems to be a focus on visual- and textual data [13, 18, 19]. Therefore, this paper aims to shed more light on *tabular data*, which is widely used in domains where explanations are important [16]. We consider tabular data to be information arranged in rows and columns, where rows represent instances, and columns encode certain features (or vice versa).

In short, the contribution of this paper is to *provide a survey of the recent techniques for obtaining post-hoc explanations for machine learning models applied to tabular data*.

The rest of the survey is structured as follows; First, this paper is grounded within a context of consensus definitions, concepts, and a taxonomy of explainable artificial intelligence as a whole. Secondly, explainable machine learning techniques will be described, with notable features highlighted, and categorized according to our taxonomy. This section is divided based on the types of ML models they can be applied to, starting with methods that can be applied to any model (model-agnostic), and other model-specific methods. The results are summarized in Table 1 and Table 2, which could serve as starting points for researchers looking for a technique to explain their ML model results.

# Background

*What?*

Explainable machine learning is a relatively underdefined subject, with a myriad of terminology being used differently and interchangeably [13, 20, 21]. An extensive analysis of the terminology within the field has been conducted, since no universally accepted terms were defined [13]. From their analysis, we aim to follow their definition of the umbrella term “XAI”: *“Given an audience, an explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand”* [13]. This definition is quite broad, but therefore captures the various other terms used to describe parts of the same idea.

Two of the most frequently mentioned terms besides XAI are “explainability” and “interpretability”*.* Quite often these terms are used interchangeably [21-23], while others point out the differences explicitly [15, 16, 19, 24, 25]. For instance, “interpretability” can be understood as a way to show the model internals in a human-understandable way [19]. While “explainability” aims to inform humans of the reasons why a decision has been made by an already interpretable model. In this way, explainable models are always interpretable, but the reverse does not have to be true [19, 24]. Rudin calls to make all models interpretable, meaning that the models itself provide explanations and to remove the term “explainability” altogether by arguing that this terms often just provides “summary statistics” [16].

Throughout this paper, *explainable machine learning* will primarily be used to describe the topic at hand. This is derived from the prior definition of the umbrella term XAI, yet refers specifically to ML applications. Note that this does not refer to the term “explainability”, which is defined differently among the literature as discussed above (alongside “interpretability”). As such, we intent to steer clear of these terms, either by choosing more specific terms, descriptions, or by using the more general explainable machine learning.

*Why?*

There are a number of main reasons that the literature discusses when it comes to the demand for XAI. The first of which is to increase *trust* in the model [13, 20, 21, 23, 26], meaning that the audience has confidence that the model acts as intended when given a problem [13]. This concept of trust also relates to how it performs in relation to humans; if it only makes mistakes that a human would make as well, it might be easier for humans to trust the model to be in control, knowing that it will not make mistakes that could be prevented by having a human perform the same task [20].

Explanations can also help manage and improve the model, or *identify mistakes* [20, 25-27]. For instance, to make sure that the model does not make decisions based on non-predictive features [20]. In a similar fashion, this can help ensure that predictions are fair and therefore not discriminatory in nature towards protected groups [13, 20, 22, 23].

Lastly, inspection of a model might uncover *useful information* about the problem that was previously unknown [15, 20, 27]. ML models might use features that humans have not previously identified as predictive for the issue at hand.

*How?*

The literature demonstrates various post-hoc strategies to make humans understand ML models. In general, to generate such explanations, we observe two main trends. The first trend is to leverage the *transparency* of simpler models to uncover information about the black-box algorithm, whereas the second trend follows *causation*; the dependency of the outcome on certain feature (values). These can be applied to explain the entire model (global), or a part of it (local). Refer to Figure 1 and 2 for examples of such techniques.

**Transparency** is linked to directly understanding how a model uses input features to form the output. A transparent ML model shows how to use features to form predictions based on its inherent structure (e.g. weights of a linear model). However, this study focusses on black-box models, which inherently are not transparent. Therefore, to obtain this type of explanations, one must *train* *a transparent surrogate model to approximate (part of) a black-box model* [13, 23, 25, 27]. Other equivalent terms are proxy-, student-, or mimic models. The mimic models are transparent in at least one of the following three levels, where each level subsumes the next.

*Simulatability* means that a human can reason about the mechanism of the model *as a whole* [13, 15, 20, 23, 25, 26]. If a human is given the input data, along with the parameters of the model, they should be able to come to a prediction that is in line with the model, within reasonable time [13, 20]. In this way, one could argue that a large decision tree with copious branches cannot be “simulated” in a human’s mind, while a single layer neural network can. This means that a model must both be simple and compact enough to fulfill this condition.

When a human can understand the separate *components* of the model (input, parameters, calculation), we speak of *decomposability* [13, 15, 20, 23, 25]. In this way, the previously discussed larger decision tree can be understood. A human can reason about the variables entering the tree, how a split is made based on their values, and how one can arrive at a leaf node. If the variables that went into the tree were highly engineered, this model would not fall into this category, since then the input component is not understood [13, 20, 23].

The third level concerns *algorithmic transparency*, which concerns just the *algorithm* with which the model is optimized [13, 15, 20, 25]. For example, humans can understand and reason about the optimal solution for a linear classifier. Or in our previous example, understand how a decision tree splits the feature space into different predictions. However, when applying deep neural networks, we have no such clear idea in our heads, as we might use an algorithm as gradient descent. This might seem quite intuitive, yet due to the heuristic nature of the hyperparameters chosen for this algorithm, humans cannot envision a specific optimal solution [13, 20].

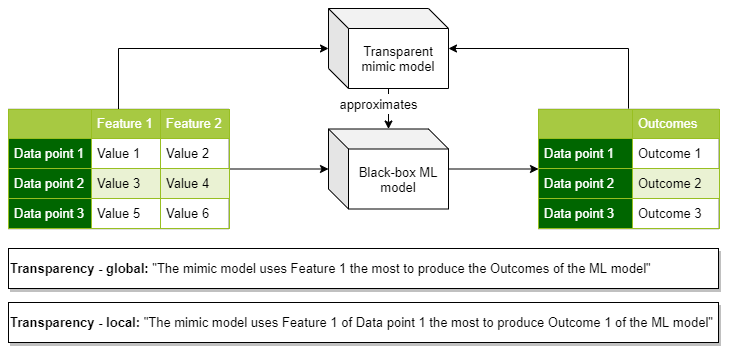


Figure 1: Simplified example of the transparency explanation technique. A mimic model is trained on the data and predictions made by the black-box model. Using the transparent mimic model, one can obtain information about how features are used to make predictions.

**Causation** is a more indirect type of explanation, which follows a set of causes that have affected certain predictions, observations or decisions. This comes down the *influence, importance or relevance of certain features to the outcome of the model*. [13, 15, 21-23, 25-28]. For this type of explanation, no transparent models are used. Instead, various algorithmic techniques are used that change or remove features and note the effect it has on the generated outcomes by the black-box model. It is important to consider that although these explanations are based on causation, ML models typically are optimized around similarity (correlation) [26, 29]. It is therefore important to inform the audience of such relations.

Unlike the concept of transparency, this idea of causation is not as uniformly described by the literature. Some refer to this concept as “interpretability”[15], “contrastive explanations”[25], or just mention the underlying concepts of “feature influence / relevance” as parts of a different kind of categorization [13, 21-23, 27]. Causation could further be split up into “causal attribution”, referring to a broader explanation, and “causal explanation”, which refers to selected cases [26, 28]. To keep the terminology more general, we opt to use the term of *causation*.

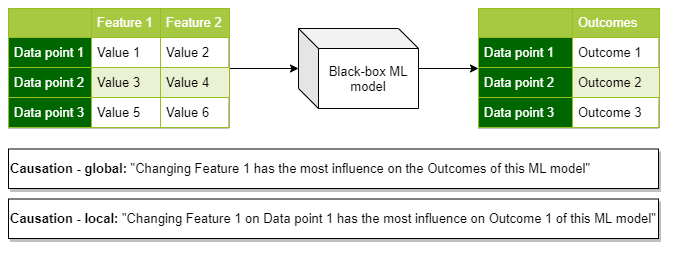


Figure 2: Simplified example of the causation explanation technique. No mimic model is used, instead the influence of certain features on the outcome of the model is calculated.

**Summary of definitions**

* *XAI*: “Given an audience, an explainable Artificial Intelligence is one that produces details or reasons to make its functioning clear or easy to understand” [13].
* *Explainable machine learning*: The subset of XAI pertaining to machine learning models.
* *Explanation*: The product of explainable machine learning.
* *Transparency*: A type of explanation that aims to uncover details about the functioning of the model by training a transparent surrogate model based on the black-box model.
* *Causality*: A type of explanation that aims to uncover which features are the most influential, important or relevant to produce a certain outcome of the black-box model.

*Assessment?*

As XAI continues to grow, multiple studies have uncovered the need for certain metrics to evaluate the quality of explanations [13, 19-24, 27]. No unified metric has been made to assess how good an explanation is, yet several aspects have been made clear that a good explanation must fulfil.

By producing an explanation, it is inevitable that some information is lost. Otherwise, we would present the entire model as it is (like a simulatable transparent model) [16]. The explanation must still be faithful, or *accurate* to the original model; be it for a local area of the model, a certain data point, or a higher-level concept the model follows [21, 25, 28]. One could also argue that this accuracy is based on how closely the explanation resembles the original model [24], or how closely it captures the relationships learned by the model [22].

From the perspective of the social sciences, some aspects have been uncovered to make explanations suitable for humans [25, 28]; First of all, humans seek explanations that are *contrastive*; For example, people ask why event P happened instead of some event Q [25, 28].

Secondly, humans select explanations in a biased manner. Humans reason according to context, which can include their profession, their experience, the domain of the data, etc. This context forms a bias, which must be accounted for by the explainer by providing a *fitting* explanation [22, 25, 28]. Unwanted bias can also be *counteracted*. For example, by providing different outcomes and their feature attributions, one could prevent anchoring bias that forms a skewed perception of the whole. For example, by showing only a correct outcome produced by the model, a human might gain too much trust in the model. By showing both correct and erroneous outcomes, we can be more nuanced [16, 26]. From this selective perspective, it is also important to note that explanations should be *small enough* [23], as humans tend to pick only a few key factors rather than consider every important part of the explanation [28].

Third, explanations are social [25, 28]. Humans explain concepts to each other in interactive conversation. In this way, a human can clear up any uncertainties by asking more about a particular point of interest. Therefore, explanations could use *interactivity*.

Finally, it is noted that *probabilities are not as important as causal links* [25, 28]. Statistical relationships are not seen as a satisfying explanation, without being accompanied by its underlying causes [28].

Other surveys have recognized the value in having humans directly assess explanations [24, 27]. For example, a metric called the System Causability Scale was developed to determine if explanations are useful [29]. It consists of a set of questions presented to humans to evaluate if explanations are suited towards a certain purpose.

**Taxonomy**

Based on the literature, the taxonomy in Figure 3 shows a categorization of explainable machine learning. Among the literature surveys that were considered in this review, most label the explanation techniques with similar concepts. Alternative taxonomies exist, such as one targeted specifically towards scientific discovery based on “transparency”, “interpretability”, and “domain knowledge” [22]. However, they still acknowledge some common terms regarding the types of models to explain, explanation approach, scope and audience. This shows that there seems to be a consensus among these definitions.

When looking at the *types of models* that are to be explained, a distinction is made similarly in the literature between *transparent* and *black-box* models [13, 16, 19-23, 25, 27]. Transparent models are defined as was discussed prior; they must fulfil one of the three levels of transparency. If a model does not satisfy any of these conditions, it is labelled as black-box, which requires *post-hoc* explanations. Transparent models might be called (intrinsically / inherently) interpretable / transparent, but the same type of model is depicted. Note that the models are categorized by their typical use. A Neural Network (NN) with a single neuron might be considered transparent, yet in practice, most NNs are not implemented as such.

Explanations have a certain *scope*. Either they refer to the entire model and the data that it represents (*global*), or to specific parts of it (*local*) [13, 16, 20-25, 27]. Local explanations can for example pertain to describing a decision boundary around a certain area in feature space, show similarly classified samples, or provide information about a specific instance in relation to the complete dataset.

Furthermore, *explanation approaches* can be categorized to be either *model-specific* or *model-agnostic*. Model-specific approaches can only be applied for one type of model, while model-agnostic techniques can be applied to any ML model [13, 21, 23, 27]. Model-specific methods leverage specific parts of the internal model to create an explanation. For example, a NN-specific method that is based on explaining neurons, cannot be applied to a tree-based model since it does not have neurons.

Not often directly referenced in the literature is the *audience*. We explicitly make the distinction between *data scientists* and *domain experts*. Where explanations targeted towards data scientists are often used to compare the performance of ML models in an intuitive way, explanations for domain experts are mainly concerned with the goals discussed in this paper so far.

Lastly, we only make the distinction between explanations based on *transparency*, and ones based on *causation*. This means that black-box models that are explained using a proxy model fall under transparency, while those who do not, leverage causation. We have chosen not to make a further distinction for the techniques used to produce explainable ML models, or the results than stem from it. This is because it seems that the techniques and results do not have consensus categorizations among the literature yet. For example, a technique called LIME is classified by one survey to be a “local explanation” [13], though it also falls under “explanation by simplification” [13, 20], and fits their definition of explanations based on “feature relevance” [13]. Others define LIME to be a “local surrogate model” [23, 27], or just a technique to produce a “visualization” [27]. Even from this single example, it quickly becomes clear that various researchers use different definitions to communicate the same concept, or use the same terminology to describe something different. For this reason, a more specific categorization of techniques is not present in this taxonomy.

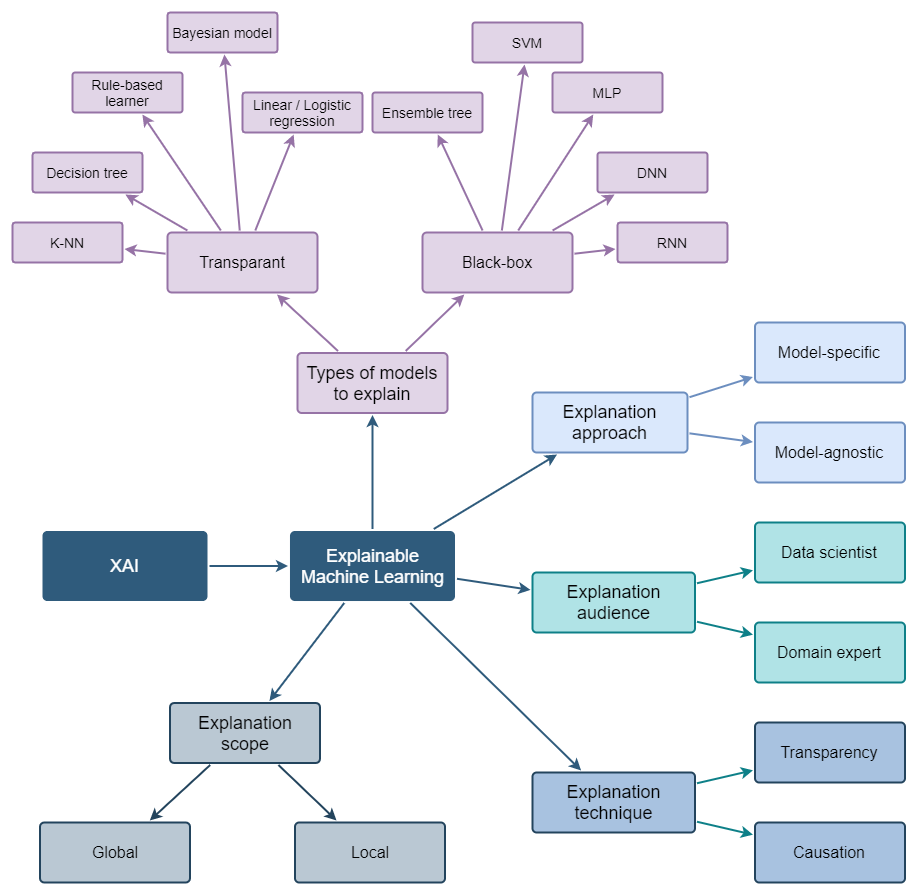


Figure 3: Taxonomy of explainable machine learning according to well-defined concepts consistently found in the literature, with the addition of Explanation technique according to our definitions of Transparency and Causation.

In this survey, we discuss the specific techniques and results structured by the explanation approach. Note that we focus on explanations of **black-box** models on **tabular data** for an audience of **domain experts**. We therefore omit papers discussing systems aimed towards data scientists that show explanations solely for the sake of inspecting the parameters of different ML models, or comparing their performance to each other. Furthermore, transparent models will only be discussed in the context of using it as a proxy model trained on the black-box model to be explained.

# Post-hoc explainable machine learning

This section will describe the various implementations available for the generation of post-hoc explanations that are appropriate to tabular data and various machine learning applications. Section 3.1 discusses model-agnostic techniques that can be applied to any model, while section 3.2 delves into the methods that were created with a certain type of ML model in mind. Each section will have their own overview table in which the techniques are classified according to the taxonomy of Figure 3, and provides links to any available code. In this way, the reader can select a method appropriate to the type of black-box model, preferred explanation scope and technique from the table. Any details regarding appropriate data features, applications, and any evaluations can be found in the survey text. Section 3.3 highlights some example results that are often produced by the discussed techniques.

## 3.1 Model-agnostic approaches

Model-agnostic techniques do not make any assumptions about the type of black-box model that they are attempting to explain. As such, they are appropriate to most applications which makes them quite popular. Table 1 contains an overview of the techniques that are discussed in this section.

|  |  |  |  |
| --- | --- | --- | --- |
| **[Paper(s)]: name** | **Scope** | **Technique** | **Code** |
| [30]: LIME | Local | Transparency | https://github.com/marcotcr/lime |
| [31]: Anchors | Local | Causation | https://github.com/marcotcr/anchor |
| [2, 3, 32]: Kernel SHAP | Global\* | Transparency | https://github.com/slundberg/shap |
| [33-36]: -, PALM, BETA, Distill-and-Compare | Global | Transparency |  |
| [37]: RuleMatrix | Global | Transparency | https://github.com/rulematrix/rule-matrix-py |
| [38]: explAIner | Global | Transparency | https://explainer.ai/system/explainer/\*\* |
| [39]: PD plots | Global | Causation | https://scikit-learn.org/stable/modules/partial\_dependence.html |
| [40]: ICE plots | Global | Causation | https://CRAN.R-project.org/package=ICEbox |
| [41]: ALE plots | Global | Causation | https://CRAN.R-project.org/package=ALEPlot |
| [42]: Feature permutation | Global | Causation |  |
| [43]: MMD-critic | Local | Causation | https://github.com/BeenKim/MMD-critic |
| [44]: Influence Functions | Local | Causation | https://github.com/kohpangwei/influence-release\*\* |
| [45]: Counterfactuals | Local | Causation | https://github.com/susanne-207/moc |
| [46]: What-If Tool | - | - | https://github.com/pair-code/what-if-tool |

Table 1: Overview of the model-agnostic explainable machine learning techniques used by the papers discussed in this section. They are classified according to their scope and technique. If code is available, a link is provided as well. Some techniques are grouped together according to scope and technique when no implementation can be found (e.g. [33-36]). The techniques marked in blue seem to have obtained the most support in research and are either well-established implementations or have an actively-maintained code base.

\* = can also be used for local explanations

\*\* = seems to be no longer under active development or maintenance (last update > 2 years ago)

**LIME**

Perhaps one of the most well-known model-agnostic explanation techniques is LIME [30]. This technique creates a *transparent* model that is faithful to the black-box classifier in a certain region of the feature space. The process of forming a transparent model is based around creating new samples by perturbing a single data point in a certain “neighborhood”. These new perturbed samples are run through the black-box model, and the changes in predictions with respect to the original instance are measured, weighted by the proximity to the original instance. Based on this new data, a linear model is learned to fit the predictions of the black-box model. This can be any linear transparent model, such as logistic regression or a decision tree. At the same time, the complexity is minimized by keeping the number of features low. The output of LIME for tabular data is a list of understandable input features, even if the black-box model uses these in a more highly engineered form. Based on the weights in the linear model, the most important or influential features are placed at the top. As such, LIME is quite a simple approach which is easily understood. It also provides the indication of fidelity, showing how well the model explains the predictions made by the black box model [30].

Because LIME is only *locally* accurate, it cannot be used to explain an entire model. Since the number of features used is low as well, this might not be a good choice to explain a prediction in detail. Moreover, the perturbations are sampled in a way that does not take into account the correlation between features [30]. This means that unlikely samples might be produced, or certain features are accredited too much importance due to a correlation effect with other features. Therefore, the user must evaluate if the explanations seem valid based on their knowledge of the problem. Lastly, LIME fits a linear model to the local neighborhood, even though the decision boundary might not be linear around that area, thus causing an incorrect model.

LIME has been shown to be vulnerable to adversarial attacks [47]. This is because the way that LIME perturbs the selected point can lead to samples that do not fit the original distribution of the dataset. An adversary classifier could thus identify the samples generated by the perturbation, and classify those in a different way than those of the original dataset. In this way, a classifier can effectively hide its original bias from the explanation model. For example, the black-box model might make decisions based on race, while effectively hiding this bias from the explanations created by LIME.

**Anchors**

The authors of the original LIME paper have since come out with an improved version of LIME called Anchors [31]. Their main aim was to improve the notion of the “neighborhood” for which the explanation holds. Anchors are decision rules, or feature values, that “anchor” a prediction in a *certain region* of the feature space. This means that if such an anchor holds, the prediction remains the same with high probability, even if other features are varied. Anchors are created based on the same data perturbation approach as LIME, but instead has clear boundaries, or coverage, for which this explanation holds. These anchors are not learned by generating a surrogate model. Instead, the decision rules are optimized by an exploration-exploitation technique using the generated perturbations, based in reinforcement learning. In this way anchors do not exploit the transparency of simple ML models, but they leverage *causation*.

This approach improves significantly on LIME since it is able to define for which area an explanation holds with its notion of coverage, and can handle non-linearities well due to not fitting a linear model. However, since the subsampling procedure remains the same, we suspect that Anchors are also susceptible to the adversarial approach used to attack LIME [47].

**SHAP**

Another popular technique for generating explanations is SHAP; or SHapley Additive exPlanations. SHAP is based on calculating Shapley values for inputs that were used in a ML model [32]. Shapley values originate from game theory, and define how much features contributed to the value of a certain prediction, in comparison to the average prediction. This is achieved by calculating the average difference in the value of predictions when including and omitting a certain feature value in increasingly large sets of other features. Because one cannot simply omit a feature in a trained ML model that uses tabular data, the feature is set to a sampled value. For instance, when predicting whether or not someone will be granted a loan, the feature value of “female” is investigated in a set with no other features, a set with 1 other feature, and a set with 2 other features. For each set, the difference in value (probability) when including “female”, versus a sampled value, is noted. This value is then averaged over all sets. For example, “female” could contribute -0.2 on average over these sets. The comparison to the average prediction of the dataset, or a subset of the dataset allows for *contrastive explanations*.

SHAP can be used to create *global* explanations by calculating Shapley values for all possible combinations of feature sets, though this is a rather computationally intensive procedure [32]. To handle this undesirable property, SHAP includes a wide variety of implementations for the approximation of Shapley values, optimized for specific models such as linear models, tree-ensembles [48], and DNNs [32]. We will discuss each of these versions in their model-specific sections.

Kernel SHAP is the default SHAP implementation which approximates the Shapley values from all possible combinations, using a linear model [32], leveraging its *transparency*. This linear model is fitted similarly to LIME, except that the weight for each sampled prediction is computed with the SHAP kernel. This kernel assigns the largest weights to the smallest and largest sets. The intuition behind this being that we would like to study features in isolation, as well as if it was excluded from a large group of other features. These sets hold the largest discriminatory power.

The same paper that detected the vulnerability of LIME to an adversarial classifier, applied the same strategy to SHAP [47]. They found that SHAP was less vulnerable to exploitation than LIME, but still found significant ways to avoid exposing the black-box classifier’s true bias. This could be because sampling approach used by SHAP suffers from the same property as LIME, as it does not account for correlated features [32]. Perhaps the reason why SHAP is less vulnerable than LIME could be attributed to a local accuracy property. This property ensures that the difference between the given prediction and the average prediction on the background distribution is fully determined by the feature attributions. In this way, information about the original data distribution is kept, therefore undermining the adversarial approach [47].

SHAP was put to the test by anesthesiologists during surgery to assist in predicting the risk of hypoxemia and explaining any risk factors, both pre-surgery and in real time during surgery [2]. The underlying black-box model was a gradient boosting machine, which was trained on electronic medical records of past surgeries. After training the model, Kernel SHAP was applied to compute the Shapley values. They reported that without the ML model and accompanied explanations, anesthesiologists anticipated 15% of hypoxemia events, in contrast to 30% when the model and explanations were available. They also report to have “shed new light on physiological relationships” [2], as a new marker for hypoxemia was found. It seems that the explanations helped provide *more information* to the anesthesiologists in real-time as well as after the surgery. Please note that this study was performed by the same authors that created SHAP.

In a different study, Kernel SHAP was applied to another prediction of mortality in the ICU based on both static and dynamic clinical features [3]. Here the black-box model was an LSTM. As the paper only makes reference to the original SHAP paper, we assume they used the Kernel SHAP implementation. The explanations that were obtained were compared to a handcrafted set of features that are used to assess physiology. In their results, they state that the obtained explanations were in line with what was expected in comparison to the handcrafted feature importance. The only notable difference was that the SHAP feature importance showed that a feature can drive the prediction in either direction based on context, whereas the handcrafted model did not show such flexibility. Though the paper did not detail any information about the quality of the explanations, it did demonstrate that using explanations allows for more complex models to be used as *decision support tools*, while they are still in line with domain knowledge.

**Global surrogate models**

There are various other studies that have attempted to create similar types of explanation frameworks to LIME, Anchors or SHAP, though none in particular have left an equally impactful mark on the field. These approaches can be summarized as *global* surrogate / student / mimic models, since they approximate a black-box model by a simpler *transparent* model [33-37]. These transparent models can be anything from linear models, to decision trees and -rules, as long as they fit one of the transparency levels described prior. The surrogate model is trained on a dataset with labels produced by the black-box model. Most methods partition the dataset in various ways to obtain better representations of the subsets in the original data, and therefore better surrogate models of the original black-box models. Usually, the surrogate model is evaluated on how well it performs in relation to the black-box model. This is often called an agreement rate or -score, or is modelled using an R-squared value (variance of the black-box model explained by the surrogate model).

For instance, a method called BETA has creates *global* nested decision sets for any black-box model, proving to uphold a set of optimality guarantees [33]. The decision sets are obtained by grouping instances of the data together using a local search of feature similarity, and then fitting those groups to a linear model. Note that other studies use almost identical set-ups, but employ different algorithms. For example, one study groups instances using the EM-algorithm, and then fits those groups using decision trees as the transparent model [36]. We continue to focus on BETA, since they report to have outperformed LIME on the number of rules necessary to obtain an 85% agreement rate [33]. They show that BETA requires only 5 rules, in contrast to 20 for LIME. However, these results do not specify that these 5 rules are the outer rules of the decision set, which each could include any number of inner rules. Moreover, in their user study which showed increased *human accuracy* in answering questions related to feature importance, they did not include the comparison to LIME, only to simple decision rule algorithms. In this way, it is difficult to see the added value of this approach, and whether it holds up to the current state-of-the-art methods. They did provide a comparison of the same method with-, and without an interactive interface in their user study. The time required to answer the questions was almost halved by using the interactive interface, further supporting the notion that *interactive explanations* have added value in practice.

A similar approach called PALM uses a two-part surrogate model to approximate a black-box model. The first part consists of a decision tree that captures the structure in the data, while the second part is consists of fitting sub-models to the different parts of the data [34]. This is constructed similarly to BETA’s outer and inner decision rules. The partitioning of the data can be adjusted as a trade-off between how specific the partitions are, and how closely the surrogate model represents the black-box model. The approach used is based in an optimized K-means algorithm, except each cluster represents a sub-model with points assigned to it that are best explained by this sub-model. The authors mainly aim the interactivity to be targeted towards *debugging faulty predictions*; the user can identify in which sub-partition, and therefore in which sub-model the problem lies, and adjust the model accordingly. It is interesting to note that the authors found that often faulty predictions are predicted by specific sub-models, therefore identifying which parts of the subspace are more difficult to model with a simple model. They provide an agreement score of 90% for 16 partitions of two different datasets, using a NN model as the black-box.

An approach called Distill-and-Compare [35], highlights how many proprietary black-box models do not allow access to the training data, thus making perturbation-based mimic models infeasible as we cannot measure the changes in the model when we do not have the original training data. Instead they use only the input-output relationships by inserting new data into the black-box model, to train a student model on, which minimizes the difference between the mimic model- and black-box predictions. They compare this mimic model with a model of the same class, trained solely on the input-output pairs. In this way, the authors aim to find discrepancies in feature relevance between a mimic model trained to match predictions of the black-box model, and an outcome model that is trained on input-output pairs only. This approach thus combines predictions made on the data, and mimicking a black-box model.

In a similar fashion RuleMatrix extracts rules from a black-box classifier. But their more impactful contribution is a well-constructed visual interface to present decision rules in a user-friendly way, and allows for interactivity [37]. Another visual interface called explAIner demonstrated the value in providing users with interactive explanations of ML models [38]. However, their user study showed that the way in which explanations of ML models were presented, was not easy to understand. In short, these studies do present the importance of creating an *interactive interface* to help humans understand ML models.

This set of global transparent explanations is often quite intuitive. However, one must be careful with understanding that a surrogate model explains the black-box model, not the underlying data. If the black-box model is wrong, the surrogate model is wrong. The Distill-and-Compare paper highlights this. Another aspect to consider is that the surrogate model is simple, thus has high bias, therefore it could fit one subset of the data quite well, while being wrong for another subset.

**PD plots**

Contrary to these global transparent applications that leverage the simplicity of surrogate models, there are several simpler alternatives that focus on using *causation* to visualize the influence, importance, or relevance of certain features on the outcome.

The simplest could be considered the Partial Dependence Plot (PDP) which shows how features affect the model predictions [39]. Partial dependence is calculated by changing the feature under survey to certain values for all the input data, while keeping the original values for the other features fixed, then noting the *average* prediction value. In a PDP, the different values of the feature under survey are plotted against the prediction values. In this way, PDPs are quite intuitive, it shows exactly how the average prediction changes when a certain feature is changed. This therefore shows *causal* relationships of the model.

However, this partial dependence relation only holds if the feature is not correlated with other features [39]. If two features are highly correlated, though only the second feature has an impact on the predictions, a PD plot of the first feature would still show an effect on the predictions. This is due to the correlation of the first feature with the second feature. Another downside to PD plots is that we might sample feature combinations that are unlikely for the feature under investigation. Similarly, it is important to visualize the distribution of the feature values in the original data, so that conclusions can be made about the validity of the prediction. If certain values of the feature show to influence the predictions strongly, yet there are only a few of those values in the data, we cannot place a lot of trust in those predictions. Lastly, it is important to note that PD plots show average results. This means that if a feature value shows a positive correlation with the outcome on half the data, while on the other half, a negative correlation, the effects cancel each other out. In the final PDP, the feature thus shows up as having no effect on the outcome. This type of heterogeneous relationship of features to the outcome can thus not be uncovered with PD plots.

**ICE plots**

Where PD plots show a global averaged effect, Individual Conditional Expectation (ICE) plots visualize the effect of a feature change for single instances [40]. So instead of a single global line, you obtain one line per data point. In effect, a PD line is the average of the ICE lines. With ICE plots, heterogeneous relationships can be visualized since we can observe the different effects for each instance individually. Because these lines are all stacked in one plot, variation might be difficult to discern. To mitigate this issue, the ICE plot can be centered at a certain base value, to demonstrate the effects in relation to this anchor point. This shows the difference in prediction values in comparison to the chosen base prediction.

**ALE plots**

As a possible solution to dealing with correlated features, which PD- and ICE plots do not handle well, Accumulated Local Effects (ALE) plots were designed [41]. With ALE, these correlations are accounted for by averaging and accumulating local changes to the predictions over the conditional distribution of the two features. In general, it might be better to use ALE plots when features are correlated. Furthermore, ALE plots have a significantly faster computation time. However, when computation time is not as crucial, PD plots might be preferred, as they can be coupled with ICE plots to uncover possible heterogeneity.

**Permutation feature importance**

Besides the previous visual interpretations of feature influence, there exist several techniques that measure feature importance based around the idea of permuting features artificially [42]. In short, such methods work by perturbing the feature in question, and measuring the increase or decrease in the model’s error. This might remind the reader of LIME and SHAP, as they also permute feature values. However, no additional model is trained on the permuted features, therefore only leveraging *causation* by noting the change on the outcome of the black-box model. Also, these methods work on a *global* scale, since they measure the influence on the model in general. This technique is quite intuitive to understand as the change in error of the black-box model when destroying the feature in question. By calculating the ratio between the original error, and the error after performing permutations, a measure of feature importance is obtained that can be compared among different models and problems with various ranges of errors. Note that when a feature is correlated with another, this interaction effect is removed as well when calculating the new error rate.

**Prototypes / Criticisms**

We end this section with example-based explanations, which highlight *certain instances* in the dataset. We will discuss prototypes and criticisms [43, 44], and in the next section counterfactuals [45, 49].

A prototype is a sample that is well-represented by the model for a certain class, while a criticism is poorly represented. These types of explanations are well-suited to visual data, since images can be understood by the user at a glance. For instance, it might be intuitive for a human to understand why an image of a cat with a hat is poorly-represented by a model which is trained to classify cat pictures; Since cats are usually not pictured with hats on, the model therefore classifies such criticism samples based on little data. A method that evaluated their prototype and criticism algorithm called MMD-critic, demonstrated that humans more easily understand the underlying models when presented with such examples on visual data, though no experiments were done on tabular data [43]. For tabular data, an instance might consist of a multitude of features that are not as easily understood by a human and therefore provide less information on what this means for the model.

**Counterfactuals**

Counterfactual explanations seem to be an example-based explanation that is inherently easily understood for tabular data. Counterfactuals stem from the need from humans to want *contrastive* explanations. Such counterfactual explanations can be used to explain *individual instances*, adding a statement of how the outcome could be changed if different conditions were true [45, 49]. An example of this could be: “You were denied a loan because your annual income was £30,000. If your income had been £45,000, you would have been offered a loan.” [49], or “This cat would have been adopted from the shelter if it was younger”. Such an explanation could be useful to inspect certain decisions, for instance to make sure that no discriminatory features were used such as gender or race. Another use case could be to find features on which to improve. For instance, in the first example, the person in question could work towards getting a higher income to get the loan.

In essence, most counterfactual explanation algorithms optimize a loss function that takes into account a certain instance, its original outcome, and the desired outcome [45, 49]. This loss function is minimized on the notion of a certain measure of distance between the original instance, and changes in feature values to obtain the desired outcome. A counterfactual explanation should ideally be the closest to the original scenario, therefore this notion of closeness must be defined based on the used features. The pitfall of counterfactuals is that there are several changes to features that will lead to the same outcome, and therefore either a selection must be made, or multiple options should be presented. Counterfactual explanations seem to be an active research direction, where different algorithms are being explored to obtain the most suitable counterfactual instance [45].

Counterfactual explanations are intuitive [28]. The ease of understanding could be attributed to that fact that these are explanations for a single instance, and directly translate the effect of feature values. It also does not require access to the model, retraining or access to the training set. No assumptions are made about local areas or decision boundaries. As such, they could be interpreted similarly to Anchors, except that these show feature values for which the outcome changes, rather than for which the outcome stays the same. One implementation has shown to scale well with many features such as on The German Credit Risk dataset [45].

**Google visualization tool**

Several of the discussed techniques are accessible via a visualization platform developed by Google called the What-If Tool [46]. Trained classification- and regression models can be loaded into this tool, which allows the user to explore various statistics on the data, the model, and the predictions made. Several of the techniques that we have discussed so far have been incorporated such as Counterfactuals, PD plots, and the visualization of any local feature attribution technique such as SHAP.

## 3.2 Model-specific approaches

Model-specific explanation techniques often leverage some structural part of the model which they are trying to explain. Therefore, this section is divided based on the types of ML models for which explanations are generated. The types of models that we encountered most frequently are tree ensembles, Support Vector Machines (SVMs), and (deep) neural networks ((D)NNs). In Table 2, an overview of the model-specific techniques discussed in this section can be found.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **[Paper(s)]: name** | **Model** | **Scope** | **Technique** | **Code** |
| [48, 50]: TreeSHAP | Boosted trees | Global\* | Causation | https://github.com/slundberg/shap |
| [51]: Feature contribution | RFs | Global\* | Causation | https://rdrr.io/rforge/rfFC/ |
| [52]: Forest floor | RFs | Global | Causation | https://CRAN.R-project.org/package=forestFloor\*\* |
| [53]: - | Trees | Global | Transparency |  |
| [54]: inTrees | Trees | Global | Transparency | https://CRAN.R-project.org/package=inTrees\*\* |
| [55]: - | Trees | Local | Causation |  |
| [56]: RF classCenter | RFs | Local | Causation | https://CRAN.R-project.org/package=randomForest |
| [57]: Feature tweaking | Trees | Local | Causation | https://github.com/gtolomei/ml-feature-tweaking\*\* |
| [58]: ALBA | SVMs | Local | Causation |  |
| [59]: SQRex-SVM | SVMs | Local | Causation | https://github.com/summer2sama/LIGO-SQRex-SVM\*\* |
| [60]: DeepLIFT | DNNs | Local | Causation | https://github.com/kundajelab/deeplift |
| [32, 61]: Deep SHAP | DNNs | Global\* | Causation | https://github.com/slundberg/shap |
| [62]: Integrated gradients (+ SHAP) | DNNs | Global\* | Causation | https://github.com/ankurtaly/Integrated-Gradients  https://github.com/slundberg/shap |
| [63]: DeepRED | DNNs | Global | Transparency | https://github.com/cansyl/DEEPred\*\* |
| [64, 65]: - | DNNs | Global | Transparency |  |
| [66]: DkNN | DNNs | Local | Transparency | https://github.com/tensorflow/cleverhans/tree/master/cleverhans/model\_zoo/deep\_k\_nearest\_neighbors |
| [18]: GRACE | NNs | Local | Transparency | https://github.com/lethaiq/GRACE\_KDD20 |

Table 2: Overview of the model-specific explainable machine learning techniques used by the papers discussed in this section. They are classified according to their scope and technique. If code is available, a link is provided as well. Some techniques are grouped together according to scope and technique when no implementation can be found (e.g. [64, 65]). The techniques marked in blue seem to have obtained the most support in research and are either well-established implementations or have an actively-maintained code base.

\* = can also be used for local explanations

\*\* = seems to be no longer under active development or maintenance (last update > 2 years ago)

### 3.2.1 Tree ensembles

Tree ensembles combine multiple decision trees with a certain algorithm [67]. In a decision tree, an instance is evaluated from the *root* node of the tree, via internal nodes, until a decision is made at a *leaf* node. At each internal node, a certain feature value is evaluated, which decides the path which the instance takes through the tree. This hierarchical structure of if/else rules makes this model quite easy to understand, given that the tree is not too big, and that the features used are human-understandable. In practice, ensemble trees are more common which are inherently more complex and larger as well. There are two main classes of ensemble trees. The first is a Random Forest (RF) which averages the decisions of multiple decision trees which are trained on different feature-, and sample subsets of the data. The second is a Gradient boosted tree, which chains shallow decision trees to correct the previous trees’ mistakes to obtain a decision. Explanations for ensemble trees often leverage the hierarchical structure in which these models are built to obtain a sense of which features are the most telling.

**TreeSHAP**

For understanding how TreeSHAP works, we recommend first reading about Shapley values and the general implementation of SHAP in the model-agnostic section. TreeSHAP is optimized for usage on boosted tree ensembles, which guarantees consistent results of *global* feature importance [48]. In comparison to the default implementation of SHAP (KernelSHAP), TreeSHAP is much faster and does not use a linear model to approximate all combinations of feature subsets, it actually arrives at an exact solution. The algorithm leverages the structure of trees where paths can be discarded if they do not include the features in the subset that is currently being evaluated. If for example we are investigating subset of features {x1, x2} and {x1, x4}, while the first split is made on feature x1, these two subsets are followed down the same branch simultaneously, without the need to trace through the tree twice. Another positive feature that TreeSHAP brings, is that it takes into account interaction effects, meaning that no unlikely combination of features is made.

The authors’ user study showed that human participants would score feature attribution mostly the same as TreeSHAP does, showing some support for its consistency with *human scoring*. As such, TreeSHAP has been incorporated into frameworks such as XGBoost and scikit-learn. As computation time scales with tree depth, it is ideal for running on boosted trees as they are quite shallow. It can be applied to other (deeper) ensemble trees such as a random forest, but it will perform slower.

TreeSHAP was applied to a classification problem where patients were categorized into four options for laser eye surgery by an XGBoost model [50]. The training data consisted of 142 features per instance, including demographics data, certain measurements of the eyes, and answers to questionnaires. Because it is difficult to compute SHAP values for a multi-class problem, the authors used two types of XGBoost models; a one-vs-rest (option A vs (options B and C)), and a one-vs-one (option A vs option B, and option A vs option C). In this way, they provided *multiple*, and *contrastive explanations* as well. They report that the SHAP values reflect the advantages and disadvantages of the different laser surgery methods well, though no specific metrics were reported. A notable disadvantage was that TreeSHAP does not handle strongly correlated features well, causing those features to having inconsistent importance. This was shown with features measured on both eyes.

**Feature contribution**

Since TreeSHAP is not specifically optimized towards Random Forests (RF), some studies have focused on these models instead [51, 52]. Most explanations for RFs rely on feature contributions which are calculated in two steps: first for each tree, and then aggregated over the forest [51]. The contribution of a feature is calculated from the change in probability of an instance being in a certain class, provided that the split occurs based on that feature. This contribution is then summed over all nodes from root to leaf for each tree, and then averaged over the trees in the forest. The authors note that in contrast with standard RF feature importance, this method can also be used to inspect feature importance for *single instances*, since the algorithm can compare feature importance for that single instance to the average. In this way, they obtained *new information* that was hidden by global feature importance; Some features that were not deemed important in the global classification, came up to be most influential to correctly classify specific instances.

The previous algorithm was integrated into a framework called Forest Floor, which visualizes the data structure of a RF by these feature contributions, quite similar to how PD plots show feature influence [52]. This framework also allows for *interactivity*; for instance, to discover interactions between features (which would not be possible with PD plots), or to plot different lines for a multiclass problem. The author demonstrated the visualizations for various problems showing how to use the platform, though did not perform any evaluations.

**Global surrogate models**

Several other attempts have been made to make ensembles trees more interpretable, but none of these methods have gained considerable attention, or have an actively maintained code base (last contributions over two years ago). These are developed using student models, that mimic the global behavior of the black-box teacher model. Some examples include approximation of ensemble trees into a small number of decision rules by leveraging the EM algorithm [53], and algorithms that extract rules and variable interactions from tree ensembles based on feature contributions [54]. As was noted in the model-agnostic section as well, these types of algorithms seem to get overshadowed by the more popular frameworks.

**Prototypes / Counterfactuals**

A few example-based methods have also been developed specifically for tree ensembles. These highlight *certain instances* in the dataset. For image data, this is a rather popular type of method, since instances can be understood clearly; its many features, pixels, can be understood by the user in a glance. However, for tabular data, an instance might consist of a multitude of features that are not as easily understood by a human and therefore provide less information on what this means for the model. A prototype is an instance that is representative of a certain class in the dataset as determined by the model [55].

One methods selects prototypes based on a tree-specific distance measure, and has found results that seem to show promise for using these prototypes even with tabular data [55]. The authors conducted an experiment in which users were asked to classify cars as fuel-efficient or not. Users were presented with either a ranking of feature influence based on SHAP, or with prototypes. The authors of the paper reported an increase in *user accuracy* when using the prototypes, even on this tabular data. Even though the authors specify that specific, tree-based adaptations of their algorithm outperforms the prototype algorithm as presented by Kim et al. [43], this increase in score is minimal. They also did not compare their methods to that of Kim et al. in the user study. From this study, it seems that prototypes might have some merit for tabular data when there are few features, though the custom algorithms for tree ensembles do not add much additional value.

In the standard R package for random forests, a simple Nearest-Neighbor approach is used to compute one prototype for each class [56], in comparison to the previously discussed algorithm that allows for multiple prototypes per class [55]. Though a single prototype might provide less information, it might still give an indication of what the RF considers to be the most representative sample.

Lastly, a specific implementation was developed which allows the transformation of a true negative instance to the positive class by feature tweaking on ensemble trees [57]. This should remind the reader of *contrasive explanations*, the strategy in which humans like to reason about how a prediction might have been different if certain conditions were changed. In the algorithm, the structure of trees is exploited to find paths back up in the tree where the prediction could become positive again. In this way, a set of feature values is returned for which the prediction could become positive. We also call these explanations *counterfactuals*. The results showed that about 60% of feature tweaks indeed led to more helpful advertisements.

### 3.2.2 Support Vector Machines (SVMs)

In short, SVMs try to find a decision boundary that separates two classes maximally [67]. Using a technique called the *kernel trick*, this boundary can be obtained even for high dimensional feature spaces. As such, the decision boundary is quite abstract and not easily understood by humans as is. The *Support Vectors* (SVs) are those samples of the training data which the SVM uses to define the decision boundary. As such, analyzing these instances could yield an explanation of how classes are separated.

**Decision rules from SVs**

The search for explainable ML techniques for SVMs seems to uncover mainly techniques that date back a bit further that the ten years that were initially set for this survey. Research on explaining SVMs was mainly conducted in the period between 2000-2010 [58, 68]. In general, most research is dedicated to extract decision rules as explanations by analyzing the SVs [58, 59, 68].

A method called SQRex-SVM extracts rules for binary classification [59]. The most discriminative features are selected from the SVs based on their z-score from the mean, and are ranked accordingly. From this set of features and their values based on the SVs, decision rules are formed, that hold for the same feature value region as the SV. Any rules that do not produce true positives are pruned. The results from this paper compare their methods to other techniques to obtain higher comprehensibility which they define as a low number of rules, and comparable or higher accuracy and fidelity on various tabular datasets.

Similarly, ALBA uses SVs to sample more instances around the same local area (defined as the average distance in all feature dimensions), after which decision rules are inferred using different algorithms [58]. The results they obtain show better accuracy and fidelity of the decision rule algorithms when using this sampled dataset rather than the original data.

These strategies seem similar to how the model-agnostic Anchors are formed; starting from a certain instance, sampling more in this local feature space without changing the prediction, and forming decision rules based on that. One might argue this to be the reason why none of these specific SVM-based techniques have actively maintained software at this time, and why little research for SVM-specific explanations is conducted recently.

### 3.2.3 (Deep) Neural Networks ((D)NNs)

A neural network consists of *neurons* which approximate certain functions [7]. These neurons are organized in *layers*; the input layer, hidden layers, and an output layer. The *hidden* layers are the most interesting as the learning algorithm has to decide how to optimally activate the neurons within the layer to produce the desired output. This is done with a process called *backpropagation*, which calculates the gradient of the error with respect to how the neurons in the hidden layers are activated. In this way the NN is optimized. The term *deep* neural networks stems from the number of hidden layers used, or the depth of the NN. Though there exist many different types of (D)NNs, it is most important to understand the basics of how neurons are activated in hidden layers, and how the error of the outcome is backpropagated to optimize these activations. Most of the methods discussed below are available for DNNs of any depth.

**DeepLIFT**

DeepLIFT could be considered one of the main methods for explaining DNNs due to the interest in its research, its implementation, as well as the results it has obtained. Its approach is based on measuring feature importance in differences from a problem-specific *reference state*, which constitutes a default or neutral input [60]. The authors call DeepLIFT a difference-from-reference approach, as opposed to gradient-based techniques, which would fail if gradients are zero. DeepLIFT measures the change in activation of an output neuron of interest, as compared to the activation in that neuron with the reference input. For defining this reference, one must ask themselves the question “what would I like to measure differences against?”.

The authors conducted an experiment on genomics data, where the reference consisted of a sequence of ACGT bases in expected frequencies, and the task at hand was to detect motifs of certain genes. From all methods that were available at that point, DeepLIFT performed the best. Since then, DeepLIFT has been incorporated into Deep SHAP.

**Deep SHAP**

For understanding Deep SHAP, we recommend first reading about Shapley values and the general implementation of SHAP in the model-agnostic section. Although KernelSHAP could also be applied to DNNs, Deep SHAP was developed to specifically optimize the computation of Shapley values for DNNs [32]. This implementation was an adaptation of using Shapley values with DeepLIFT, similar to how Kernel SHAP is equivalent to combining Shapley values with LIME. Deep SHAP uses a distribution of reference samples instead of a single reference input for DeepLIFT, and uses the neuron attributions to approximate Shapley values [32]. Where DeepLIFT linearizes components in a heuristic manner, Deep SHAP achieves the same by computing Shapley values for each component. Computing these component-wise contributions recursively for the entire network, Deep SHAP can approximate Shapley values for the *entire network*.

Deep SHAP was applied to an implementation of a denoising autoencoder on multi-omics data consisting of various biological pathway features [61]. The task at hand was to cluster patients into possible disease subtypes. Deep SHAP was then applied to derive most *influential features*, which showed results that are in line with current knowledge of the various disease subtypes.

**Integrated Gradients**

Integrated Gradients is a technique that is quite similar to DeepLIFT, in the sense that it compares a certain instance to a baseline prediction [62]. However, this approach calculates the accumulated gradient of the prediction from the baseline sample to the current instance, instead of the change in activation in DeepLIFT.

With this technique, the authors conducted an experiment where Integrated Gradients were computed for a network that predicts whether a molecule is active against a certain protein or not. A zeroed vector was used as the baseline input. The obtained feature contributions were in line with the assumption that atom-pairs with a bond between them account for a larger part of the prediction score than the atom-pairs without a bond. This technique is also incorporated into the SHAP framework, though it is slower than Deep SHAP due to the additional computational costs required to obtain high quality integrals [32, 60].

Both DeepLIFT and Integrated Gradients make assumptions and thus break certain rules. DeepLIFT breaks that notion that attributions must be identical for two networks that encode the same function [62]. On the other hand, Integrated Gradients can also produce misleading results due to ignoring dependencies between input variables [60]. To decide on the best type of explanation, one must decide on which factor to concede, and whether computation time is important. The SHAP documentation helps with this type of decision as both options are available.

**Global surrogate models**

Various types of surrogate models have been used to approximate DNNs. DeepRED learns surrogate decision trees for each layer of the DNN [63]. To then create a set of decision rules that holds for the entire DNN, the authors extract rules for each hidden layer, based on the preceding layer, continuing this process from output to input. The authors have further decreased the number of rules by using a pruning method from RxREN. RxREN is an algorithm created for simple NNs, where insignificant input neurons are pruned from a trained network to then obtain decision rules only for important features [69]. Input neurons are deemed insignificant if they influence the accuracy of the classification up to a certain maximum threshold. Even though the DeepRED authors show that their method has high fidelity and thus is *accurate* in comparison to the underlying DNN, their rule set can explode to consist of more than a hundred rules, which violates *simple* rules preferred by humans.

One approach does not leverage mimic models in the traditional sense, but trains a so-called soft binary decision tree [65]. This adapted decision tree learns the same way as a usual mimic model; training on the classifications made by the black-box model. However, instead of highlighting feature values in a hierarchical order as in a traditional decision tree, this approach shows a hierarchy of simplified activations. As the authors focused on visual input, we cannot vouch for its applicability to tabular data.

A method applied to electronic patient records to predict the outcome of an intensive care visit, did show the practical use of a mimic model [64]. However, their mimic model was a gradient boosted tree, which in turn they needed to interpret with PD plots or another student decision tree. The obtained main contributing features were in line with prior expectations and findings, demonstrating that at least some valuable information could be extracted from this rather convoluted approach.

**Similar examples / Counterfactuals**

An interesting technique that was primarily developed to deal with adversarial samples, uses a k-Nearest Neighbors (kNN) classifier to determine an *instance’s closest training samples* in what they call Deep kNN (DkNN) [66]. In this way, one can evaluate if the test sample lies on the underlying training manifold, and can thus be confidently classified or not. These neighbors can be extracted for a certain layer, or for the overall prediction. The notion of “nearest” is determined based on cosine similarity with locality-sensitive hashing. DkNN shows promise to help with adversarial examples, and for inspecting *faulty classifications*, though no experiments were done on tabular data.

Finally, GRACE generates contrastive explanations for neural network predictions in the sense of “Why is this instance classified as X rather than Y” [18]. Note that this does not explicitly include DNNs, as only a single-layer NN was mentioned in the paper. One can think of this explanation as *counterfactual*, where the algorithm searches for a set of feature changes that changes the prediction outcome, while keeping the other features unchanged. A trade-off is made between the number of feature changes, and the influence of the set on the explanation. The top features are derived from a training a transparent model around the decision boundary, similar to how LIME does. Once the top features are defined, it is more straightforward to derive a limited set of features to alter. Robust experiments on eleven tabular datasets were conducted to show the explanations’ fidelity, conciseness, and *user understandability*. The latter experiment involved asking participants to adjust an instance’s feature values to change the prediction, which achieved an average accuracy of 75% for GRACE’s explanations, in comparison to 16% for LIME’s.

## 3.3 Example results

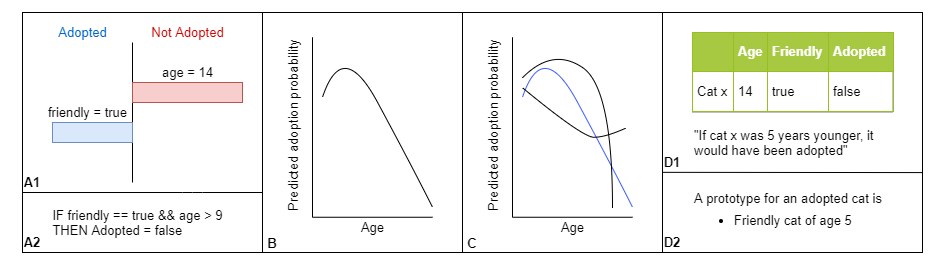
Figure 4 represents some dummy example results that the techniques discussed typically produce. Note that this is in no way an exhaustive figure and that many more types of results exist. Some of the techniques that we name to correspond to the examples have other types of results as well. However, we mean to give the reader an impression of what the different types of explanations look like.

Figure 4: Dummy results from a fictional classification case for predicting whether or not a cat will be adopted from a shelter based on its age and friendliness. Please refer to the descriptions below for more details. A1 shows feature attributions to different prediction results, A2 summarizes the model with a single decision rule, B and C represent PD- and ICE plots respectively, D1 and D2 demonstrate a counterfactual-, and a prototype explanation.

Figure 4A1 contains typical results from methods such as LIME / Anchors, different versions of SHAP and global surrogate models. This explanation shows features of a cat, and how strongly and in which direction their values contribute to either side of the prediction of a certain ML model. Here we thus show a friendly 14-year-old cat. Its age most strongly corresponds to the prediction, that is why it is on top. The fact that it is a friendly cat corresponds to a positive outcome (it being adopted), while the older age makes it likely to not be adopted. Note that from this result we only say something about a single instance.

The same techniques can often also produce the results of Figure 4A2, which is a decision rule. Specific methods for producing decision rules evidently also result in this type of explanation. The decision rule summarizes (some area of) the model in one if/then-statement, though usually, more rules are necessary to approximate the model. From this explanation we know that if a cat is friendly, but older than 9, it will be predicted to not be adopted.

Figure 4B shows a PD- or ALE plot. For the sake of simplicity, we assume it to be a PD plot. In this case, it visualizes the average influence of age on the probability that a cat will be adopted. Kittens are likely to be adopted, though slightly older cats are preferred since they do not require as much work. Once the cats reach a certain age, they are increasingly less likely to be adopted. This holds on average for the entire model. The same PD plot is shown with the blue line in Figure 4C, which demonstrates an ICE plot. Here the other two lines are individual instances where the other features of the model cause the age factor to have a different influence on the prediction. The blue line is the average of the two black lines.

Figure 4D1 is a counterfactual explanation showing how to change a certain instance from a negative prediction to a positive one. It originates from a prediction of a friendly, 14-year-old cat that was predicted to not be adopted. If the cat was five years younger, according to the model it would be predicted to be adopted.

Lastly, in Figure 4D2 we can see a prototype for the adopted class according to the underlying model. Apparently, the most representative sample for the adopted prediction is a friendly, 5-year-old cat. The model has seen a lot of such cats, and has consistently predicted them to be adopted.

# Conclusion

In this survey, we identified techniques for producing explanations of black-box machine learning models on tabular data. As several reviews have been conducted about the definition and taxonomies in the field of XAI, this survey was placed into this consensus terminology. In contrast to the more unanimous definitions such as explanation scope and approach, the explanation type seems to cause some debate. This survey therefore only distinguished between transparency and causation; respectively leveraging transparent mimic models, and determining feature influence on model outcomes. The main contribution of this paper was to identify the recent model-agnostic and model-specific techniques used for explaining ML models, placed into the predefined categories. Table 1 and 2 summarize these findings, while the text provides more details about how such explanations are obtained, their notable features, and any practical examples and evaluations. In this way, a scientist who has optimized their ML model could find an appropriate explanation implementation. To give an impression of the types of explanation results, we provided the reader with some dummy examples, which were frequently encountered while conducting this research.

During the course of constructing this survey we identified some issues and future opportunities. The first of which concerns the fact that few techniques take correlated features into account, so researchers must be wary of feature importance scores and account for correlations appropriately. This is especially pertinent to tabular data, since its features are often correlated. The second major issue was already acknowledged by other research into explainable ML, which is the lack of unified metrics. There seem to be two major approaches taken by the explainable ML community; either a more mathematical perspective by for instance measuring the accuracy to the black-box model, or a more human-centered approach by performing user studies. For any current researchers using explanation techniques on their ML models, we encourage them to evaluate the explanations from both an accuracy-, and user-centered perspective. It seems that for the future, these two types of evaluations must be standardized into unified tests or metrics. Lastly, it seems that a lot of the research into generating explanations, for (D)NNs in particular, is tailored to visual data. It would be a good contribution to experiment with these explanations on tabular data as well, or start new developments with tabular data as a priority.

Finally, we conclude that there is a handful of established methods in the field for generating post-hoc explanations, which are actively being maintained. Mainly model-agnostic approached seem to have the most support, with some optimizations made for specific ML models. Not many new approaches are being researched as most of the works dates from around 2016-2018. The most recent contributions seem to focus on counterfactual explanations, or attempt to steer more into the direction of creating ML topologies that are transparent, or have a transparent component within them.

**Bibliography**

1. Russell, S. and P. Norvig, *Artificial intelligence: a modern approach.* 2002.

2. Lundberg, S.M., et al., *Explainable machine-learning predictions for the prevention of hypoxaemia during surgery.* Nat Biomed Eng, 2018. **2**(10): p. 749-760.

3. Thorsen-Meyer, H.-C., et al., *Dynamic and explainable machine learning prediction of mortality in patients in the intensive care unit: a retrospective study of high-frequency data in electronic patient records.* Lancet Digital Health, 2020. **2**(4): p. E179-E191.

4. Ghosal, S., et al., *An explainable deep machine vision framework for plant stress phenotyping.* Proceedings of the National Academy of Sciences, 2018. **115**(18): p. 4613.

5. Grosenick, L., et al., *Interpretable whole-brain prediction analysis with GraphNet.* NeuroImage, 2013. **72**: p. 304-321.

6. Dornadula, V. and S. Geetha, *Credit Card Fraud Detection using Machine Learning Algorithms.* Procedia Computer Science, 2019. **165**: p. 631-641.

7. Courville, I.G.a.Y.B.a.A., *Deep Learning.* 2016.

8. Kim, J. and J. Canny, *Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention*. 2017. 2961-2969.

9. Bahdanau, D., K. Cho, and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate.* CoRR, 2015. **abs/1409.0473**.

10. Wang, D., X. Wang, and S. Lv, *An Overview of End-to-End Automatic Speech Recognition.* Symmetry, 2019. **11**: p. 1018.

11. Silver, D., et al., *Mastering the game of Go with deep neural networks and tree search.* Nature, 2016. **529**(7587): p. 484-489.

12. Ciregan, D., U. Meier, and J. Schmidhuber. *Multi-column deep neural networks for image classification*. in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012.

13. Barredo Arrieta, A., et al., *Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI.* Information Fusion, 2020. **58**: p. 82-115.

14. Holzinger, A., et al., *Causability and explainability of artificial intelligence in medicine.* WIREs Data Mining and Knowledge Discovery, 2019. **9**(4): p. e1312.

15. Roscher, R., et al., *Explainable Machine Learning for Scientific Insights and Discoveries.* Ieee Access, 2020. **8**: p. 42200-42216.

16. Rudin, C., *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.* Nature Machine Intelligence, 2019. **1**(5): p. 206-215.

17. Benschop, C.C.G., et al., *Automated estimation of the number of contributors in autosomal short tandem repeat profiles using a machine learning approach.* Forensic Science International: Genetics, 2019. **43**: p. 102150.

18. Le, T., S. Wang, and D. Lee, *GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction*, in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining*. 2020, Association for Computing Machinery: Virtual Event, CA, USA. p. 238–248.

19. Chatzimparmpas, A., et al., *A survey of surveys on the use of visualization for interpreting machine learning models.* Information Visualization, 2020. **19**(3): p. 207-233.

20. Lipton, Z.C., *The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.* Queue, 2018. **16**(3).

21. Du, M., N. Liu, and X. Hu, *Techniques for interpretable machine learning.* Communications of the ACM, 2020. **63**(1): p. 68-77.

22. Murdoch, W.J., et al., *Definitions, methods, and applications in interpretable machine learning.* Proceedings of the National Academy of Sciences of the United States of America, 2019. **116**(44): p. 22071-22080.

23. Carvalho, D.V., E.M. Pereira, and J.S. Cardoso, *Machine learning interpretability: A survey on methods and metrics.* Electronics (Switzerland), 2019. **8**(8).

24. Gilpin, L.H., et al. *Explaining Explanations: An Overview of Interpretability of Machine Learning*. in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018.

25. Mittelstadt, B., C. Russell, and S. Wachter, *Explaining Explanations in AI*. 2018.

26. Wang, D., et al. *Designing theory-driven user-centric explainable AI*. in *Conference on Human Factors in Computing Systems - Proceedings*. 2019.

27. Adadi, A. and M. Berrada, *Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI).* IEEE Access, 2018. **6**: p. 52138-52160.

28. Miller, T., *Explanation in artificial intelligence: Insights from the social sciences.* Artificial Intelligence, 2019. **267**: p. 1-38.

29. Holzinger, A., A. Carrington, and H. Müller, *Measuring the Quality of Explanations: The System Causability Scale (SCS): Comparing Human and Machine Explanations.* KI - Kunstliche Intelligenz, 2020. **34**(2): p. 193-198.

30. Ribeiro, M.T., S. Singh, and C. Guestrin. *"Why should i trust you?" Explaining the predictions of any classifier*. in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016.

31. Ribeiro, M.T., S. Singh, and C. Guestrin. *Anchors: High-Precision Model-Agnostic Explanations*. in *AAAI*. 2018.

32. Lundberg, S. and S.-I. Lee, *A Unified Approach to Interpreting Model Predictions*. 2017.

33. Lakkaraju, H., et al., *Interpretable & Explorable Approximations of Black Box Models.* 2017.

34. Krishnan, S. and E. Wu, *PALM: Machine Learning Explanations For Iterative Debugging*. 2017. 1-6.

35. Tan, S., et al. *Distill-and-Compare: Auditing Black-Box Models Using Transparent Model Distillation*. in *AIES 2018 - Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 2018.

36. Bastani, O., C. Kim, and H. Bastani, *Interpretability via Model Extraction.* 2017.

37. Ming, Y., H. Qu, and E. Bertini, *RuleMatrix: Visualizing and Understanding Classifiers with Rules.* Ieee Transactions on Visualization and Computer Graphics, 2019. **25**(1): p. 342-352.

38. Spinner, T., et al., *explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning.* Ieee Transactions on Visualization and Computer Graphics, 2020. **26**(1): p. 1064-1074.

39. Friedman, J., *Greedy Function Approximation: A Gradient Boosting Machine.* The Annals of Statistics, 2000. **29**.

40. Goldstein, A., et al., *Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation.* Journal of Computational and Graphical Statistics, 2015. **24**(1): p. 44-65.

41. Apley, D.W. and J. Zhu, *Visualizing the effects of predictor variables in black box supervised learning models.* Journal of the Royal Statistical Society. Series B: Statistical Methodology, 2020. **82**(4): p. 1059-1086.

42. Fisher, A., C. Rudin, and F. Dominici, *Model Class Reliance: Variable Importance Measures for any Machine Learning Model Class, from the "Rashomon" Perspective.* 2018.

43. Kim, B., O. Koyejo, and R. Khanna. *Examples are not enough, learn to criticize! Criticism for Interpretability*. in *NIPS*. 2016.

44. Koh, P.W. and P. Liang, *Understanding Black-box Predictions via Influence Functions.* ArXiv, 2017. **abs/1703.04730**.

45. Dandl, S., et al. *Multi-Objective Counterfactual Explanations*. in *Parallel Problem Solving from Nature – PPSN XVI*. 2020. Cham: Springer International Publishing.

46. Wexler, J., et al., *The What-If Tool: Interactive Probing of Machine Learning Models.* IEEE Transactions on Visualization and Computer Graphics, 2020. **26**(1): p. 56-65.

47. Slack, D., et al. *Fooling LIME and SHAP: Adversarial attacks on post hoc explanation methods*. in *AIES 2020 - Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020.

48. Lundberg, S. and S.-I. Lee, *Consistent feature attribution for tree ensembles.* 2017.

49. Wachter, S., B. Mittelstadt, and C. Russell, *Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR.* Harvard journal of law & technology, 2018. **31**: p. 841-887.

50. Yoo, T.K., et al., *Explainable Machine Learning Approach as a Tool to Understand Factors Used to Select the Refractive Surgery Technique on the Expert Level.* Transl Vis Sci Technol, 2020. **9**(2): p. 8.

51. Palczewska, A., et al., *Interpreting Random Forest Classification Models Using a Feature Contribution Method*, in *Integration of Reusable Systems*, T. Bouabana-Tebibel and S.H. Rubin, Editors. 2014, Springer International Publishing: Cham. p. 193-218.

52. Welling, S.H., et al., *Forest Floor Visualizations of Random Forests.* ArXiv, 2016. **abs/1605.09196**.

53. Hara, S. and K. Hayashi, *Making Tree Ensembles Interpretable.* 2016.

54. Deng, H., *Interpreting Tree Ensembles with inTrees.* arXiv:1408.5456, 2014.

55. Tan, H.F., G. Hooker, and M. Wells, *Tree Space Prototypes: Another Look at Making Tree Ensembles Interpretable.* ArXiv, 2016. **abs/1611.07115**.

56. Liaw, A. and M. Wiener, *Package 'randomForest': Breiman and Cutler's random forests for classification and regression.* R Development Core Team, 2014. **4**: p. 6-10.

57. Tolomei, G., et al., *Interpretable Predictions of Tree-based Ensembles via Actionable Feature Tweaking.* Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

58. Martens, D., B. Baesens, and T. Van Gestel, *Decompositional Rule Extraction from Support Vector Machines by Active Learning.* Knowledge and Data Engineering, IEEE Transactions on, 2009. **21**: p. 178-191.

59. Barakat, N. and A. Bradley, *Rule Extraction from Support Vector Machines: A Sequential Covering Approach.* Knowledge and Data Engineering, IEEE Transactions on, 2007. **19**: p. 729-741.

60. Shrikumar, A., et al., *Not Just a Black Box: Learning Important Features Through Propagating Activation Differences.* 2016.

61. Lemsara, A., S. Ouadfel, and H. Fröhlich, *PathME: Pathway based multi-modal sparse autoencoders for clustering of patient-level multi-omics data.* BMC Bioinformatics, 2020. **21**(1).

62. Sundararajan, M., A. Taly, and Q. Yan. *Axiomatic Attribution for Deep Networks*. in *ICML*. 2017.

63. Zilke, J., E. Mencía, and F. Janssen, *DeepRED – Rule Extraction from Deep Neural Networks*. 2016. 457-473.

64. Che, Z., et al., *Interpretable Deep Models for ICU Outcome Prediction.* AMIA Annual Symposium Proceedings, 2017. **2016**: p. 371-380.

65. Frosst, N. and G. Hinton, *Distilling a Neural Network Into a Soft Decision Tree.* 2017.

66. Papernot, N. and P. McDaniel, *Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning.* 2018.

67. Guido, A.C.M.a.S., *Introduction to Machine Learning with Python.* 2017.

68. Fortuny, E.J.d. and D. Martens, *Active Learning-Based Pedagogical Rule Extraction.* IEEE Transactions on Neural Networks and Learning Systems, 2015. **26**(11): p. 2664-2677.

69. Augasta, M.G. and T. Kathirvalavakumar, *Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems.* Neural Processing Letters, 2012. **35**(2): p. 131-150.