GLOBAL TRANSLATION OF MACHINE LEARNING MODELS TO INTERPRETABLE MODELS

by

Mohammad Almerri

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering

Indianapolis, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Zina Ben Miled, Chair

Department of Electrical & Computer Engineering

Dr. Lauren Christopher

Department of Electrical & Computer Engineering

Dr. Paul Salama

Department of Electrical & Computer Engineering

Approved by:

Dr. Brian King

To my brother, Abdulmohsen.

ACKNOWLEDGMENTS

I would like to thank Dr. Zina Ben Miled for her support and patience; Dr. Paul Salama, and Dr. Lauren Christopher for their guidance; and Sherrie A. Tucker for her assistance in editing and formatting this thesis. Finally, I would like to extend my thanks to the Electrical and Computer Engineering department for making available all the needed resources to support my project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The widespread and growing usage of machine learning models, especially in highly critical areas such as law, predicate the need for interpretable models. Models that cannot be audited are vulnerable to inheriting biases from the dataset. Even locally interpretable models are vulnerable to adversarial attack. To address this issue a new methodology is proposed to translate any existing machine learning model into a globally interpretable one. This methodology, MTRE-PAN, is designed as a hybrid SVM-decision tree model and leverages the interpretability of linear hyperplanes. MTRE-PAN uses this hybrid model to create polygons that act as intermediates for the decision boundary. MTRE-PAN is compared to a previously proposed model, TRE-PAN, on three non-synthetic datasets: Abalone, Census and Diabetes data. TRE-PAN translates a machine learning model to a 2-3 decision tree in order to provide global interpretability for the target model. The datasets are each used to train a Neural Network that represents the non-interpretable model. For all target models, the results show that MTRE-PAN generates interpretable decision trees that have a lower number of leaves and higher parity compared to TRE-PAN.

1. INTRODUCTION

Since 2018, the European Union has placed regulations on personal data usage, and algorithmic decision making systems [1]. European Union citizens may as a result have a “right to explanation”, whereby they are entitled an explanation to an algorithmic decision and have the ability to contest it [1]. In the United States, 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” to establish a formal committee to review the usage of AI and ultimately recommend necessary regulations [3]. As algorithmic tools grow in use, laws regulating the ethical use of these tools with the intent to help prevent misuse as well as reduce potential algorithmic decision failures are likely to continue to appear globally.

However, such laws are double edged, since the ones writing them are unlikely to posses the necessary technical information required to judge the validity of any individual model when it is used to recommend critical decisions that can affect human life. The continued and growing usage of various permissible legal machine learning (ML) models in court in spite of their many public failures supports this observation. One such instance is the widespread usage of recidivism prediction instruments, which attracted controversy as these instruments have the potential to inherit biases present in the training data, especially when applied without consideration of the distribution of the data [4]. In the United States, 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 all of the methodology of the algorithm was a trade secret. Therefore the court only saw and used the output of the algorithm in deciding the sentence. 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]. A racial bias causes a system to over-target people of a certain race, generating more biased training data as a result of the increased arrests [4]. In order to overcome some of these legal and ethical pitfalls, ML models need to be interpretable, and therefore open to auditing. [7].

In response to this concern the IEEE has 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 specifically addressing AI standardization. Subsequently, the P7001 and P7003 standards required transparency and algorithmic bias considerations for autonomous systems. These standards emphasize transparency and as a consequence the need for interpretability for all ML models.

In general, ML models can often be classified into two main categories in terms of interpretability. The first class consists of easily interpretable models such as Bayesian Networks [9], Decision Trees [10] and to some extent Random Forests [11]. The second class includes more complex models such as neural networks [12] and Support Vector Machines [13]. This second class of models is often more accurate and generalizes better to new data [14]. However, it suffers from a reduced interpretability [15]. In fact, the more complex the model, the less interpretable it becomes. Examples include Deep Neural Networks [12], Convolution Neural Network [16], and Recurrent Networks [17]. Similarly, the interpretability of SVM decreases with high 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 explanations*. Transparency aims at delivering a model level, or *global* level of interpretability, and post-hoc explanations is a per input "after the fact" explanation, that provides a *local* level of interpretability. Both the local and the global interpretability of complex models have been investigated in previous studies. These studies propose a translation mechanism where a non-interpretable model is first developed and then it is translated to an interpretable model. For instance, the Local Interpretable Model-agnostic Explanations (LIME) [18] technique translates a non-interpretable model to a locally interpretable one by generating data around a query from the non-interpretable model. The resulting input and labeled data is then 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]. TRE-PAN translates a neural network by training a decision tree model using data generated from the original model. Another global translation technique is CRED [20]. This technique uses the internal structure of a neural network to generate rules from an induced decision tree.

Global interpretation is the focus of this thesis. A decision tree with linear hyperplanes as separators is used to provide global interpretability for a complex neural network model. The main benefit of this approach compared to previously proposed global translation techniques is that it generates a more compact explainable model.

2. RELATED WORK

Several methods for translating non-interpretable ML models to interpretable ML models have been proposed in the literature [18], [19]. These methods fall under two categories: *Transparency* and *Post-hoc Interpretability*, or in other words *Global Translation* and *Local translation*, respectively [15]. Global translation corresponds to transparency since its aim is to provide a holistic understanding of the behavior of the target model. Local translation corresponds to post-hoc interpretation as it is done after the fact and only 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; and *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 behavior of the target model by translating it into a model whose behavior is understood [15]. For instance, TRE-PAN is an algorithm that generates a decision tree which describes the behavior of a deep neural network [16], [17]. Under this approach, the deep neural network is already trained, and the training data is available. TRE-PAN uses the trained network to generate data that is in turn used to train the interpretable decision tree. As the tree is being trained, the original network is used to generate as many examples as needed to help define the best splits for each node in the tree. Previous nodes in the tree are used as constraints for the input features of the deep neural network.

TRE-PAN uses 2 of 3 decision trees. Instead of performing a binary split of the sample data using a single feature that maximizes an individual gain, these trees use 2 out 3 features to perform the split at every node. That is, for each node, the top three features are selected and the corresponding thresholds are established based on the potential gain in entropy from the split. 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. This 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 2 out of 3 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 going to be greater in comparison to regions further away from the decision boundaries. This is anticipated since near the decision boundaries, the data will have a more balanced proportion of positive and negative samples, requiring a significantly higher number of splits to represent each boundary.

When the decision boundary of the neural network has a non-linear shape, representing the area constrained by this shape, using a singular dimension hyperplanes is analogous to representing the constrained area with 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 will come at a cost of lower accuracy [19].

The above global translation approach used in TRE-PAN treats the target model as a black box and is applicable to several ML models. An alternative approach that is 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.

2.2 Local Translation

The objective of *Local Translation* is to observe a limited subset of the feature space and attempt to explain it by using specific input examples. Therefore, the focus 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 target 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 is then weighted by using a distance metric from the original input and used to train the interpretable linear model. By observing this linear model, in particular the feature weights, one can identify the most important features with respect to the classification. This is only a locally valid interpretation.

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] since a single hyperplane is unlikely to be able to capture the behavior of the boundary.

Other local translation techniques include utilizing a heat map to visualize the activation patterns in the input data [15], [21]. For instance, given an input image, the pixels which were the most influential in selecting a predicted class can be identified [22].

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 miss-classify the image [23]. A globally interpretable model is more resilient to such an adversarial attack since it provides the opportunity to audit how a given decision is reached which will in turn reveal any gaps in the model’s inference.

3. METHODOLOGY

The global translation technique proposed in this thesis translates a 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 [24] and [25], is extended to global translation. The proposed technique, MTRE-PAN, leverages the interpretability of the decision tree and the generalizability of SVM in order to generate an interpretable model. It uses SVM with linear kernels instead of a higher order kernel in order to facilitate interpretability. Moreover, MTRE-PAN only requires a function that returns a binary output. However, in the present thesis it is demonstrated for neural networks.

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 $\mathbf{b_k} \in \mathbb{R}^N$. 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. This data consists 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 are defined below:

$$\text{left child: } Q_l = \{\forall \mathbf{x} \in Q_{Parent(l)} \mid \mathbf{x}_{\mathrm{i}}^T \mathbf{C}_{Parent(l)} \leq b_{Parent(l)}\} \tag{3.1}$$

$$\text{right child: } Q_r = \{\forall \mathbf{x} \in Q_{Parent(r)} \mid \mathbf{x}_{\mathrm{i}}^T \mathbf{C}_{Parent(r)} > b_{Parent(r)}\} \tag{3.2}$$

Leaf nodes inherit the label from the SVM classifier according to the side of the split they fall into. This relationship between the parent node and the child nodes is exemplified in Figures 3.1 and 3.2, where $f(x)$ is a circle. The right child of the root node is expanded

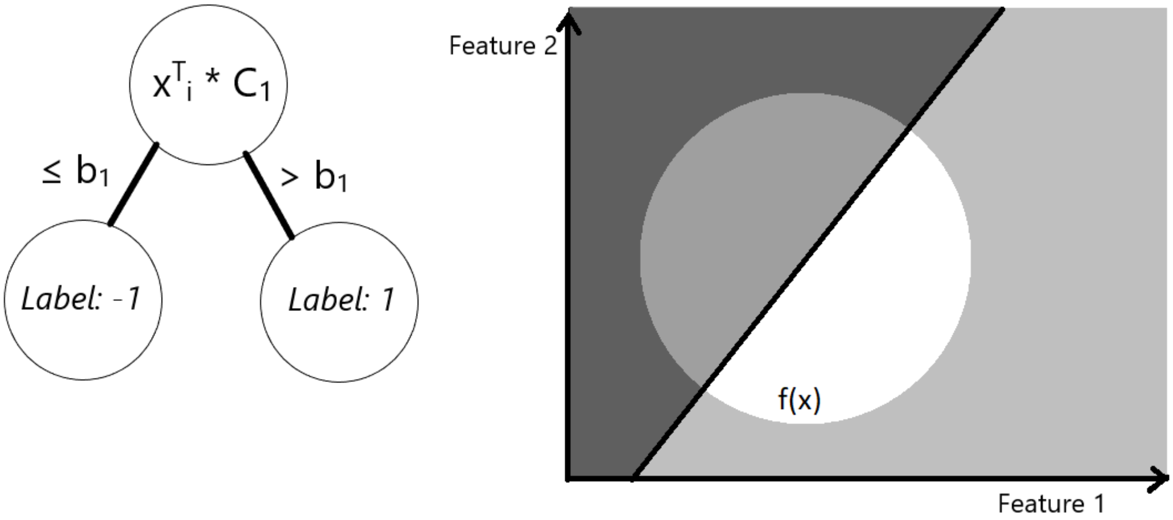


Figure 3.1. Parent-child sampling in MTRE-PAN.

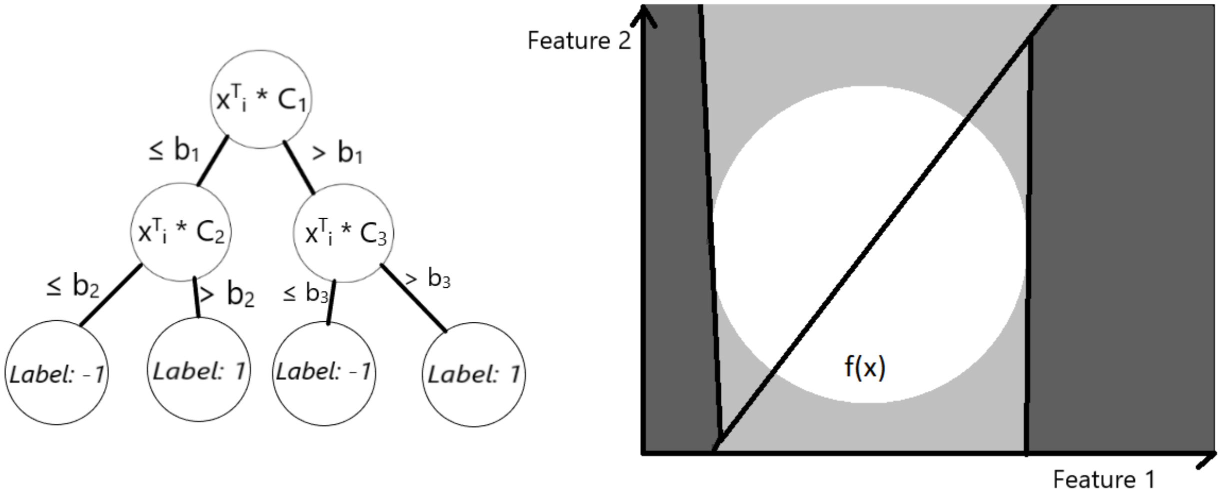


Figure 3.2. Multilevel expansion of the decision tree generated by MTRE-PAN.

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 $\mathbf{x}_\mathrm{i}^T\mathbf{C}_1 > b_1$. 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 training must satisfy $\mathbf{x}_\mathrm{i}^T\mathbf{C}_1 \leq b_1$.

The MTRE-PAN algorithm maintains the nodes in a queue. The decision to split the node is made according to a standard binary gain measure G given by:

$$G(Q_k) = -E(\frac{p}{p+n}) \tag{3.3}$$

where the entropy E is defined as:

$$E(q) = -(q \log_2 q + (1-q)log_2(1-q)) \tag{3.4}$$

where Q_k is the data received from the parent node; and p, and n are the number of positive and negative examples of Q_k, respectively [26]. Nodes whose gains falls below a preset threshold are considered uncertain and therefore are candidates for splitting. After being split, new leaf nodes are sampled for data and added into the queue. The steps of MTRE-PAN are outlined in algorithms 1, 2 and 3. Each node generated by MTRE-PAN may sample the original non-interpretable model for more training data if the available data is not sufficient to represent the subspace. This is accomplished by calling *Sampling-on-demand* as discussed next.

3.2 Sampling on Demand

The need for additional training data is measured by comparing the sample variance of the entropy to a user-defined threshold, the variance cutoff. Increasing this threshold improves the accuracy of the classifier when the leaf is split, and improves the accuracy of the measured entropy (by lowering it's variance). However, it also increases the space and time required by the sampling computation. The sampling process is repeated in the subspace of interest until the sample variance of the entropy converges to a value below the above-mentioned threshold as shown in Algorithm 3.

Algorithm 1: M-TRE-PAN Algorithm.

```
Input:      Training data and labels: (Q,Y)
            Entropy threshold: entropyT
            Non-interpretable model oracle: $f(x)$
            Variance cutoff: varCutoff

Initialize: Root node: R
            R.data = (Q,Y)
            Gain priority queue: Pg = {}
            Pg.enqueue(R)

while Pg is not empty do
    node = pG.dequeue()
    [leftChild, rightChild] = node.train() /* Trains the parameters of the node, propagates data to children, and initializes them as leaf nodes. */
    leftChild.Sample_On_Demand($f(X)$, leftChild.data, leftChild.constraints, varCutoff)
    rightChild.Sample_On_Demand($f(X)$, rightChild.data, rightChild.constraints, varCutoff)
    if leftChild.Get_Entropy() > entropyT then
        Pg.enqueue(leftChild)
    end
    if rightChild.Get_Entropy() > entropyT then
        Pg.enqueue(rightChild)
    end
end
```

Algorithm 2: Node splitting.

```
/* trains the weights and bias based on data                              */
[weights, bias ] = LinearFit(data)

/* Partitions the data, as defined in Equation (3.1) for the left
   child, and (3.2) for the right child                                   */
[leftData, rightData ] = LinearSplit(data,weights, bias)

/* Generate more data for each child if needed:                           */
leftData = SampleOnDemand(pWeightsC ∪ weights, pBiasB ∪ bias, leftData)

rightData = SampleOnDemand(pWeightsC ∪ −1∗weights, pBiasB ∪ −1∗bias,
 rightData)

/* Children are initialized as leaf nodes, where the majority class in
   the propagated dataset is the class label of the leaf                  */
leftChild = Create_leaf (leftData)
rightChild = Create_leaf (rightData)
```

Algorithm 3: Sample on demand

```
Input:      Non-interpretable model: $f(\mathbf{x})$;
            Complete Data $Q$: Data;
            Parent constraints of the current node: $C_k$ and $b_k$;
            User defined sample variance cutoff: sVarianceCutoff;

Initialize: Set GenData, the set of generated data to Data;
            Set sAvg to the sample average of the input Data;
            Set internalBoundingBox to contain all Data in a bounding box;

while the sample variance of sAvg is less than sVarianceCutoff do
    tempSample ← Sample from $f(\mathbf{x})$, bounded by internalBoundingBox
    if tempSample ∈ Parent constraints then
        Data ← Data ∪ tempSample
        sAvg ← SampleAverage(Data)
```

As the MTRE-PAN tree is expanded, each node is concerned with classifying a subspace $M \subseteq \mathbb{R}^N$ of the original feature space $\mathbb{R}^N$ that is defined by a hyper-polygon. After a node is split, the data that belongs to the leaf node (as defined by equations 3.1 and 3.2) is surrounded by a bounding box. Afterwards, data is uniformly sampled for the leaf nodes in the intermediary tree based on the dimensions of their respective bounding boxes. The samples are then filtered according to the constraints associated with each node. This sampling continues until the sample variance of the entropy falls below the variance cutoff.

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 2 out of 3 split as described in Section 2.1. In order to simplify the comparison with MTRE-PAN, in particular with respect to the depth of the resulting interpretable models, the C4.5 binary implementation of TRE-PAN was used [27]. MTRE-PAN, the model proposed in the present thesis, 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 the positive samples that is populated with synthetic data. The remaining target models are feed forward neural network models which are trained using three public domain datasets.

Table 4.1. Hyper-parameters definition and value for experimental setup.

Hyper-parameter	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 therefore, is a candidate for further splitting.	0.0808
Cutoff Variance	An upper limit on the number of data points sampled in a current leaf. It is based on the sample variance of the entropy. A lower cutoff variance will result in a more accurate value for the sample entropy by lowering it's variance.	10^-5
Margin:	The width around the decision boundary of the target ML from which data is 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

The hyper-parameters of MTRE-PAN and TRE-PAN include the Maximum Depth, the Cutoff Entropy, the Cutoff Variance and the Margin which are described in Table 4.1. The ML target models consist of the following general architecture:

- 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.

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 is normalized to [-1,1]. After training, the model that achieved the highest accuracy on the validation data across all the epochs of a dataset is used for that dataset.

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

- Model Parity: The accuracy of the resulting decision tree with respect to the underlying 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 $\frac{TP+TN}{TP+TN+FP+FN} * 100$.
- Certain Model Parity: This metric is similar to the Model Parity. Except, in this case, the sample in the validation data set that are assigned to uncertain nodes (i.e., nodes whose entropy is below the Cutoff Entropy) are labeled *uncertain*. Theses samples cannot match any label from f and as such are considered as misses. This metric is calculated as $\frac{TP+TN}{TP+TN+FP+FN+uncertain} * 100$ and allows the comparison of two techniques while taking into consideration uncertain nodes that need further expansion.
- Boundary Model Parity: This metric is similar to the Certain Model Parity. However, in this case, the validation data set is filtered to only include the samples that are near the decision boundary within a predefined margin. The Boundary Model Parity measures the progress of the interpretable model towards replicating the behavior of

the target model near the decision boundaries. It identifies an interpretable model that has a high Certain Model Parity but may not fair well around the decision boundaries.

- Leaf Count: The number of leaves in the decision tree generated by either MTRE-PAN or TRE-PAN.

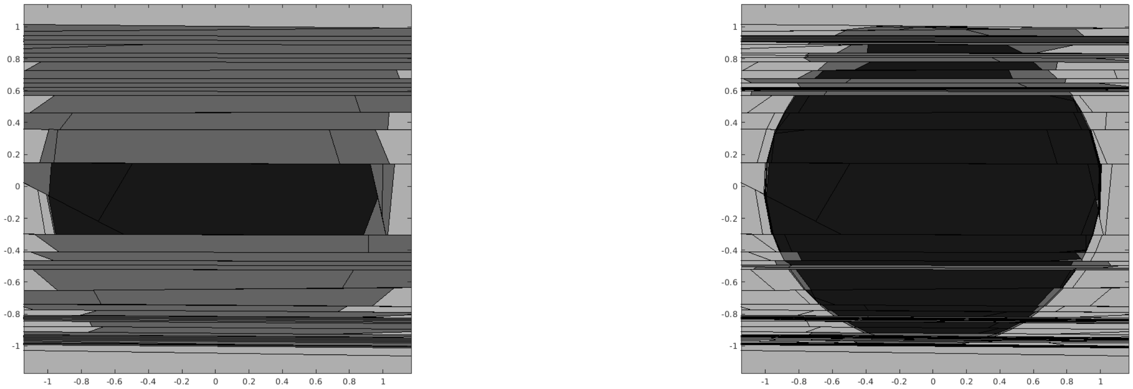
4.1 Synthetic Data

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. 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 Tables 4.2 and 4.3, respectively.

Figure 4.1 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 is smaller at depth 12 than at depth 9. Moreover, as expected, the Figure shows that the uncertain polygons always contain the decision boundaries of f. Otherwise, it would not be possible for opposing labeled data to appear in the polygon, giving the polygon an entropy of 0, therefore making it certain rather than uncertain. This behavior is also seen in the TRE-PAN example in Figure 4.2. As in MTRE-PAN, the uncertain polygons (hyper-rectangles in this case) also contain the decision boundary. Due to this, uncertain polygons can be used to represent the underlying decision boundary. However in the case of TRE-PAN, this representation of the decision boundary generally requires more nodes compared to MTRE-PAN.

Since uncertain polygons are mutually exclusive, they can be used as an estimation for the decision boundary and therefore the overall behavior of the underlying model. As mentioned



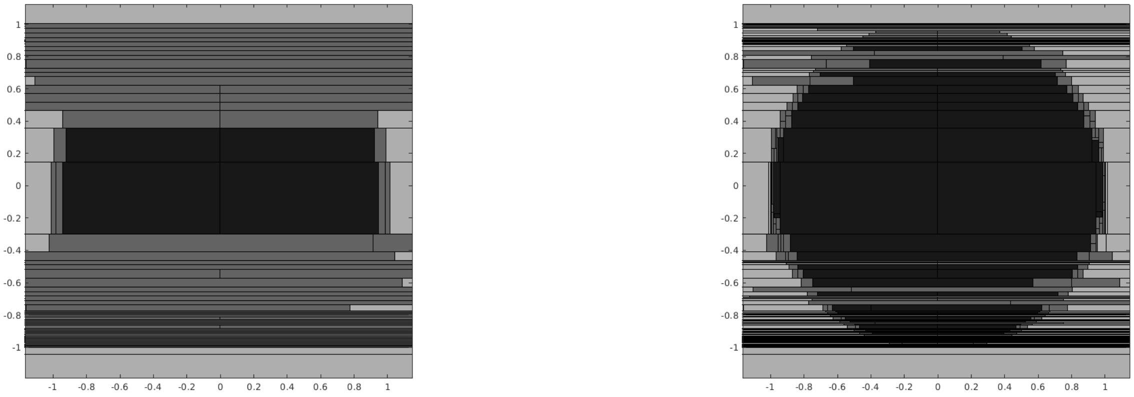
(a) Depth = 9

(b) Depth = 12

Figure 4.1. 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 light and dark shaded polygons define the boundaries of the negative and positive samples, respectively. The medium shaded polygons represent uncertain regions.

above, the accuracy of the model in representing the decision boundary is dependant on the values set for 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 is 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 lead to potentially 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.



(a) Depth = 9

(b) Depth = 12

Figure 4.2. 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 light and dark shaded polygons define the boundaries of the negative and positive samples, respectively. The medium shaded polygons represent uncertain regions.

MTRE-PAN provides a global explanation of the underlying function f in the form of a set of uncertain polygons, and of polygons that have an estimated boundary as a constraint. The polygon's other constraints define the limits of this estimate. This prevents LIME's unbounded plane issue. Whereby LIME generates a plane as a local estimate but never specifies the valid region in the feature space for the estimate.

4.2 Abalone Data

The Abalone dataset consists of recorded physical characteristics for the abalone mollusks [28]. It includes 4,177 samples where each sample has 8 features and an integer label representing the physical characteristics of the 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 the 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 met-

Table 4.2. Evaluation metrics for the circle function interpretable models generated by MTRE-PAN at depths ranging from 1 to 12.

Depth	Model Parity	Certain Model parity	Boundary Model parity	Leaf Count
1	91.50	0.00	0.00	1
2	42.88	34.38	0.00	2
3	78.65	34.38	0.00	3
4	81.75	66.53	0.00	5
5	85.22	66.53	0.00	8
6	84.58	66.53	0.00	14
7	87.70	69.85	0.00	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	94.40	69.12	206
12	98.90	96.75	82.35	305

rics of the corresponding interpretable models generated by MTRE-PAN and TRE-PAN are reported in Tables 4.4 and 4.5, respectively.

From the results, both MTRE-PAN and TRE-PAN approach the underlying 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 (7 leaf nodes vs 201 leaf nodes). The cost of training MTRE-PAN's linear classifiers is superseded by the exponentially higher number of splits that is needed for TRE-PAN to achieve a similar accuracy. It can also be argued that while simple in their separators, the sheer number of TRE-PAN nodes complicate the interpretability of the decision tree generated by TRE-PAN compared to shallower tree generated by MTRE-PAN.

Table 4.3. Evaluation metrics for the circle function interpretable models 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	0.00	0.00	1
2	50.50	0.00	0.00	2
3	77.72	0.00	0.00	4
4	81.33	64.85	0.00	8
5	83.45	64.85	0.00	14
6	83.58	66.20	0.00	26
7	87.15	70.50	0.00	49
8	88.70	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

4.3 US Adult Census Data

This dataset is a collection from the US 1994 adult census data [29]. 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 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 4.6 and 4.7, respectively.

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 a depth of 6 and 9 respectively to begin to converge to the boundary. While the Certain Model Parity increases steadily for both TRE-PAN and MTRE-PAN, the Boundary

Table 4.4. Evaluation metrics for the Abalone interpretable models generated by MTRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model parity	Boundary Model parity	Leaf Count
1	35.60	0.00	0.00	1
2	93.72	0.00	0.00	2
3	81.27	51.81	0.00	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

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 in the case of the Abolone dataset. This is also 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 [30]. 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-hour 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 4.8 and 4.9, respectively. Unlike the Census data, the boundary model parity converges faster. However the starting parity is still significantly lower compared to the Abalone data. Considering that this dataset has the same number of dimensions as the

Table 4.5. Evaluation metrics for the Abalone interpretable models 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	0.00	0.00	1
2	74.71	0.00	0.00	2
3	69.25	0.00	0.00	4
4	69.20	0.00	0.00	8
5	70.91	15.67	0.00	16
6	74.93	30.71	16.40	31
7	79.60	38.41	21.60	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

Abalone dataset, it is likely that the lower starting Boundary Model Parity and worse target model accuracy (70.8% vs 84.4%) are due to Diabetes dataset being more non-linear than the Abalone dataset. Moreover, even though both MTRE-PAN and TRE-PAN are closer in Boundary Model Parity at depths 6 and 7, and they have similar starting Parities (i.e., 14.37% and 13.18%, respectively); MTRE-PAN converges faster over 4 levels reaching a Parity of 45.17%, while TRE-PAN reaches only a Parity of 23.43%.

4.5 Discussion

Across all of the datasets, the performance metrics provide a comparative insight into the behavior of MTRE-PAN and TRE-PAN.

The Model Parity measures the difference between the behavior of the interpretable model produced by either M-TRE-PAN or TRE-PAN and the underlying 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: 4.5, 4.6, 4.7, and 4.4. For example in Table 4.4, the Model Parity at depth 2 increases to 93% from its previous value of 35% while the Certain Model Parity remains 0%. This means that all of the nodes currently in the tree exceed the

Table 4.6. Evaluation metrics for the US Adult Census interpretable models 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	0.00	0.00	1
2	95.76	0.00	0.00	2
3	90.53	84.25	0.00	4
4	95.37	84.25	0.00	7
5	95.24	88.64	0.00	13
6	95.86	89.21	12.50	24
7	95.64	89.68	12.50	45
8	96.02	89.68	12.50	86
9	96.11	90.32	15.57	168
10	96.40	90.73	21.05	327

Cutoff Entropy, and are uncertain. The Model Parity stabilizes as the depth and Certain Model Parity increase. As the total area of the uncertain polygons diminishes, the variance of the Model Parity also diminishes since most of the data which now resides within certain polygons, is never going to be re-labeled.

Certain Model Parity is an increasing function as the depth of the tree increases. Since this metric includes an *uncertain* label for all data that fall within *uncertain* nodes, this intermediary class is always considered to be a miss when calculating the parity. Therefore, a given sample will not transition from a True positive/negative to a False positive/negative as the tree is expanded. This case can occur when Model Parity is calculated. In the worst case, a polygon's Entropy may fall below the Cutoff Entropy and becomes *certain*, as a result a sample might be falsely labeled. However, this scenario does not affect Certain Parity since the sample was considered falsely labeled before the split occurred.

Overall the Certain Model Parity is a closer indicator of the convergence of the interpretable model to the target model's decision boundaries. A major difference between MTRE-PAN and TRE-PAN can be observed for the Census data in Tables 4.6, and 4.7. Based on Certain Model Parity, MTRE-PAN at depth 3 is as good of an approximation of

Table 4.7. Evaluation metrics for the US Adult Census interpretable models generated by TRE-PAN at depths ranging from 1 to 10.

Depth	Model Parity	Certain Model parity	Boundary Model parity	Leaf Count
1	5.30	0.00	0.00	1
2	92.21	0.00	0.00	2
3	80.76	72.92	0.00	4
4	90.94	72.92	0.00	7
5	91.21	72.92	0.00	13
6	90.16	78.64	0.00	25
7	92.35	80.08	0.00	47
8	92.56	82.10	0.00	90
9	93.07	84.25	1.90	173
10	93.65	84.84	1.90	330

the target model as TRE-PAN is at depth 9. Similar trends can be observed for the other datasets.

The ultimate goal of MTRE-PAN and TRE-PAN is to translate the target model into an interpretable model. Therefore, the Boundary Model Parity is the most important of the metrics in illustrating the progress towards this goal. It is a test of each model's ability to represent the decision boundaries of the target model without the potential for easily interpretable data to inflate 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 is exemplified when the Abalone (Table 4.4) and Diabetes models (Table 4.8) 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 difficult. Although both models have similar Certain Model Parity, the Boundary Parity for the diabetes does not converge easily.

The leaf count serves as an indication of the potential complexity in 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. This case is exemplified in Tables 4.6 and 4.7. In this instance, MTRE-PAN requires

Table 4.8. Evaluation metrics for the Diabetes interpretable models 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	0.00	0.00	1
2	87.41	0.00	0.00	2
3	74.61	53.57	0.00	4
4	81.52	53.57	0.00	7
5	79.88	53.57	0.00	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	58.00	275

Table 4.9. Evaluation metrics for the Diabetes interpretable models 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	0.00	0.00	1
2	70.27	0.00	0.00	2
3	74.45	43.13	0.00	4
4	82.79	43.13	0.00	7
5	80.49	43.13	0.00	13
6	81.48	52.33	0.00	25
7	84.40	60.00	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

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 between TRE-PAN and MTRE-PAN with similar Certain Model Parity, the leaf count doubles.

5. CONCLUSION

This thesis proposes MTRE-PAN, a global translation model that can be used for any target ML model. MTRE-PAN builds the explainable model as a hybrid combination of a decision tree and SVM linear separators at each node of the tree. This explainable model organizes the feature space into a set of polygons that approach the behavior of the target model.

MTRE-PAN was inspired by TRE-PAN which is also a global translation model that was previously proposed in the literature. TRE-PAN uses a decision tree to partition the space, instead of a linear hyperplane as in the case of the proposed MTRE-PAN. The performance of MTRE-PAN is compared to that of TRE-PAN for three target ML models which were developed using public domain datasets. The results show that MTRE-PAN achieves a higher parity across all metrics at shallower depths compared to TRE-PAN.

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 similar results to MTRE-PAN thereby offsetting the difference in computational complexity. This is especially true for highly non-linear target models.

Future work include the pruning of the MTRE-PAN interpretable tree model around the decision boundaries of the target model and potentially developing a loss function that is specific to each target model to replace the current SVM classifiers. This loss function could use the target model's probability distribution to locate the decision boundaries. However, this will be at the cost of a more model dependent methodology. An example of this technique applied to image classification is provided in [31]. Future work also includes developing more robust metrics that can help measure the level of explainability produced by each approach.

REFERENCES

[1] B. Goodman and S. Flaxman, "European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"," *AI Magazine*, vol. 38, no. 3, pp. 50–57, 3 Oct. 2, 2017, ISSN: 2371-9621. DOI: 10.1609/aimag.v38i3.2741. [Online]. Available: https://www.aaai.org/ojs/index.php/aimagazine/article/view/2741 (visited on 07/06/2020).

[2] (Oct. 12, 2016). The Administration's Report on the Future of Artificial Intelligence, whitehouse.gov, [Online]. Available: https://obamawhitehouse.archives.gov/blog/2016/10/12/administrations-report-future-artificial-intelligence (visited on 05/17/2021).

[3] E. M. Stefanik. (May 22, 2018). H.R.5356 - 115th Congress (2017-2018): National Security Commission Artificial Intelligence Act of 2018, [Online]. Available: https://www.congress.gov/bill/115th-congress/house-bill/5356 (visited on 05/17/2021).

[4] A. Chouldechova. (Oct. 24, 2016). Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. arXiv: 1610.07524 `[cs, stat]`, [Online]. Available: http://arxiv.org/abs/1610.07524 (visited on 07/07/2020).

[5] Supreme Court of Wisconsin, "State v. Loomis," *NW 2d*, vol. 881, p. 749, Argued April 5, 2016.

[6] K. Freeman, "Algorithmic Injustice: How the Wisconsin Supreme Court Failed to Protect Due Process Rights in State v. Loomis," vol. 18, p. 33,

[7] A. Ferguson, "Policing Predictive Policing," *Washington University Law Review*, vol. 94, no. 5, pp. 1109–1189, Jan. 1, 2017, ISSN: 2166-8000.

[8] R. Chatila and J. C. Havens, "The IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems," in *Robotics and Well-Being*, ser. Intelligent Systems, Control and Automation: Science and Engineering, M. I. Aldinhas Ferreira, J. Silva Sequeira, G. Singh Virk, M. O. Tokhi, and E. E. Kadar, Eds., vol. 95, Cham: Springer International Publishing, 2019, pp. 11–16, ISBN: 978-3-030-12523-3 978-3-030-12524-0.

[9] I. Ben-Gal, "Bayesian Networks," in *Encyclopedia of Statistics in Quality and Reliability*, American Cancer Society, 2008, ISBN: 978-0-470-06157-2. DOI: 10.1002/9780470061572.eqr089. [Online]. Available: http://onlinelibrary.wiley.com/doi/abs/10.1002/9780470061572.eqr089 (visited on 04/16/2020).

[10] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar. 1, 1986, ISSN: 1573-0565. DOI: 10.1007/BF00116251. [Online]. Available: https://doi.org/10.1007/BF00116251 (visited on 04/16/2020).

[11] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, Aug. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[12] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, Jan. 1, 2015, ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135 (visited on 04/16/2020).

[13] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995, ISSN: 0885-6125, 1573-0565. DOI: 10.1007/BF00994018. [Online]. Available: http://link.springer.com/10.1007/BF00994018 (visited on 03/26/2020).

[14] Y. S. Kim, "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 Systems with Applications*, vol. 34, no. 2, pp. 1227–1234, Feb. 1, 2008, ISSN: 0957-4174. DOI: 10.1016/j.eswa.2006.12.017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417406004118 (visited on 04/18/2020).

[15] Z. C. Lipton. (Jun. 10, 2016). The Mythos of Model Interpretability. arXiv: 1606.03490 `[cs, stat]`, [Online]. Available: http://arxiv.org/abs/1606.03490 (visited on 06/19/2019).

[16] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object Recognition with Gradient-Based Learning," in *Shape, Contour and Grouping in Computer Vision*, ser. Lecture Notes in Computer Science, D. A. Forsyth, J. L. Mundy, V. di Gesú, and R. Cipolla, Eds., Berlin, Heidelberg: Springer, 1999, pp. 319–345, ISBN: 978-3-540-46805-9.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 6088 Oct. 1986, ISSN: 1476-4687. DOI: 10.1038/323533a0. [Online]. Available: http://www.nature.com/articles/323533a0 (visited on 04/18/2020).

[18] M. T. Ribeiro, S. Singh, and C. Guestrin. (Feb. 16, 2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv: 1602.04938 `[cs, stat]`, [Online]. Available: http://arxiv.org/abs/1602.04938 (visited on 06/19/2019).

[19] M. W. Craven and J. W. Shavlik, "Extracting tree-structured representations of trained networks," in *Proceedings of the 8th International Conference on Neural Information Processing Systems*, ser. NIPS'95, Denver, Colorado: MIT Press, Nov. 27, 1995, pp. 24–30.

[20] M. Sato and H. Tsukimoto, "Rule extraction from neural networks via decision tree induction," in *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, vol. 3, Jul. 2001, 1870–1875 vol.3. DOI: 10.1109/IJCNN.2001.938448.

[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," in *International Conference on Machine Learning*, Jun. 11, 2016, pp. 1995–2003. [Online]. Available: http://proceedings.mlr.press/v48/wangf16.html (visited on 07/08/2020).

[22] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps," Dec. 20, 2013. [Online]. Available: https://arxiv.org/abs/1312.6034v2 (visited on 07/09/2020).

[23] P. Tabacof and E. Valle, "Exploring the space of adversarial images," in *2016 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2016, pp. 426–433. DOI: 10.1109/IJCNN.2016.7727230.

[24] K. Bennett and J. Blue, “A support vector machine approach to decision trees,” in *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*, vol. 3, May 1998, 2396–2401 vol.3. DOI: 10.1109/IJCNN.1998.687237.

[25] G. Madzarov and D. Gjorgjevikj, “Multi-class classification using support vector machines in decision tree architecture,” in *IEEE EUROCON 2009*, St.-Petersburg: IEEE, May 2009, pp. 288–295, ISBN: 978-1-4244-3860-0. DOI: 10.1109/EURCON.2009.5167645. [Online]. Available: https://ieeexplore.ieee.org/document/5167645/ (visited on 10/26/2021).

[26] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3 edition. Upper Saddle River: Pearson, Dec. 11, 2009, 1152 pp., ISBN: 978-0-13-604259-4.

[27] M. Craven, “Extracting Comprehensive Models From Trained Neural Networks,” Thesis, University Of Wisconsin - Madison, Sep. 1996, 212 pp.

[28] (Apr. 29, 2021). UCI Machine Learning Repository: Abalone Data Set, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/abalone (visited on 04/29/2021).

[29] (Apr. 29, 2021). UCI Machine Learning Repository: Census Income Data Set, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/census+income (visited on 04/29/2021).

[30] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes, “Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus,” *Proceedings of the Annual Symposium on Computer Application in Medical Care*, pp. 261–265, Nov. 9, 1988, ISSN: 0195-4210. pmid: null. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/ (visited on 10/09/2021).

[31] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. (Aug. 12, 2019). Adversarial Examples Are Not Bugs, They Are Features. arXiv: 1905.02175 `[cs, stat]`, [Online]. Available: http://arxiv.org/abs/1905.02175 (visited on 11/05/2021).