information

MDPI

Article

Global Translation of Classification Models

Mohammad Al-Merri and Zina Ben Miled *

Electrical and Computer Engineering Department, Indiana University-Purdue University, 723 W. Michigan St., SL 160, Indianapolis, IN 46202, USA; malmerri@iupui.edu
* Correspondence: zmiled@iupui.edu

Abstract: The widespread and growing usage of machine learning models, particularly for critical areas such as law, predicate the need for global interpretability. Models that cannot be audited are vulnerable to biases inherited from the datasets that were used to develop them. Moreover, locally interpretable models are vulnerable to adversarial attacks. To address this issue, the present paper proposes a new methodology that can translate any existing machine learning model into a globally interpretable one. MTRE-PAN is a hybrid SVM-decision tree architecture that leverages the interpretability of linear hyperplanes by creating a set of polygons that delimit the decision boundaries of the target model. Moreover, the present paper introduces two new metrics: certain and boundary model parities. These metrics can be used to accurately evaluate the performance of the interpretable model near the decision boundaries. These metrics are used to compare MTRE-PAN to a previously proposed interpretable architecture called TRE-PAN. As in the case of TRE-PAN, MTRE-PAN aims at providing global interpretability. The comparisons are performed over target models developed using three benchmark datasets: Abalone, Census and Diabetes data. The results show that MTRE-PAN generates interpretable models that have a lower number of leaves and a higher agreement with the target models, especially around the most important regions in the feature space, namely the decision boundaries.

Keywords: explainability; global interpretation; translation; machine learning; TRE-PAN; explainable AI

Citation: Al-Merri, M.; Ben Miled, Z. Global Translation of Classification Models. *Information* **2022**, *13*, 246. https://doi.org/10.3390/info13050246

Academic Editor: Gabriele Gianini

Received: 22 February 2022
Accepted: 9 May 2022
Published: 11 May 2022

1. Introduction

Since 2018, the European Union (EU) has placed regulations on personal data usage and algorithmic decision making systems [1]. As a result, EU citizens are entitled to explanations of algorithmic decisions and are able to contest them [1]. In the United States (US), regulatory bodies have begun investigating the widespread usage of artificial intelligence (AI). In 2014 and 2016, the executive office of the National Science and Technology Committee published two reports related to the ethical usage of AI and its regulatory recommendations [2]. This was followed by the introduction of the National Security Commission Artificial Intelligence Act of 2018 that established a formal committee to review the usage of AI and recommend necessary regulations [3].

Laws that regulate the use of machine learning (ML) applications are difficult to draft since they require extensive technical knowledge to accurately assess the outcomes produced by the underlying algorithms. However, these laws are needed to prevent misuse and decision failures. For instance, recidivism prediction instruments are widely used but are also the subject of controversy because they can inherit biases from the training data [4]. In the US, the judiciary presiding over State of Wisconsin vs. Eric. L. Loomis used an algorithm, COMPASS, to recommend sentencing. It sentenced the accused to 6 years in prison [5]. The defense argued that the usage of a black-box algorithm violated Mr. Loomis's right to due process since the algorithm was a trade secret. On appeal to the Wisconsin supreme court, the judgment was upheld [5]. The court's decision was heavily criticized by law scholars as having "failed to protect due process rights" [6]. These systems

may perpetuate a cycle of incarceration [4]. In order to overcome some of these legal and ethical pitfalls, ML models need to be interpretable and open to auditing [7].

In response to this concern, the IEEE published the "The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems", a set of guiding principles for ethical AI usage [8]. These principles formed the foundation of the IEEE P7000 series of standards addressing AI standardization. Subsequently, the P7001 and P7003 standards required transparency and algorithmic bias considerations for autonomous systems, highlighting the need for interpretability for all ML models.

In general, ML models can be classified into two main categories in terms of interpretability. The first category consists of easily interpretable models, such as Bayesian networks [9], decision trees [10] and random forests [11]. The second category includes more complex models such as neural networks [12] and support vector machines [13]. This second category of models is often more accurate and generalizes better to new data [14]. However, it suffers from reduced interpretability [15]. In fact, the more complex the model, the less interpretable it becomes. More examples of this category include deep neural networks [12], convolution neural networks [16] and recurrent networks [17]. Similarly, the interpretability of SVM decreases with higher-order SVMs, which rely on RBF or polynomial kernels as opposed to the simpler linear kernels.

In [15], Lipton divides the notion of interpretability into two main categories: transparency and post hoc explanation. Transparency aims to deliver model- or global-level interpretability, whereas post hoc explanation is a per input "after the fact" explanation that provides a local level of interpretability. Both the local and the global interpretability of ML models have been investigated in previous studies. These studies propose a translation mechanism, where a non-interpretable model is translated to an interpretable model. For instance, the local interpretable model-agnostic explanations (LIME) technique translates a non-interpretable model to a locally interpretable one by sampling data around a query from the non-interpretable model [18]. The sampled data are labeled using the non-interpretable model and then are used to train a simple linear separator. The weights of the linear separator are provided as the explanation. An example of a global interpretation technique is TRE-PAN [19], which translates a neural network by training a decision tree model using data generated from the original model. These two approaches treat the target model as a black-box where only the outcome produced for a given input is available. A different type of global translation techniques relies on the complete knowledge of the architecture and parameters of the target model. For instance, the internal structure of a neural network was used to generate rules from an induced decision tree in CRED [20].

Global interpretation is the focus of this paper. A decision tree that relies on linear hyperplanes as separators is used to provide global interpretability for a target neural network model. The target model is considered to be a black-box. The present paper also introduces two metrics that can more accurately compare the target model and the interpretable model, specifically around the decision boundaries. Agreement between the target model and the interpretable model near the decision boundaries is critical since these regions delineate between different outcomes of the target model.

2. Related Work

Several methods have been proposed in the literature for translating non-interpretable ML models to interpretable ML models [18,19]. As mentioned above, these methods fall into two categories: transparency and post hoc explanation or, in other words, global translation and local translation, respectively [15]. Global translation corresponds to transparency because it aims to provide a comprehensive understanding of the behavior of the target model. Local translation corresponds to post hoc explanation, as it focuses on a subspace of the entire model. Lipton [15] further divides these two categories, where simulatability, decomposability and algorithmic transparency are sub-categories of global translation, whereas text explanations, visualization, local explanations, and explanation by example are sub-categories of local translation.

2.1. Global Translation

The aim of algorithmic transparency is to create an interpretable model that estimates the target model by translating it into a model whose behavior is understood [15]. For instance, TRE-PAN generates a decision tree which describes the behavior of a deep neural network [16,17]. This approach treats the target model as a black-box and is not limited to neural networks. It uses the target model to generate data that are used to train the interpretable decision tree. For each child node in the tree, enough data are generated to obtain the best split for the constrained sample space inherited from the parent node. TRE-PAN uses two of three decision trees. The top three features are selected, and the corresponding thresholds are established based on the potential gain in entropy from the split for each node. At most, two conditions need to be satisfied for a sample to be assigned to the left subtree.

The motivation behind TRE-PAN is that decision trees require substantially more training data than neural networks in order to achieve the same accuracy. These data may not be available. Therefore, the non-interpretable model, once trained with the available data, can be used to generate the additional synthetic data needed to train the interpretable model [19]. Some limitations of TRE-PAN include the fact that two out of three trees are more difficult to interpret than binary splits since three different possibilities are evaluated at each node. Moreover, the depth of the tree in TRE-PAN is primarily dictated by the complexity of the non-linear decision boundaries of the target model under consideration [19]. As TRE-PAN generates data near the decision boundaries, the information gain from splitting is likely to be greater than the gains from splitting regions that are farther away from the decision boundaries. This is anticipated because the data will have a more balanced proportion of positive and negative samples near the decision boundaries, requiring a higher number of splits to represent them. In fact, when the decision boundary of the target model has a non-linear shape, representing the area constrained by this shape requires several rectangles of varying sizes. Therefore, these boundaries often correspond to a large number of leaves in the TRE-PAN decision tree. Limiting the depth of the tree comes at the cost of lower accuracy [19].

An alternative global translation approach only applicable to neural networks was proposed in [20]. This approach requires access to the hidden nodes of the target neural network. Rules describing the global behavior of the network are extracted using the "continuous/discrete rule extractor via decision tree induction" (CRED) algorithm [20]. This algorithm builds a decision tree by clustering data around the training samples that activate a hidden node for a specific output class. CRED builds decision trees for each layer, generates intermediate rules, and combines them into global rules.

Another rule extraction technique was proposed in [21]. This technique is labeled "rule extraction by reverse engineering the neural networks" (RxREN). It extracts the rules in two phases. RxREN requires access to the internal architecture of the target neural network as well as the training data. In the first phase, RxREN prunes the input nodes based on their significance. In the second phase, the misclassified training examples are used to infer the feature ranges of the remaining (i.e., significant) input nodes, and to create threshold rules [21]. A subsequent improvement to RxREN, named "deep neural network rule extraction via decision tree induction" (DeepRED) was proposed in [22]. DeepRED combines RxREN and CRED. It first uses RxREN to prune the input nodes, and then uses a modified version of CRED to develop the explainable decision tree [22]. The modification consists of extracting intermediate rules from the target neural network model before merging them into complete decision trees [22].

2.2. Local Translation

The objective of local translation is to observe a limited subset of the feature space and attempt to explain it using specific input examples. Therefore, the focus of local translation is on explaining individual decisions, rather than the behavior of the entire model. An example of this type of translation is the model-agnostic explanations (LIME) [18].

This technique relies on a post hoc approach to explain local classification results. Specifically, LIME uses a linear model to represent a decision derived from a non-interpretable model. Given a target model and an input vector with a corresponding class prediction, the input is perturbed to generate synthetic data in the local neighborhood of the input vector under consideration. This synthetic data are then weighted by using a distance metric from the original input and used to train the interpretable linear model. By observing the feature weights of this linear model, the features that dictated the classification can be identified.

There are two potential limitations to LIME: random explanations and unconvincing explanations. LIME uses randomly perturbed data to create a local linear model. Therefore, it can generate different explanations for the same input depending on the distribution of the sampled data. Moreover, the local explanation can become less reliable if the input vector is near a non-linear decision boundary of the target model [18].

Other local translation techniques include utilizing a heat map to visualize the activation patterns in the input data [15,23,24]. For instance, given an input image, the pixels which were the most influential in selecting a predicted class can be identified [25]. In particular, it is possible to decompose the output of an image classifier such that each pixel in an image is weighted on how much it contributed to deciding the final outcome as in [26]. This decomposition is applied on a layer-by-layer basis, where the weights are propagated from one layer to the next layer. The decomposition can be done on pre-trained models, but does require access to the internal parameters of the models.

A similar decomposition technique can be applied to neural networks using DeepLIFT [24]. In this case, the contribution of the input feature to each layer is decomposed into a summation of weights as the value of the feature is propagated from the input layer to the output layer [24]. This technique requires a significant amount of domain knowledge, as a "reference" input (i.e., near the decision boundary) needs to be used as the baseline [24].

In [27], locally interpretable models such as LIME and DeepLIFT are considered to belong to the same "class of additive feature attribution methods". These methods decompose the target model into a sum of weights of the input features. A technique for finding these weights according to some properties (e.g., local accuracy, missingness, and consistency) is needed. Towards this goal, the "Shapley additive explanation" (SHAP) is proposed as a unified measure of feature importance [27]. The output of a given model is compared to the output of the same model after the removal of some features. The difference is assumed to be proportional to the contribution of the missing features [27].

Since SHAP is a local explanation technique, it is unable to capture the global behavior of the model. SHAP explains examples in a post hoc fashion and describes the aggregate behavior of a target model by only considering a representative sample of the data. Therefore, it is unable to describe the decision boundaries of the target model and is vulnerable to adversarial attacks [28].

In general, while local translation is easier to implement than global translation, it is prone to adversarial manipulation in various applications, including image classification and insurance decision support systems [15]. For example, an adversarial fake image can be overlaid on top of a real image, causing the model to misclassify the image [29]. A globally interpretable model is more resilient to such adversarial attacks and provides an opportunity to audit how a given decision is reached. The latter is important, as it can identify gaps in the inference mechanisms used by the target model.

2.3. Evaluation Metrics

A survey of previous research on model translation indicates that most of the techniques use common metrics to describe how accurately the interpretable model represents the target model. TRE-PAN calls this metric fidelity [19]; CRED refers to it as accuracy [20]; and RxREN calls it rule accuracy [21]. This metric is called model parity in the present paper. It compares the outcome of the target model and the interpretable model on a test dataset. Model parity is unable to represent the behavior of the target model near the decision boundaries. Data near the decision boundaries are significantly more sparse, compared to

the rest of the feature space. When validation data are sampled uniformly from the entire feature space, they often fail to focus on the boundaries. As a result, interpretable models may generate high parity values but fail near the decision boundaries. The parity metric needs to consider the entropy of the subspace when evaluating the match between the target model and the interpretable model. Ideally, if the entropy is too low, all data should be considered mislabeled.

3. Materials and Methods

The global translation technique proposed in this paper translates a target ML model into a hybrid model consisting of a decision tree with linear SVM classifiers at each node. This hybrid ML architecture, which was previously proposed in [30,31], is extended to global translation. The proposed technique, MTRE-PAN, leverages the interpretability of the decision tree and the generalizability of SVM. It uses SVM with linear kernels instead of higher order kernels to facilitate interpretability. As in TRE-PAN, MTRE-PAN treats the target model as a black-box.

3.1. Model Overview

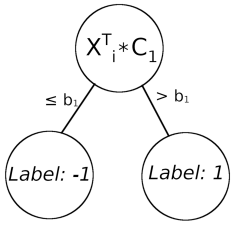
Let $f(\mathbf{x}), \mathbf{x} \in \mathbb{R}^N$ represent a pre-trained, non-interpretable model, where $\mathbf{x}$ is the feature vector consisting of N dimensions and $f(\mathbf{x})$ is a binary classifier with codomain $\{-1, 1\}$. MTRE-PAN builds an interpretable model for f consisting of a decision tree that uses hyperplanes to split each node into subtrees.

Each node k in MTRE-PAN is associated with a weight matrix $\mathbf{C_k} \in \mathbb{R}^{N*M}$ and a bias vector $\mathbf{b_k} \in \mathbb{R}^N$. $\mathbf{C_k}$ and $\mathbf{b_k}$ form the set of linear constraints for each node. When applied simultaneously, they combine to form a *convex hyper-polygon*, where each node represents a mutually exclusive partition of the space of f. In the first phase, MTRE-PAN is trained using the original training data used to develop f, along with additional training data sampled from f. These data consist of the input set $Q = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots \mathbf{x}_m\}$ and the corresponding label set $Y = \{l_1, l_2, l_3, \ldots l_m\}$. Let $Parent(k)$ represent the parent node of node k. The set of samples that are passed from a parent node to its left and right children is defined below:

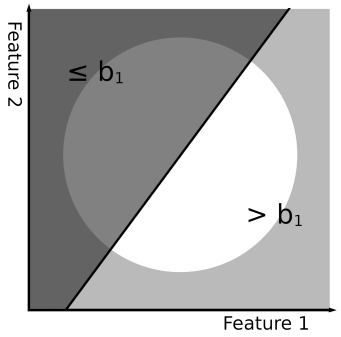
$$\text{left child: } Q_l = \{\forall \mathbf{x} \in Q_{Parent(l)} \mid \mathbf{x}_i^T \mathbf{C}_{Parent(l)} \leq b_{Parent(l)}\} \quad (1)$$

$$\text{right child: } Q_r = \{\forall \mathbf{x} \in Q_{Parent(r)} \mid \mathbf{x}_i^T \mathbf{C}_{Parent(r)} > b_{Parent(r)}\} \quad (2)$$

Leaf nodes inherit a label from the SVM classifier according to the side of the split they fall into. Figures 1 and 2 show two instances of an MTRE-PAN tree for the same function f, one at depth 1 and the second at depth 2. The target model f is a circle, where samples inside the circle are labeled "1" and those outside the circle are labeled "-1". The feature space for this synthetic example consists of two dimensions: the horizontal (feature 1) and vertical (feature 2) positions of the sample. In Figure 2, the right child of the root node is expanded by training an SVM on data sampled from a subspace of the feature space of $f(x)$, such that any data used in the training must satisfy $\mathbf{x}_i^T \mathbf{C}_1 > b_1$. Similarly, the left child of the root node is expanded by training an SVM on data sampled from a subspace of the feature space of $f(x)$, such that any data used in the training must satisfy $\mathbf{x}_i^T \mathbf{C}_1 \leq b_1$. As the nodes continue to be split, the MTRE-PAN estimate of f improves. The splits induced by the new leaf nodes in Figure 2a start to converge to the boundaries of f (Figure 2b).

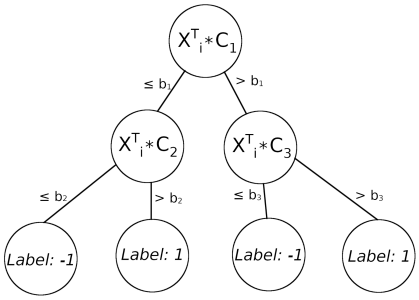


(a) MTRE-PAN at depth 1

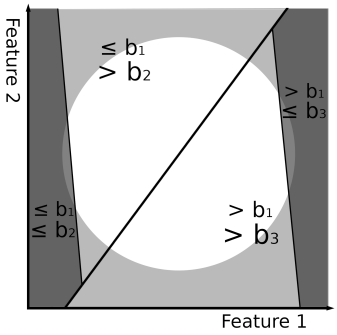


(b) MTRE-PAN partitioning the feature space of a target model

Figure 1. (**a**) represents a trained MTRE−PAN model of depth 1, with the hyperplane(s) acting as separators. It shows how an input vector X_i is classified to the correct leaf using the constraints C_1 and the bias b_1 as per Equations (1) and (2). (**b**) illustrates how MTRE−PAN partitions the feature space of an existing target model. For this example, the decision boundary is a circle. The light regions are assigned label 1, and the dark regions are assigned label −1.



(a) MTRE-PAN at depth 2



(b) Sub-partitioning of the feature space

Figure 2. MTRE−PAN is applied to the same target model in Figure 1 with the depth expanded to 2.

In MTRE-PAN, the decision to split a node is made according to the standard binary gain measure G [32] given by

$$G(Q_k) = -E(\frac{p}{p+n}) \tag{3}$$

where Q_k are the data of node k, p is the number of positive examples of Q_k, n is the number of negative examples of Q_k, and the entropy E is defined as follows:

$$E(\alpha) = -(\alpha \log_2 \alpha + (1-\alpha)\log_2(1-\alpha)) \tag{4}$$

Nodes whose gains fall below a preset threshold are considered uncertain and therefore are candidates for further splitting. The main steps of MTRE-PAN are outlined in Algorithms 1 and 2. Each node generated by MTRE-PAN may sample the original non-interpretable model for more training data to add to its dataset Q if the available data are not sufficient to represent the subspace. This is accomplished by calling *sampling-on-demand*, as discussed next.

Algorithm 1: M-TRE-PAN algorithm.

Begin *Initializing Variables*
- Load training data of the target model (Tf);
- Enter entropy threshold (Et);
- Enter variance cutoff (Vc);
- Initialize an empty priority queue (Pq) ordered on gain as in Equation (3);

Begin *Building the MTRE-PAN tree*
- Create the Root node of the tree (kRoot);
- Populate kRoot with Tf;
- Call *Sampling_on_Demand* (Algorithm 2) on kRoot;
- Place kRoot in the queue Pq;
- **while** *The queue* Pq *is not empty* **do**
  - Dequeue a node (k) from Pq;
  - **if** *The entropy of* k *is above* Et **then**
    - Train a linear classifier (Sep_k) on the data in k;
    - Add the parameters of Sep_k to the constraints of k ($\mathbf{C}_k$ & b_k) from Equations (1) and (2) ;
    - Create two new empty nodes for the left (childL) and right (childR) children of k;
    - Use $\mathbf{C}_k$ & b_k to split the data already sampled and stored in k;
    - As in Equations (1) and (2), data are partitioned (Q_l & Q_r) between childL and childR;
    - Call *Sampling_on_Demand* for both childL and childR;
    - Attach childL and childR as the children of k;
    - Place childL and childR in the queue Pq;
- Return the Root node kRoot;

Algorithm 2: Sampling on demand.

Begin *Initializing Variables*
- Load input node k;
- Load trained target model as a function $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^N$;
- Load user defined entropy threshold Et;
- Load user defined variance cutoff Vc;
- Load the constraints ($\mathbf{C}_k$ & b_k) of node k;
- Load the data (Q_k) stored in k;

Begin *Sample the subspace of the node*
- Calculate the point estimate (Pe) of the entropy of Q_k stored in k using Equation (6) ;
- Calculate the variance (Pv) of the point estimate from Equation (7);
- Build a bounding box (Box) as a set of constraints for every feature of Q_k. This bounding box constrains the maximum and minimum of each feature thereby surrounding all the data in Q_k;
- **while** Pv *is greater than* Vc **do**
  - Sample data uniformly along the dimensions of Box;
  - Discard samples that do not satisfy the constraints $\mathbf{C}_k$ & b_k of the node k. If k is the left child of its parent, Equation (1) is used to accomplish this process, otherwise Equation (2) is used;
  - Add the data to Q_k, the list of data of node k;
  - Recalculate Pe and Pv on Q_k;

3.2. Sampling on Demand

After a leaf node is created in Algorithm 1, it might be necessary to generate more data to measure the entropy of the subspace encapsulated by its constraints. The leaf node was created by imposing additional constraints on the region of the feature space available to the parent node. The decision to sample more data is made by comparing the variance of the data assigned to the leaf to a predefined variance cutoff as shown in Algorithm 2.

The constraints associated with the leaf node (i.e., Equations (1) and (2)) make the subspace available for sampling a hyper-polygon. Data were already generated by the parent and distributed to the left and right child nodes in accordance to their respective constraints. A bounding box is built around these data. The bounding box is simply a hyper-rectangle that surrounds the data by placing bounds on the maximum and minimum of each feature plus a slight margin. The bounding box is uniformly sampled for data that is labeled using f and if the sample fits within the constraints of the leaf node, it is added to its dataset.

The entropy is measured using a point estimate of the probability. This is because in Equation (4), the value of α (the probability) is unknown. Every sampled data point can be considered to come from a series of independent and identically distributed *indicator random variables:* $I = \{I_1, I_2, I_3, \ldots I_j\}$, and since $ExpectedVal[I] = \alpha$, we can estimate α with the sample mean $\overline{\alpha}$:

$$\alpha \approx \overline{\alpha} = \frac{1}{N}\sum_{0}^{N} x_j \tag{5}$$

and the entropy as

$$E(\alpha) \approx E(\frac{1}{N}\sum_{0}^{N} x_j) \tag{6}$$

In order to determine when enough samples have been collected, the variance of $\overline{\alpha}$ is observed, and since the variance of an indicator R.V is $var[I] = \alpha * (1 - \alpha)$

$$var[\overline{\alpha}] = \frac{\overline{\alpha} * (1 - \overline{\alpha})}{N} \tag{7}$$

While sampling the data, if the variance of the point estimate falls below a user-defined variance cutoff, sampling is stopped, and the entropy estimate of Equation (6) is considered to be an accurate estimate.

4. Results

Multiple experiments were conducted to assess the efficacy of MTRE-PAN and compare it to that of TRE-PAN [16,17]. In its proposed implementation, TRE-PAN generates an interpretable decision tree for each target model using a two out of three split as described in Section 2.1. In order to simplify the comparison with MTRE-PAN with respect to the depth of the resulting interpretable models, the C4.5 binary implementation of TRE-PAN was used [33]. MTRE-PAN, the model proposed in the present paper, consists of a hybrid combination of a binary decision tree and a linear SVM classifier at each node of the tree. Both TRE-PAN and MTRE-PAN were used to generate interpretable models with varying tree depths for several target models.

The first target model is a simple function with a circular boundary delineating the negative and positive samples. This model was used earlier to illustrate the methodology. The remaining target models are feed forward neural networks, which were trained using three public domain datasets.

The hyperparameters of MTRE-PAN and TRE-PAN include the maximum depth, the cutoff entropy, the cutoff variance and the margin as described in Table 1. All the ML target models follow the same general architecture that consists of the following:

- An input layer and two hidden layers, each with a number of nodes equal to the number of input variables in the dataset.
- An output layer consisting of a single node.
- All the nodes use a sigmoid activation function.

Table 1. Hyperparameter definitions and values used in the study.

Hyperparameter	Description	Value
Maximum Depth	The maximum allowable depth of the interpretable decision tree.	10
Cutoff Entropy	A leaf node with an entropy higher than the Cutoff Entropy is considered *uncertain* and is a candidate for further splitting.	0.0808
Cutoff Variance	Places an upper limit on the number of data points sampled in a given leaf. A lower cutoff variance will result in a more accurate value for the sample entropy.	10^{-5}
Margin	The width around the decision boundary of the target ML from which data are being sampled. It is based on the input of the last layer of the target ML model. The margin is not used during the training of the interpretable model. It is simply used to calculate the post hoc metric, boundary model parity, defined below in order to test the efficacy of the algorithm near the decision boundaries of the target model.	0.05

This architecture is trained with 70% of the original data over 100 epochs. The remaining 30% are held-out samples that are used for validation and testing. The data are normalized to [−1,1]. After training, the model that achieved the highest accuracy on the validation data across all the epochs is retained.

Four metrics are used to compare MTRE-PAN and TRE-PAN in this study:

- Model parity: The agreement in classification between the decision tree and the target model f. It is measured as the ratio of matching labels between f and either the decision tree generated by MTRE-PAN or TRE-PAN over the total number of samples in the validation set. Model parity is calculated as

$$\frac{TP + TN}{TP + TN + FP + FN} * 100. \tag{8}$$

- Certain model parity: This metric is similar to the model parity, except, in this case, the sample in the validation dataset that is assigned to uncertain nodes (i.e., nodes with entropy below the cutoff entropy) are labeled uncertain. These samples cannot match any label from f and as such are considered misses. This metric is calculated as

$$\frac{TP + TN}{TP + TN + FP + FN + uncertain} * 100 \tag{9}$$

  and takes into consideration uncertain nodes that require further expansion.
- Boundary model parity: This metric is also similar to the certain model parity. However, the validation dataset is limited to the samples that are near the decision boundary within a predefined margin (Table 1). The boundary model parity measures the progress of the interpretable model toward replicating the behavior of the target model near the decision boundaries. It identifies an interpretable model that may have a high certain model parity but may not fare well around the decision boundaries.
- Leaf count: The number of leaves in the interpretable decision tree generated by either MTRE-PAN or TRE-PAN.

4.1. Synthetic Data

MTRE-PAN makes use of linear separators at each node of the tree. This is similar to LIME [18]. However unlike LIME, MTRE-PAN uses multiple separators whose constraints define a hyper-polygon at each leaf node. As the tree grows, the set of polygons from the root to a leaf node decrease in entropy after every split. This corresponds to a decrease in the total area of the uncertain polygons. Therefore, the certain polygons start to approach the boundaries of the target model.

In order to illustrate this aspect, MTRE-PAN was applied to a simple function f consisting of a circle with a radius of 1 introduced earlier. The samples that fall inside the circle are positive and those outside are negative. TRE-PAN was also applied to the same synthetic function. Results for both MTRE-PAN and TRE-PAN are provided in Tables 2 and 3, respectively.

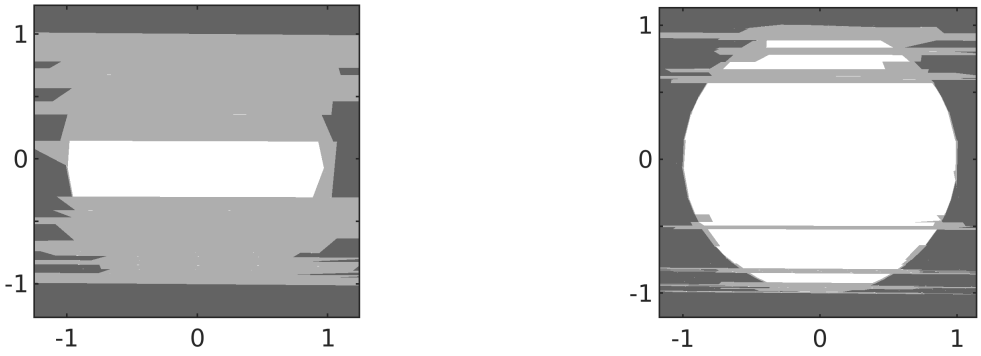
Table 2. Evaluation of the interpretable models for the circle function generated by MTRE-PAN at depths ranging from 1 to 12.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	91.5	0	0	1
2	42.88	34.38	0	2
3	78.65	34.38	0	3
4	81.75	66.53	0	5
5	85.22	66.53	0	8
6	84.58	66.53	0	14
7	87.7	69.85	0	26
8	88.88	76.45	1.47	48
9	94.03	81.85	25.29	80
10	95.95	90.08	48.82	132
11	97.78	94.4	69.12	206
12	98.9	96.75	82.35	305

Table 3. Evaluation of the interpretable models for the circle function generated by TRE-PAN at depths ranging from 1 to 12.

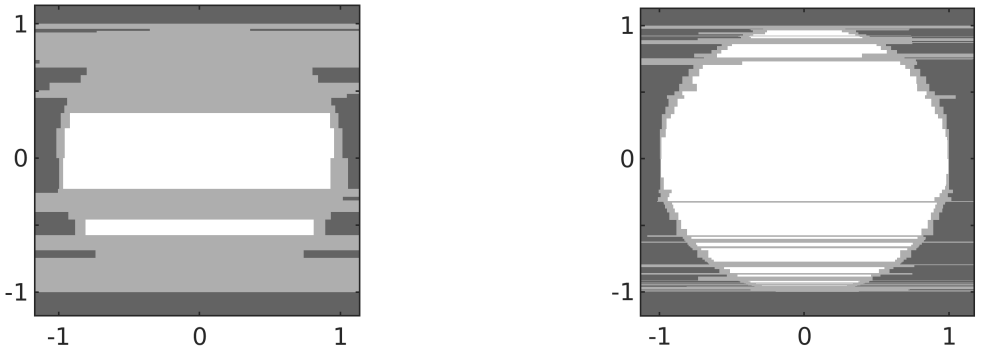
Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	90.95	0	0	1
2	50.5	0	0	2
3	77.72	0	0	4
4	81.33	64.85	0	8
5	83.45	64.85	0	14
6	83.58	66.2	0	26
7	87.15	70.5	0	49
8	88.7	76.12	12.15	91
9	92.85	83.28	39.23	166
10	95.95	89.62	58.01	286
11	98.12	94.75	75.97	476
12	99.15	96.85	86.19	775

Figure 3 is a visualization of the polygons generated by MTRE-PAN for f at depths 9 and 12. It illustrates the convergence of the polygons to f. That is, the collective area of the uncertain polygons becomes smaller at depth 12 compared to depth 9. Moreover, as expected, the figure shows that the uncertain polygons always contain the decision boundaries of f. Otherwise, the polygon will not include both positive and negative labels and would have an entropy of 0. This behavior is also seen in Figure 4 for TRE-PAN. The uncertain polygons (i.e., hyper-rectangles in this case) also contain the decision boundaries of f.



(a) Depth = 9 **(b)** Depth = 12

Figure 3. The polygons generated by MTRE−PAN for the circle function f with a radius of 1 when the maximum tree depth is set to (**a**) 9 and (**b**) 12. The dark and white-shaded polygons define the boundaries of the negative and positive samples, respectively. The medium shaded polygons represent uncertain regions.



(a) Depth = 9 **(b)** Depth = 12

Figure 4. The polygons generated by TRE−PAN for the circle function f with a radius of 1 when the maximum tree depth is set to (**a**) 9 and (**b**) 12. The dark and white-shaded polygons define the boundaries of the negative and positive samples, respectively. The medium shaded polygons represent uncertain regions.

Since the uncertain polygons are mutually exclusive, they can be used as an estimate of the decision boundaries and the overall behavior of the underlying model. As mentioned above, the accuracy of the interpretable model in representing the decision boundaries depends on the values of the cutoff variance and cutoff entropy. If the cutoff variance is high, it may not be possible to generate enough data to accurately label a polygon as positive, negative, or uncertain. On the other hand if it is low, more data are needed to ensure that the sample variance of the entropy is below the cutoff variance. Similarly, if the cutoff entropy is high, it may not be possible to decide whether a leaf node is certain or uncertain. This may potentially lead to labeling polygons that contain a decision boundary as certain. A cutoff entropy close or equal to zero with a sufficiently low cutoff variance will ensure that no decision boundary is missed.

If a decision boundary falls within a certain polygon (i.e., a polygon with an entropy lower than the cutoff entropy), it is still possible to estimate the missing decision boundary. This entails finding neighboring leaves that do not have the same label since an estimated boundary is simply a shared constraint that separates neighboring polygons of different labels.

MTRE-PAN provides a global explanation of the function f in the form of a set of uncertain polygons and of polygons that have an estimated boundary as a constraint. The remaining polygons provide additional constraints that help define the limits of this estimate in the feature space. This characteristic is important, as it avoids the unbounded plane issue observed in LIME. In LIME, a plane is generated as a local estimator of the decision boundary. However, the plane is not delimited in the feature space.

Tables 2 and 3 show that the parity of both interpretable models of f are high for depths greater than 3. However, MTRE-PAN produces interpretable models with a lower number of leaves and similar certain model parity and boundary model parity to those produced by TRE-PAN. At depth 12, the interpretable model generated by MTRE-PAN consists of 305 leaves, whereas the one generated by TRE-PAN includes 775 leaves.

4.2. Abalone Data

The Abalone dataset consists of recorded physical characteristics for the Abalone mollusks [34]. It includes 4177 samples, where each sample has 8 features and an integer label representing the physical characteristics of abalone gastropods. The input features are sex, length, diameter, height, whole weight, shucked weight, viscera weight, and shell weight. The label, rings, is an integer number that represents the age of abalone mollusks. For the purpose of this study, it was converted to -1 for all values below the median and 1 for all values above the median in order to enable binary classification. The target model for this dataset achieved an 84.4% accuracy over the validation dataset. The performance metrics of the corresponding interpretable models generated by MTRE-PAN and TRE-PAN are reported in Tables 4 and 5, respectively.

Table 4. Evaluation of the interpretable models for the Abalone dataset generated by MTRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	35.6	0	0	1
2	93.72	0	0	2
3	81.27	51.81	0	4
4	83.39	67.65	45.87	7
5	86.68	67.65	45.87	12
6	88.27	74.25	64.99	22
7	92.12	74.25	64.99	40
8	92.77	76.95	68.74	76
9	93.93	83.17	77.74	144
10	95.04	87.02	83.86	259

Table 5. Evaluation of the interpretable models for the Abalone dataset generated by TRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	36.14	0	0	1
2	74.71	0	0	2
3	69.25	0	0	4
4	69.2	0	0	8
5	70.91	15.67	0	16
6	74.93	30.71	16.4	31
7	79.6	38.41	21.6	58
8	80.61	49.87	37.16	109
9	84.49	55.34	42.98	201
10	85.46	61.73	50.11	374

From the results, both MTRE-PAN and TRE-PAN approach the target model in terms of model parity. MTRE-PAN begins to achieve a non-zero certain model parity earlier in

comparison to TRE-PAN (depth 3 vs. depth 5). MTRE-PAN also starts from a higher certain model parity when compared to TRE-PAN (51.81% vs. 15%). Both models converge near the boundaries, with MTRE-PAN achieving a non-zero boundary model parity sooner (depth 4 vs. depth 6), with a significantly higher starting parity (45.87% vs. 16.40%). As the leaf count indicates, MTRE-PAN at depth 4 has a significantly lower number of leaf nodes when compared to the closest TRE-PAN tree (with respect to parity) at depth 9 (i.e., 7 leaf nodes vs. 201 leaf nodes). The cost of training the linear classifiers of MTRE-PAN is superseded by the exponentially higher number of splits that is needed for TRE-PAN to achieve a similar parity. It can also be argued that while the separators used by TRE-PAN are simple, the high number of nodes complicates the interpretability of the resulting decision tree.

4.3. US Adult Census Data

This dataset is a collection from the US 1994 adult census data [35]. It includes 13 input variables and one output variable for 32,561 individuals that responded to the census. The input variables are age, work class, level of education, education years, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, and native country. The output variable is the income of the individual. In the original dataset, the income is a binary label that is set to -1 if the income is less than USD 50,000 and 1 otherwise. The target neural network model for the US Adult Census Data achieved an accuracy of 82.9%. The performance metrics of the corresponding interpretable models generated by MTRE-PAN and TRE-PAN for this dataset are included in Tables 6 and 7, respectively.

Table 6. Evaluation of the interpretable models for the US Adult Census dataset generated by MTRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	5.13	0	0	1
2	95.76	0	0	2
3	90.53	84.25	0	4
4	95.37	84.25	0	7
5	95.24	88.64	0	13
6	95.86	89.21	12.5	24
7	95.64	89.68	12.5	45
8	96.02	89.68	12.5	86
9	96.11	90.32	15.57	168
10	96.4	90.73	21.05	327

Table 7. Evaluation of the interpretable models for the US Adult Census dataset generated by TRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	5.3	0	0	1
2	92.21	0	0	2
3	80.76	72.92	0	4
4	90.94	72.92	0	7
5	91.21	72.92	0	13
6	90.16	78.64	0	25
7	92.35	80.08	0	47
8	92.56	82.1	0	90
9	93.07	84.25	1.9	173
10	93.65	84.84	1.9	330

Similar to the Abalone dataset, these results reflect an overall higher certain model parity with MTRE-PAN compared to TRE-PAN. However, both MTRE-PAN and TRE-PAN struggle when attempting to converge to the decision boundary within the margin.

They need depths of 6 and 9, respectively, before they start to show convergence toward the decision boundary. While the certain model parity increases steadily for both TRE-PAN and MTRE-PAN, the boundary model parity stagnates at depth 9 for TRE-PAN. The decision boundaries for the target model associated with the US Adult Census dataset are more complex than those of the Abalone dataset. This complexity is compounded by the increased number of dimensions in this dataset compared to the Abalone dataset.

4.4. Diabetes Diagnosis Data

This dataset covers a population of 798 Pima Indian women. It consists of eight input features, which were selected based on the WHO suggested predictors for diabetes mellitus [36]. The label is either 1 or −1 based on whether or not diabetes is detected for each individual. The eight input features are age, number of pregnancies, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-h serum insulin, body mass index, and diabetes pedigree function. The target model for this dataset achieved an accuracy of 70.8%, which is lower than the previous two target models. The performance metrics of the corresponding interpretable models generated by MTRE-PAN and TRE-PAN are included in Tables 8 and 9, respectively.

Table 8. Evaluation of the interpretable models for the Diabetes dataset generated by MTRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	16.36	0	0	1
2	87.41	0	0	2
3	74.61	53.57	0	4
4	81.52	53.57	0	7
5	79.88	53.57	0	13
6	83.21	64.99	14.37	25
7	89.49	70.22	24.89	46
8	91.64	74.98	33.88	85
9	93.43	79.95	45.17	154
10	94.17	85.78	58	275

Table 9. Evaluation of the interpretable models for the Diabetes dataset generated by TRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model Parity	Boundary Model Parity	Leaf Count
1	16.72	0	0	1
2	70.27	0	0	2
3	74.45	43.13	0	4
4	82.79	43.13	0	7
5	80.49	43.13	0	13
6	81.48	52.33	0	25
7	84.4	60	13.18	47
8	86.87	64.02	13.18	87
9	87.48	68.92	20.91	163
10	88.98	70.77	23.43	303

Unlike the Census data, the boundary model parity converges faster. However the starting value of the non-zero model parity is still significantly lower compared to the Abalone dataset. Considering that the Diabetes dataset has the same number of dimensions as the Abalone dataset, it is likely that the lower starting boundary model parity and target model accuracy (70.8% vs. 84.4%) are due to the Diabetes dataset being more non-linear than the Abalone dataset. Moreover, even though both MTRE-PAN and TRE-PAN are close in terms of boundary model parity at depths 6 and 7 (i.e., 14.37% and 13.18%, respectively); MTRE-PAN converges faster over 4 levels reaching a boundary model parity of 45.17%

compared to the 23.43% boundary model parity achieved by TRE-PAN over the same number of levels.

4.5. Explanation

TRE-PAN uses the decision tree as the explanation of the target model [19]. However, this decision tree may not represent the decision boundaries of the target model. In fact, as shown in the above tables, at extremely shallow depths, model parity can reach 90% while the boundary model parity is low (e.g., Table 3).

MTRE-PAN represents the target model using several matrices that place constraints on subsets of the feature space of the target model. These subsets can be certain positive, certain negative, or uncertain (i.e., boundary polygons). The constraints take the form of $x^T C \leq \vec{b}$ as indicated in Equations (1) and (2). Since the uncertain polygons contain the boundary of the target model, they can be used to generate linear approximations of the decision boundaries. Each linear approximation created by this approach is similar to the explanation employed by LIME [18]. However, in the case of MTRE-PAN, the linear approximation is bounded within the uncertain polygon used to generate it.

The explanation process is illustrated using the Diabetes Diagnosis dataset. The MTRE-PAN model at depth 8 shown in Table 8 has a total of 85 polygons. Each of these polygons is constrained by a bounding box, where the bounds are defined by the maximum and minimum of each feature as shown in Table 10. For brevity, the features of the Diabetes Diagnosis Dataset are abbreviated as follows:

- NTP: Number of times pregnant;
- GTT: Plasma glucose concentration at 2 h in an oral glucose tolerance test;
- DBP: Diastolic blood pressure (mm Hg);
- TST: Triceps skin fold thickness (mm);
- $2SI$: 2-h serum insulin (μU/ml);
- BMI: Body mass index (weight in kg/(height in m)^2);
- DPF: Diabetes pedigree function;
- AGE: Age (years).

Table 10. Bounding box defined by the maximum and minimum of each feature in the input space.

	NTP	***GTT***	***DBP***	***TST***	***2SI***	***BMI***	***DPF***	***AGE***
Maximum	15	199	122	99	846	67.1	2.29	70
Minimum	0	56	0	0	0	0	0.08	21

Equations (10)–(12) each represents a different polygon from the set of 85 polygons mentioned above. Specifically, Equations (10) and (11) represent a certain positive and a certain negative polygons, respectively. Equation (12) depicts an uncertain polygon (i.e., boundary polygon) with a linear estimate of the boundary as defined in Equation (13).

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \end{bmatrix} * \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} C_1 & C_2 \\ \end{matrix} \begin{bmatrix} -0.04 & -0.04 \\ -0.02 & -0.02 \\ -0.01 & 0 \\ 0 & 0 \\ 0 & 0 \\ -0.02 & -0.03 \\ -1 & -1 \\ 0.01 & 0.02 \end{bmatrix} \begin{matrix} w_{NTP} \\ w_{GTT} \\ w_{DBP} \\ w_{TST} \\ w_{2SI} \\ w_{BMI} \\ w_{DPF} \\ w_{AGE} \end{matrix} \leq \begin{matrix} b_1 & b_2 \\ \end{matrix} \begin{bmatrix} -5.28 & -5.26 \end{bmatrix} \quad (10)$$

$$\begin{matrix} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & \\ \left[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\right] * & \left[\begin{matrix} 0.04 & 0.08 & 0.24 & 0.13 & -0.60 & 0.47 \\ 0.02 & 0.02 & 0 & 0.01 & 0.02 & -0.05 \\ 0.01 & -0.02 & -0.09 & -0.1 & 0.26 & -0.20 \\ 0 & 0.02 & 0.08 & 0.04 & -0.25 & 0.21 \\ 0 & 0.01 & 0.04 & 0.02 & -0.11 & 0.09 \\ 0.02 & 0.02 & 0.04 & 0.11 & -0.17 & -0.12 \\ 1 & 1 & 0.96 & 0.98 & -0.67 & -0.81 \\ -0.01 & 0.01 & 0.04 & 0.02 & -0.13 & 0.12 \end{matrix}\right] & \begin{matrix} w_{NTP} \\ w_{GTT} \\ w_{DBP} \\ w_{TST} \\ w_{2SI} \\ w_{BMI} \\ w_{DPF} \\ w_{AGE} \end{matrix} \end{matrix}$$

$$\leq \begin{matrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ \left[5.29 & 5.31 & 4.68 & 4.07 & -4.06 & 1.54\right] \end{matrix} \quad (11)$$

$$\begin{matrix} & C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & \\ \left[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\right] * & \left[\begin{matrix} 0.04 & -0.08 & -0.09 & -0.03 & 0 & -0.44 \\ 0.02 & -0.02 & 0.11 & 0.05 & 0.04 & -0.01 \\ 0.01 & 0.02 & 0.17 & 0.04 & 0.02 & 0.07 \\ 0 & -0.02 & -0.09 & -0.02 & -0.02 & 0.03 \\ 0 & -0.01 & -0.01 & 0 & 0 & 0.03 \\ 0.02 & -0.02 & -0.08 & -0.02 & 0 & -0.34 \\ 1 & -1 & 0.97 & 1 & 1 & 0.73 \\ -0.01 & -0.01 & 0.01 & -0.05 & -0.03 & -0.38 \end{matrix}\right] & \begin{matrix} w_{NTP} \\ w_{GTT} \\ w_{DBP} \\ w_{TST} \\ w_{2SI} \\ w_{BMI} \\ w_{DPF} \\ w_{AGE} \end{matrix} \end{matrix}$$

$$\leq \begin{matrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ \left[5.29 & -5.31 & 3.02 & 0.63 & 0.36 & -18.31\right] \end{matrix} \quad (12)$$

$$\left[x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\right] * \begin{matrix} C_{estimate} \\ \left[\begin{matrix} -0.04 \\ -0.04 \\ -0.03 \\ 0.04 \\ 0.01 \\ -0.02 \\ -1 \\ -0.02 \end{matrix}\right] \end{matrix} \begin{matrix} w_{NTP} \\ w_{GTT} \\ w_{DBP} \\ w_{TST} \\ w_{2SI} \\ w_{BMI} \\ w_{DPF} \\ w_{AGE} \end{matrix} \leq \begin{matrix} b \\ \left[-1.18\right] \end{matrix} \quad (13)$$

According to Equation (10), MTRE-PAN indicates that at depth 8, all the patients that fall within this specific certain polygon are considered positive for diabetes. The original Diabetes dataset includes 92 such patients, one of whom is shown in Table 11. Moreover, based on the weights of the features in Equation (10), TST and $2SI$ do not contribute to the positive prediction in this polygon, whereas the most predictive feature is DPF. This latter feature represents the existence of a family history of diabetes for the patient and is known to be a significant predictive factor for diabetes [37].

Table 11. Example patient in the positive polygon of Equation (10).

NTP	*GTT*	*DBP*	*TST*	*2SI*	*BMI*	*DPF*	*AGE*	*Label*
2	128	78	37	182	43.3	1.224	31	positive

5. Discussion

The performance metrics used in the present study facilitate the comparison of the characteristics of MTRE-PAN and TRE-PAN. Model parity measures the difference between

the interpretable model and the target model without any distinction between certain and uncertain polygons. This metric is subject to sudden changes in accuracy due to the uncertain polygons being treated as though they are certain. This trend can be observed in Tables 4–7. For example in Table 4, the model parity at depth 2 increases to 93% from its previous value of 35% while the certain model parity remains at 0%. This means that all of the nodes currently in the tree exceed the cutoff entropy and are uncertain. The model parity stabilizes as the depth and certain model parity increase. When the total area of the uncertain polygons diminishes, the variance of the model parity also diminishes since most of the data which now reside within certain polygons will never be re-labeled.

Certain model parity increases as the depth of the tree in the interpretable model increases. This metric includes an uncertain class label for the data that fall within uncertain nodes. This intermediary class is always considered to be a miss when calculating the certain model parity. Therefore, a given sample will not transition from a true positive/negative to a false positive/negative as the tree is expanded. However, this transition can occur when model parity is used. The entropy of a given polygon may fall below the cutoff entropy and become certain, and as a result, a sample might be falsely labeled. This case does not affect certain model parity since the sample was considered falsely labeled before the split occurred.

Overall the certain model parity is a better indicator of the convergence of the interpretable model to the decision boundaries of the target model. A major difference between MTRE-PAN and TRE-PAN can be observed for the Census data in Tables 6 and 7. Based on certain model parity, MTRE-PAN at depth 3 approximates the target model as well as TRE-PAN at depth 9. Similar trends can be observed for the other datasets.

The ultimate goal of global translation is to translate the target model into an interpretable model. Therefore, boundary model parity is the most important metric because it illustrates progress towards this goal. It is a test of the ability of the model to represent the decision boundaries of the target model without the potential for easily interpretable data to exaggerate the accuracy of the translation. Any interpretable model generated by MTRE-PAN or TRE-PAN can achieve high levels of certain model parity. However, if the decision boundaries are complex, the model will fail at shallow depths. This can be observed when the Abalone (Table 4) and Diabetes models (Table 8) are compared. Both datasets have the same number of dimensions (i.e., 8 input features each). However, the decision boundaries for the diabetes dataset are more complex. Although both models have similar certain model parity, the boundary parity for the diabetes model does not converge easily.

The leaf count serves as an indication of the complexity when extracting information from the interpretable models. Since MTRE-PAN is able to achieve certain model parity comparable to TRE-PAN at shallow depths, it requires a significantly lower number of nodes. For instance, when comparing Tables 6 and 7, MTRE-PAN requires only 4 leaf nodes to represent the behavior of the target model nearly as well as TRE-PAN with 173 leaf nodes. For every added level, the leaf count doubles.

Both TRE-PAN and MTRE-PAN build a tree that represents the target model. The explanation in TRE-PAN is a decision tree which can be expressed as a set of rules [20,21]. However, this study shows that TRE-PAN has limitations, as it is unable to provide an accurate explanation of the decision boundaries of the target model. In contrast, MTRE-PAN seeks to directly represent the decision boundaries of the target model. This is accomplished by translating these decision boundaries to a set of bounded linear estimators. One of the main benefits of this approach is that it reduces the complexity traditionally associated with providing a global explanation of the target model.

6. Conclusions

This paper proposes MTRE-PAN, a global translation model that can be used for any target classification model. MTRE-PAN builds the explainable model as a hybrid combination of a decision tree and SVM linear separators at each node. This explainable

model organizes the feature space into a set of polygons that approaches the behavior of the target model.

MTRE-PAN was inspired by TRE-PAN. The latter uses a decision tree to partition the feature space, whereas the former uses linear hyperplanes. The performance of MTRE-PAN was compared to that of TRE-PAN for three target classification models developed using public domain datasets. The results show that MTRE-PAN achieves a higher parity across all metrics at shallower depths. The study also shows that certain model parity and boundary model parity are able to provide a better evaluation of the interpretable model than the traditional model parity. Certain model parity quantifies the accuracy of the interpretable model within low entropy regions. Boundary model parity specifically uses data within a margin of the decision boundaries to test the ability of the interpretable model to globally represent the decision boundaries of the target model.

MTRE-PAN has a higher computational complexity than TRE-PAN because it uses linear SVM classifiers at each node. However, as the results show, TRE-PAN requires a much deeper tree to achieve comparable results. The additional levels in the tree offset the difference in computational complexity. This is especially true for highly non-linear target models.

Future work includes the pruning of the MTRE-PAN interpretable tree model around the decision boundaries. Dynamic pruning can help develop more compact global explanation by pruning old constraints as new ones are added. MTRE-PAN can also be improved by developing a loss function that is specific to each target model. This loss function could use the probability distribution produced by the target model to locate the decision boundaries. However, this will be at the cost of a more model-dependent methodology. Finally, MTRE-PAN should be compared to other global interpretable models, and it its ability to interpret biased data should be evaluated.

Author Contributions: Conceptualization, M.A.-M. and Z.B.M.; methodology, M.A.-M. and Z.B.M.; software, M.A.-M.; validation, M.A.-M. and Z.B.M.; formal analysis, M.A.-M. and Z.B.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Ethical review and approval are not required because all datasets used in this study are in the public domain.

Data Availability Statement: MTRE-PAN implementation https://github.com/malmerri42/M-TRE-PAN-Release (accessed on 21 February 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Goodman, B.; Flaxman, S. European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation". *AI Mag.* **2017**, *38*, 50–57. [CrossRef]
2. The Administration's Report on the Future of Artificial Intelligence. 2016. Available online: https://obamawhitehouse.archives.gov/blog/2016/10/12/administrations-report-future-artificial-intelligence (accessed on 21 February 2022).
3. Stefanik, E.M. H.R.5356—115th Congress (2017–2018): National Security Commission Artificial Intelligence Act of 2018. 2018. Available online: https://www.congress.gov/bill/115th-congress/house-bill/5356 (accessed on 21 February 2022).
4. Chouldechova, A. Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments. *arXiv* **2016**, arXiv:1610.07524. [CrossRef] [PubMed]
5. Supreme Court of Wisconsin, State v. Loomis. Available online: https://scholar.google.com/scholar_case?case=3222116451721963278&hl=en&as_sdt=6&as_vis=1&oi=scholarr (accessed on 21 February 2022).
6. Freeman, K. Algorithmic Injustice: How the Wisconsin Supreme Court Failed to Protect Due Process Rights in State v. Loomis. *N. Carol. J. Law Technol.* **2016**, *18*, 75.
7. Ferguson, A. Policing Predictive Policing. *Wash. Univ. Law Rev.* **2017**, *94*, 1109–1189.
8. Chatila, R.; Havens, J.C. The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. In *Robotics and Well-Being*; Aldinhas Ferreira, M.I., Silva Sequeira, J., Singh Virk, G., Tokhi, M.O., Kadar, E.E., Eds.; Springer International Publishing: Cham, Switzerland, 2019; Volume 95, pp. 11–16. [CrossRef]
9. Ben-Gal, I. Bayesian Networks. In *Encyclopedia of Statistics in Quality and Reliability*; Wiley Online Library: Hoboken, NJ, USA, 2008. [CrossRef]

10. Quinlan, J.R. Induction of Decision Trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
11. Ho, T.K. Random Decision Forests. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, Canada, 14–16 August 1995; Volume 1, pp. 278–282. [CrossRef]
12. Schmidhuber, J. Deep Learning in Neural Networks: An Overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]
13. Cortes, C.; Vapnik, V. Support-Vector Networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
14. Kim, Y.S. Comparison of the Decision Tree, Artificial Neural Network, and Linear Regression Methods Based on the Number and Types of Independent Variables and Sample Size. *Expert Syst. Appl.* **2008**, *34*, 1227–1234. [CrossRef]
15. Lipton, Z.C. The Mythos of Model Interpretability. *arXiv* **2016**, arXiv:1606.03490.
16. LeCun, Y.; Haffner, P.; Bottou, L.; Bengio, Y. Object Recognition with Gradient-Based Learning. In *Shape, Contour and Grouping in Computer Vision*; Lecture Notes in Computer Science; Forsyth, D.A., Mundy, J.L., di Gesú, V., Cipolla, R., Eds.; Springer: Berlin/Heidelberg, Germeny, 1999; pp. 319–345. [CrossRef]
17. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning Representations by Back-Propagating Errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
18. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why Should I Trust You?: Explaining the Predictions of Any Classifier. *arXiv* **2016**, arXiv:1602.04938.
19. Craven, M.W.; Shavlik, J.W. Extracting Tree-Structured Representations of Trained Networks. In Proceedings of the 8th International Conference on Neural Information Processing Systems, Denver, CO, USA, 27 November–2 December 1995; Volume 1, pp. 24–30.
20. Sato, M.; Tsukimoto, H. Rule Extraction from Neural Networks via Decision Tree Induction. In Proceedings of the IJCNN'01. International Joint Conference on Neural Networks, Washington, DC, USA, 15–19 July 2001; Volume 3, pp. 1870–1875. [CrossRef]
21. Augasta, M.; Kathirvalavakumar, T. Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems. *Neural Process. Lett.* **2012**, *35*, 131–150. [CrossRef]
22. Zilke, J.R.; Loza Mencía, E.; Janssen, F. DeepRED – Rule Extraction from Deep Neural Networks. In *Discovery Science*; Calders, T., Ceci, M., Malerba, D., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9956, pp. 457–473. [CrossRef]
23. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.
24. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning Important Features Through Propagating Activation Differences. In Proceedings of the International conference on machine learning, PMLR, Sydney, Australia, 6–11 August 2017; p. 9.
25. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In Proceedings of the Workshop at the International Conference on Learning Representations, Banff, AB, Canada, 14–24 June 2014.
26. Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.R.; Samek, W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE* **2015**, *10*, e0130140. [CrossRef] [PubMed]
27. Lundberg, S.M.; Lee, S.I. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
28. Slack, D.; Hilgard, S.; Jia, E.; Singh, S.; Lakkaraju, H. Fooling LIME and SHAP: Adversarial Attacks on Post Hoc Explanation Methods. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, 7–9 February 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 180–186.
29. Tabacof, P.; Valle, E. Exploring the Space of Adversarial Images. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; pp. 426–433. [CrossRef]
30. Bennett, K.; Blue, J. A Support Vector Machine Approach to Decision Trees. In Proceedings of the 1998 IEEE International Joint Conference on Neural Networks Proceedings, IEEE World Congress on Computational Intelligence (Cat. No.98CH36227), Anchorage, AK, USA, 4–9 May 1998; Volume 3, pp. 2396–2401. [CrossRef]
31. Madzarov, G.; Gjorgjevikj, D. Multi-Class Classification Using Support Vector Machines in Decision Tree Architecture. In Proceedings of the IEEE EUROCON 2009, St. Petersburg, Russia, 18–23 May 2009; pp. 288–295. [CrossRef]
32. Russell, S.; Norvig, P. *Artificial Intelligence: A Modern Approach*, 3rd ed.; Pearson: Upper Saddle River, NJ, USA, 2009.
33. Craven, M. Extracting Comprehensive Models From Trained Neural Networks. Ph.D. Thesis, University of Wisconsin-Madison, Madison, WI, USA, 1996.
34. UCI Machine Learning Repository: Abalone Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/abalone (accessed on 21 February 2022).
35. UCI Machine Learning Repository: Census Income Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/census+income (accessed on 21 February 2022).
36. Smith, J.W.; Everhart, J.; Dickson, W.; Knowler, W.; Johannes, R. Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus. In Proceedings of the Annual Symposium on Computer Application in Medical Care, Bethesda, MD, USA, 7–11 November 1988; American Medical Informatics Association: Bethesda, MD, USA, 1988; pp. 261–265.
37. Joshi, R.D.; Dhakal, C.K. Predicting Type 2 Diabetes Using Logistic Regression and Machine Learning Approaches. *Int. J. Environ. Res. Public Health* **2021**, *18*, 7346. [CrossRef] [PubMed]