Extracting PAC Decision Trees from Black Box Binary Classifiers: The Gender Bias Study Case on BERT-based Language Models

Ana Ozaki[1,3], **Roberto Confalonieri**[2], **Ricardo Guimarães**[3], **Anders Imenes**[3]

[1]Universitetet i Oslo, Norway
[2]Università degli Studi di Padova, Italy
[3]Universitetet i Bergen, Norway
anaoz@uio.no, roberto.confalonieri@unipd.it

Abstract

Decision trees are a popular machine learning method, valued for their inherent explainability. In Explainable AI, decision trees serve as surrogate models for complex black box AI models or as approximations of parts of such models. A key challenge of this approach is assessing how accurately the extracted decision tree represents the original model and determining the extent to which it can be trusted as an approximation of its behavior. In this work, we investigate the use of the Probably Approximately Correct (PAC) framework to provide a theoretical guarantee of fidelity for decision trees extracted from AI models. Leveraging the theoretical foundations of the PAC framework, we adapt a decision tree algorithm to ensure a PAC guarantee under specific conditions. We focus on binary classification and conduct experiments where we extract decision trees from BERT-based language models with PAC guarantees. Our results indicate occupational gender bias in these models, which confirm previous results in the literature. Additionally, the decision tree format enhances the visualization of which occupations are most impacted by social bias.

1 Introduction

Decision trees are known to be easy to explain because, given an input, one can directly trace the decisions which led to the output. However, this method lacks the generalizability of architectures based on neural networks that have hidden layers. Although there has been some progress in increasing the generalizability of decision trees (Frosst and Hinton 2017) and even translating trained neural networks into decision trees (Aytekin 2022), there appears to be a trade-off between generalizability and interpretability that is not easy to balance.

To enjoy the best of both worlds, the AI research community has been exploring the idea of *extracting decision trees from trained neural networks*. The extracted trees can then serve as an explainable approximation of these black-box models (Confalonieri et al. 2020; Krishnan, Sivakumar, and Bhattacharya 1999; Vasilev, Mincheva, and Nikolov 2020; Boz 2002; Bologna and Hayashi 2018; Schmitz, Aldrich, and Gouws 1999; Dancey, McLean, and Bandar 2004; Burkhardt et al. 2021; Setiono and Liu 1995; Nanfack, Temple, and Frénay 2021; Craven and Shavlik 1995; II et al. 2011; Awudu and Zhou 2015) (see also (Chorowski and Zurada 2011; Shih, Darwiche, and Choi 2019) for works extracting decision diagrams from trained neural networks). Previous works in this direction indicate that, in practice, decision trees are indeed useful for understanding patterns in the data, which cannot be understood from black-box models. This approach has been applied to various domains such as medicine (Dancey, Bandar, and McLean 2010), traffic signal control (Zhu, Yin, and Chen 2023), risk credit evaluation (Baesens et al. 2003), water resource management (Wei, Chen, and Hsu 2012), multimedia (Fanta, Pulc, and Holena 2019), among others. The decision tree in this case functions as a *surrogate* for the original model. The advantage of surrogate models is their ability to abstract the original model, focusing on specific inputs or aspects of interest.

In this paper, we apply this approach to unveil occupational gender bias in BERT-based language models. By extracting rules that approximate *parts of* the model of interest, we can avoid the challenges associated with approximating the whole language model, which could be computationally infeasible or inefficient. Moreover, the approach of extracting decision trees can be applied to black-box models regardless of their internal architectures, which are constantly evolving as the field of artificial intelligence tackles increasingly complex tasks.

The main challenge with such approximations lies in determining *how accurately the extracted decision tree represents the relevant parts of the original black box model*. To the best of our knowledge, current practical approaches for extracting decision trees from trained neural networks do not provide theoretical guarantees regarding the fidelity of the decision trees with respect to the original model. What guarantees would be appropriate? How could such guarantees be given? In Computational Learning Theory, the most well studied framework which gives correctness guarantees is known as the Probably Approximately Correct (PAC) framework (Valiant 1984). In the original framework, a learner receives examples classified according to some *target* function. The classification is binary and the classified examples come according to a probability distribution that is fixed but arbitrary and unknown. The learner then creates a *hypothesis* consistent with the information received. In a nutshell, the theoretical correctness guarantee is that, provided with enough data, with high probability, the difference between

any hypothesis created by the learner that is consistent with the data and the target function is minimal. This "difference" is quantified by the probability of misclassification, known as the *true error* (Shalev-Shwartz and Ben-David 2014). The *sample complexity* quantifies the amount of data needed for such a guarantee (Vapnik and Chervonenkis 1971).

In this work, we study the PAC framework and explore its potential to provide a theoretical correctness guarantee for decision trees extracted from black-box models. Our contributions can be summarized as follows:

- We provide formal definitions for the relevant notions we use. In particular, we relate the notion of fidelity in the literature on decision trees with the PAC guarantee (Section 2). We focus on binary classification, as this is the most studied setting within the PAC framework.
- On the theoretical side, we provide a bound on the sample size needed for PAC learning when the training error is bounded by $k/|\mathcal{S}|$, where $|\mathcal{S}|$ is the size of the training set and $k \in \mathbb{N}$ (Theorem 8). This bound is used in a decision tree algorithm (Section 3).
- On the practical side (Section 4), we perform experiments based on a recent gender bias study case (Blum et al. 2023), where we extract decision trees using BERT-based models (Devlin et al. 2019; Liu et al. 2019).

We conclude and mention future work in Section 5.

2 The PAC Framework

Here we introduce the relevant notation for decision trees and PAC learning (Section 2.1). We then show in Section 2.2 our theoretical bound on the number of examples when the hypothesis is inconsistent with the training set.

2.1 Basic Definitions

We now formalise the PAC learning framework. We follow the notation for the definition of PAC learning adopted in the literature (Valiant 1984; Shalev-Shwartz and Ben-David 2014; Kearns and Mansour 1999; Ozaki 2020).

Definition 1. A *concept class* $\mathbb{C}$ is a triple $(\mathcal{E}, \mathcal{H}, \mu)$ where $\mathcal{H}$ is a set of concept representations[1], called *hypothesis space*; $\mathcal{E}$ is a set of examples; and μ is a function that maps each element of $\mathcal{H}$ to a subset of $\mathcal{E}$.

Each element of $\mathcal{H}$ is called a *hypothesis*. The *target representation* (here simply called *target*) is a fixed but arbitrary element of $\mathcal{H}$, representing the kind of knowledge that is aimed to be learned. Given a target $t \in \mathcal{H}$, we say that the examples in $\mu(t)$ are *positive examples*, otherwise, they are called *negative examples*. A *classified example* w.r.t. the target t is a pair $(e, \mathsf{c}(e))$, where $\mathsf{c}(e) = 1$ (positive) if $e \in \mu(t)$ and $\mathsf{c}(e) = 0$ (negative) otherwise. A *training set* is a set of classified examples (we may omit that this is w.r.t. t). Given a training set $\mathcal{S}$, we denote by $\mathcal{S}_e$ the examples in $\mathcal{S}$. In other words, $\mathcal{S}_e$ is the result of removing the labels in $\mathcal{S}$.

[1] In the Machine Learning Theory literature, a *concept* is often defined as a set of examples and a concept representation is a way of representing such set.

Definition 2 (Induced Probability Distribution)**.** Let m be a positive integer and let $\mathcal{D}$ be a probability distribution over a set of examples $\mathcal{E}$. The probability distribution $\mathcal{D}$ *induces* the probability distribution $\mathcal{D}^m$ over the set of all sequences $\mathcal{S}_{\mathbf{e}}$ of $\mathcal{E}$ with m elements, defined as

$$\mathcal{D}^m(\{\mathcal{S}_{\mathbf{e}}\}) := \prod_{e \in \mathcal{S}_{\mathbf{e}}} \mathcal{D}(\{e\}).$$

Definition 3 (Example Query)**.** Let $\mathbb{C} = (\mathcal{E}, \mathcal{H}, \mu)$ be a concept class. Given a target $t \in \mathcal{H}$, let $\mathsf{EX}^{\mathcal{D}}_{\mathbb{C},t}$ be the *oracle* that takes no input, and outputs a classified example $(e, \mathsf{c}(e))$, where $e \in \mathcal{E}$ is drawn according to the probability distribution $\mathcal{D}$ and $\mathsf{c}(e) = 1$, if $e \in \mu(t)$, and $\mathsf{c}(e) = 0$, otherwise. An *example query* is a call to the oracle $\mathsf{EX}^{\mathcal{D}}_{\mathbb{C},t}$. A sample generated by $\mathsf{EX}^{\mathcal{D}}_{\mathbb{C},t}$ is a sequence of classified examples (possibly with repetitions), independently and identically distributed according to $\mathcal{D}$, drawn by calling $\mathsf{EX}^{\mathcal{D}}_{\mathbb{C},t}$.

Definition 4 (PAC learnability)**.** A concept class $\mathbb{C}$ is *PAC-learnable* if there is a function $f : (0,1)^2 \to \mathbb{N}$ and a deterministic algorithm s.t. for every $\epsilon, \delta \in (0,1) \subseteq \mathbb{R}$, every probability distribution $\mathcal{D}$ on $\mathcal{E}$ and every target $t \in \mathcal{H}$, given a sample of size $m \geq f(\epsilon, \delta)$ generated by $\mathsf{EX}^{\mathcal{D}}_{\mathbb{C},t}$, the algorithm always halts and outputs $h \in \mathcal{H}$ such that with probability at least $1 - \delta$ over the choice of m examples (counting repetitions) in $\mathcal{E}$, we have that $\mathcal{D}(\mu(h) \oplus \mu(t)) \leq \epsilon$.

The above condition can also be written as $\mathcal{D}^m(\{\mathcal{S}_{\mathbf{e}} \mid \mathcal{D}(\mu(t) \oplus \mu(h)) \leq \epsilon\}) \geq (1 - \delta)$. Given a training set $\mathcal{S}$, let $h_{\mathcal{S}}$ be a fixed but arbitrary element of $\mathcal{H}$ such that, for all $(e, \mathsf{c}(e)) \in \mathcal{S}$, $\mathsf{c}(e) = 1$ iff $e \in \mu(h_{\mathcal{S}})$ (i.e., $h_{\mathcal{S}}$ and t agree on their classification of the elements of $\mathcal{S}_{\mathbf{e}}$). Since $t \in \mathcal{H}$ such $h_{\mathcal{S}}$ exists. The error of h is the probability that h misclassifies the examples drawn according to $\mathcal{D}$.

Definition 5 (True Error)**.** Given a hypothesis $h \in \mathcal{H}$, a target function t and a probability distribution $\mathcal{D}$. The true error of h w.r.t. t and $\mathcal{D}$ is defined as

$$\mathsf{error}(h, t, \mathcal{D}) := \mathcal{D}(\mu(h) \oplus \mu(t)).$$

Since the learner does not know what $\mathcal{D}$ and t are, the true error is not directly available to the learner. A useful notion of error that can be calculated for a hypothesis created by the learner is the *training error*, defined as follows.

Definition 6 (Training Error)**.** Given a hypothesis $h \in \mathcal{H}$ and a training set $\mathcal{S}$, the *training error* of h is defined as the ratio of misclassified examples w.r.t. $\mathcal{S}$, in symbols $\mathsf{error}_{\mathcal{S}}(h)$ is

$$\frac{|\{e \in \mathcal{S}_e \mid \mathsf{c}(e){=}1, e \notin \mu(h)\} \cup \{e \in \mathcal{S}_e \mid \mathsf{c}(e){=}0, e \in \mu(h)\}|}{|\mathcal{S}|}.$$

We extract information from the black box models by posing membership queries, defined next.

Definition 7 (Membership Query)**.** Let $\mathbb{C} = (\mathcal{E}, \mathcal{H}, \mu)$ be a concept class. Given a target $t \in \mathcal{H}$, let $\mathsf{MQ}_{\mathbb{C},t}$ be the *oracle* that takes as input an example e and outputs its classification $\mathsf{c}(e)$, where $e \in \mathcal{E}$ is drawn according to the probability distribution $\mathcal{D}$ and $\mathsf{c}(e) = 1$, if $e \in \mu(t)$, and $\mathsf{c}(e) = 0$, otherwise. A *membership query* is a call to the oracle $\mathsf{MQ}_{\mathbb{C},t}$.

One can create a training set using $\mathsf{MQ}_{\mathbb{C},t}$ by generating examples according to some probability distribution and asking $\mathsf{MQ}_{\mathbb{C},t}$ to classify the examples.

2.2 Training Error Tolerance and Fidelity

A common assumption in PAC learning is the *realizability assumption*, which basically says that the target belongs to the hypothesis space. When the target belongs to the hypothesis space, in theory we know there is a hypothesis that is consistent with the training data, that is, the training error is 0. This is a rather strong assumption in practice because the hypothesis is usually not fully consistent with the training set (Mohri, Rostamizadeh, and Talwalkar 2018, Section 2.3). On the other extreme, the notion of agnostic PAC learning completely removes the realizability assumption but comes with other challenges. Here we keep the realizability assumption but allow the hypothesis to be inconsistent with the training set. We prove in Theorem 8 that one can allow the hypothesis to misclassify k examples in a training set and give a bound on the minimal number of examples needed for PAC learnability based on k. Given a sample $\mathcal{S}$, let $h^k_{\mathcal{S}}$ be a fixed but arbitrary element of $\mathcal{H}$ such that $e \in \mu(t)$ iff $e \in \mu(h^k_{\mathcal{S}})$ does not hold for at most k elements e in $\mathcal{S}$.

Theorem 8 (PAC learnability with Training Error)**.** Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0,1)$ and $\epsilon > 0$ and let m be an integer that satisfies

$$m \geq \frac{\ln((|\mathcal{H}| \cdot (k+1))/(\delta \cdot \epsilon^k))}{\epsilon} + k.$$

Then, for any labeling function, t, for which the realizability assumption holds, and for any distribution, $\mathcal{D}$, with probability of at least $1-\delta$ over the choice of an i.i.d. sample $\mathcal{S}$ of size m, we have that for every Empirical Risk Minimization (ERM) hypothesis, $h^k_{\mathcal{S}}$, it holds that $\mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) \leq \epsilon$.

Proof. The following proof is a non-trivial adaptation of the proof presented by Shalev-Shwartz and Ben-David (2014). Let $\mathcal{H}$ be a family of boolean functions, t a boolean function, $\mathcal{D}$ a probability distribution, and error an error function. We define, for $\epsilon \in (0,1)$, the following family of hypotheses, called "bad hypotheses":

$$\mathcal{H}_\epsilon \coloneqq \{h \in \mathcal{H} \mid \mathsf{error}(h, t, \mathcal{D}) > \epsilon\}.$$

Recall that, given a sample $\mathcal{S}$, let $h^k_{\mathcal{S}}$ be a fixed but arbitrary element of $\mathcal{H}$ such that $e \in \mu(t)$ iff $e \in \mu(h^k_{\mathcal{S}})$ does not hold for at most k elements e in $\mathcal{S}$. By the realizability assumption, $h^k_{\mathcal{S}} \in \mathcal{H}$. In other words, we have that there is $h^k_{\mathcal{S}}$ such that $|\mathcal{S}| \cdot \mathsf{error}_{\mathcal{S}}(h^k_{\mathcal{S}}) \leq k$. Let

$$M \coloneqq \{\mathcal{S}_e \mid \exists h \in \mathcal{H}_\epsilon \text{ s.t. } \mathsf{error}_{\mathcal{S}}(h) \leq \frac{k}{|\mathcal{S}|}\}.$$

Since $\mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) = \mathcal{D}(\mu(h^k_{\mathcal{S}}) \oplus \mu(t))$ we know that $\mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) > \epsilon$ only if $h^k_{\mathcal{S}} \in \mathcal{H}_\epsilon$. Therefore, $\{\mathcal{S}_e \mid \mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) > \epsilon\} \subseteq M$. Hence, $\mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) > \epsilon\}) \leq \mathcal{D}^m(M)$. By the union bound,

$$\mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) > \epsilon\}) \leq \sum_{h \in \mathcal{H}_\epsilon} \mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}_{\mathcal{S}}(h) \leq \frac{k}{|\mathcal{S}|}\}). \quad (1)$$

We now bound each summand of the second line of Inequality 1. Let us fix a bad hypothesis $h \in \mathcal{H}_\epsilon$. By i.i.d.,

$$\mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}_{\mathcal{S}}(h) \leq \frac{k}{|\mathcal{S}|}\}) = \quad (2)$$

$$\sum_{j=0}^{k} (C^m_j \cdot \mathcal{D}(\mathcal{E} \backslash (\mu(h) \oplus \mu(t)))^{m-j} \mathcal{D}(\mathcal{E} \backslash (\mu(h) \oplus \mu(t)))^j)$$

where C^m_j is the number of ways to choose j elements from a set of m elements (considering these m elements to be positions in the sequence of length m, possibly with repetitions). Eq. 2 expands to the cases in which we have between $m-k$ and m examples where the classification of the hypothesis matches with the classification of the target. For each individual sampling of an element of the training set,

$$\mathcal{D}(\mathcal{E} \backslash (\mu(h) \oplus \mu(t)) = 1 - \mathsf{error}(h, t, \mathcal{D}) \leq 1 - \epsilon.$$

Using the inequality $1 - x \leq e^{-x}$, we obtain for all $h \in \mathcal{H}_\epsilon$:

$$\mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}_{\mathcal{S}}(h) \leq \frac{k}{|\mathcal{S}|}\}) = \sum_{j=0}^{k} (C^m_j \cdot (1-\epsilon)^{m-j} \epsilon^j) \leq \sum_{j=0}^{k} (C^m_j \cdot e^{-\epsilon(m-j)} \epsilon^j)$$

By (1), $\mathcal{D}^m(\{\mathcal{S}_e \mid \mathsf{error}(h^k_{\mathcal{S}}, t, \mathcal{D}) > \epsilon\}) \leq |\mathcal{H}_\epsilon| \cdot (k+1) \cdot e^{-\epsilon(m-k)} \cdot \epsilon^k \leq |\mathcal{H}| \cdot (k+1) \cdot e^{-\epsilon(m-k)} \cdot \epsilon^k \leq \delta$. Then,

$$m \geq \frac{\ln(\frac{|\mathcal{H}| \cdot (k+1)}{\delta \cdot \epsilon^k})}{\epsilon} + k.$$

This concludes the proof of the theorem. □

We now relate the PAC notion with the notion of fidelity, classically studied within the explainable AI literature.

Definition 9 (Fidelity and Probabilistic Fidelity)**.** The *fidelity* of a post-hoc explanation h w.r.t. a black box model t and a set of examples $\mathcal{E}$ is defined as:

$$\mathsf{Fid}^{\mathcal{E}}(h, t) = \frac{|\{e \in \mathcal{E} \mid e \in \mu(h) \text{ iff } e \in \mu(t)\}|}{|\mathcal{E}|}.$$

The *probabilistic fidelity* of a post-hoc explanation h w.r.t. a black box model t, a set of examples $\mathcal{E}$, and a probability distribution $\mathcal{D}$ is:

$$\mathsf{Fid}^{\mathcal{E}}_{\mathcal{D}}(h, t) = \mathcal{D}(\{e \in \mathcal{E} \mid e \in \mu(h) \text{ iff } e \in \mu(t)\}).$$

That is, the probabilistic fidelity is $1 - \mathcal{D}(\mu(h) \oplus \mu(t))$.

Corollary 10 (Probabilistic Fidelity)**.** Let $\mathcal{H}$ be a finite hypothesis class. Let $\delta \in (0,1)$, $\epsilon > 0$, and $m \in \mathbb{N}$. If

$$m \geq \frac{\ln((|\mathcal{H}| \cdot (k+1))/(\delta \cdot \epsilon^k))}{\epsilon} + k$$

then, for any distribution $\mathcal{D}$, and for any target function t for which the realizability assumption holds, with probability of at least $1-\delta$ over the choice of an i.i.d. sample $\mathcal{S}$ of size m, for every ERM hypothesis $h^k_{\mathcal{S}}$, $\mathsf{Fid}^{\mathcal{E}}_{\mathcal{D}}(h^k_{\mathcal{S}}, t) \geq 1-\epsilon$.

Fidelity is more relevant than accuracy when the goal is to approximate black box models (not ground truth).

3 PAC Decision Trees

We first provide basic notions for defining decision trees (Section 3.1). Then, we present a version of a decision tree algorithm, called TopDown, introduced by Kearns and Mansour (1999) (Section 3.2) and the TREPAC algorithm (Section 3.3) for building decision trees with PAC guarantees.

3.1 Decision Trees

We provide formal definitions for candidate splits, constraints, trees and decision trees.

Definition 11 (Candidate Split & Constraint)**.** Let $\mathcal{F}$ be a set of *features*. Each feature f is associated with a set $\mathsf{values}(f)$ of possible values. A *candidate split* is an expression of the form $f \circledcirc v$ where $f \in \mathcal{F}$, $\circledcirc \in \{=, <, \leq, \geq, >\}$, and $v \in \mathsf{values}(f)$. We define *constraints* by induction. Every candidate split is a constraint. If ϕ and ψ are constraints then $(\phi \vee \psi)$ and $(\phi \wedge \psi)$ are constraints. Moreover, if ϕ is a constraint then $\neg(\phi)$ is also a constraint. We may write $\overline{\phi}$ as a short hand for $\neg(\phi)$. When dealing with constraints, we may treat sets and the conjunction of its elements interchangeably (e.g. $a \wedge b$ as $\{a, b\}$ and vice-versa). Moreover, we use 0 as an abbreviation for a contradiction $\psi \wedge \neg(\psi)$, with ψ being a candidate split, and we write 1 for the negation of 0.

Definition 12 (Tree)**.** A (binary) *tree* is a set τ containing $\varnothing$ and finite binary sequences closed under prefix (i.e., if e.g. $110 \in \tau$ then $11 \in \tau$ and $1 \in \tau$). The elements of τ are called *nodes* and the node $\varnothing$ is called the *root* of τ. The parent of a (non-root) node $n \in \tau$ is the result of removing the last element of n. E.g., 11 is the parent of 110.

Definition 13 (Decision Tree)**.** A (binary) *decision tree* is a relation that maps each node in a tree τ to a (possibly empty) set of constraints. We refer to this relation as a set of pairs (n, ψ) where n is a node in τ and ψ is a constraint. The *class* of a node n, denoted $\mathsf{class}(n)$, is the last bit of the sequence n. E.g., if $n = 110$ then $\mathsf{class}(n) = 0$.

Whenever we speak of the nodes of a decision tree we mean the domain of the relation. The nodes of a decision tree can be *internal nodes* or *leaf nodes*. The former are nodes which are parents of other nodes, while the latter are the remaining nodes, that is, those without children. We denote by $\mathsf{size\_of}(T)$ the number of internal nodes of a decision tree T. Also, we write $\mathsf{leaves}(T)$ for the set of leaf nodes in T. We write $T(l, \psi)$ for the tree that results of transforming a leaf l in a tree T into an internal node (with two children) and associating the constraint ψ with this internal node. In other words, $T(l, \psi) := (T \backslash \{(l, \varnothing)\}) \cup \{(l, \psi), (l0, \varnothing), (l1, \varnothing)\}$.

Given a node n in a decision tree T, we write $\mathsf{constr}_T(n)$ for the set of constraints associated with n in T (that is $\{\psi \mid (n, \psi) \in T\}$). Also, we define a function constr^*_T that maps each node to all constraints needed to be satisfied to reach the node. We define such function inductively as follows. First we set $\mathsf{constr}^*_T(\varnothing) := \varnothing$. Now, we set $\mathsf{constr}^*_T(n0) := \overline{\mathsf{constr}_T(n)} \cup \mathsf{constr}^*_T(n)$ and $\mathsf{constr}^*_T(n1) := \mathsf{constr}_T(n) \cup \mathsf{constr}^*_T(n)$.

Definition 14 (Tabular Examples & Constraint Satisfaction)**.** A *tabular example* for a set of features $\mathcal{F}$ is a tuple $(v_1, \ldots, v_n)$ with each $v_i \in \mathsf{values}(f_i)$ being a value for a feature $f_i \in \mathcal{F}$ (assume a fixed but arbitrary order for the features in $\mathcal{F}$). Then, $(v_1, \ldots, v_n)$ *satisfies*

- a candidate split $f_i \circledcirc v$ if $v_i \circledcirc v$, with $\circledcirc \in \{=, <, \leq, \geq, >\}$;
- a constraint $\neg(\psi)$ if it does not satisfy the constraint ψ;
- a constraint $(\psi \wedge \psi)$ if it satisfies ψ and ψ;
- a constraint $(\psi \vee \psi)$ if it satisfies ψ or ψ.

Given a tabular example x and a constraint ψ, we may write $x \vDash \psi$ to express that x satisfies ψ. Moreover, we say that a decision tree T *classifies* a tabular example x as *positive*, written $T(x) = 1$, if there is a leaf node $n1$ such that $x \vDash \mathsf{constraints}_T(n1)$. Otherwise, T *classifies* the tabular example x as *negative*.

3.2 The TopDown Decision Tree Algorithm

We define the concept class of (binary) decision trees.

Definition 15. The *concept class* of (binary) decision trees $\mathbb{C}_{DT}$ is a triple $(\mathcal{E}, \mathcal{H}_{DT}, \mu)$ where $\mathcal{E}$ is a set of tabular examples (according to Definition 14), $\mathcal{H}_{DT}$ is a set of (binary) decision trees, and μ is a function that maps each element T of $\mathcal{H}_{DT}$ to $\bigcup_{n1 \in \mathsf{leaves}(T)} \{x \in \mathcal{E} \mid x \vDash \mathsf{constr}^*_T(n1)\}$.

Let T be a decision tree (Definition 13). For each $l \in \mathsf{leaves}(T)$, $\mathcal{D}(\{x \in \mathcal{E} | x \vDash \mathsf{constr}^*_T(l)\})$ is the probability that an example $x \in \mathcal{E}$ (drawn according to $\mathcal{D}$) reaches l in T. The *error of a leaf* l is given by the probability that l wrongly classifies x. *The error of a decision tree* T is the sum of the error of all the leaves in T. We write $T_\star$ for the *target decision tree* that represents the black box binary classifier we attempt to approximate.

Proposition 16. The true error $\mathsf{error}(T, T_\star, \mathcal{D})$ of a decision tree T is

$$\sum_{n0 \in \mathsf{leaves}(T)} \mathcal{D}(\{x | x \vDash \mathsf{constr}^*_T(n0)\} \cap \mu(T_\star)) + \sum_{n1 \in \mathsf{leaves}(T)} \mathcal{D}(\{x | x \vDash \mathsf{constr}^*_T(n1)\} \cap \overline{\mu(T_\star)}).$$

Proof. The proof is at Ozaki et al. (2024). □

The TopDown algorithm (Algorithm 1) starts with a root node and builds a decision tree by creating new leaf nodes at each iteration of the 'while loop'. These leaves may become parents at the next iteration, and so on. The 'for loop' iterates over the space of possible leaves and constraints. The idea is to find a leaf and constraint in a way that the splitting minimizes the error of the tree.

Algorithm 1 is only useful for theoretical purposes since in practice we do not have access to the function $\mathsf{error}(\cdot)$ and, furthermore, it would be computationally expensive to make an exhaustive search on the constraints, as done in Line 4. So in the next section we adapt a decision tree algorithm to use the PAC framework.

Algorithm 1: TopDown($\mathsf{error}(\cdot, T_\star, \mathcal{D})$, size_limit, Φ) .

```
1: T := {(∅, ∅)}
2: while size_of(T) < size_limit do
3:     Δ_best ← 0
4:     for each (l, ψ) ∈ leaves(T) × Φ do
5:         Δ ← error(T, T_⋆, D) − error(T(l, ψ), T_⋆, D)
6:         if Δ ≥ Δ_best then
7:             Δ_best ← Δ
8:             (l_best, ψ_best) ← (l, ψ)
9:         end if
10:    end for
11:    T ← T ∪ {(l_best, ψ_best)}
12: end while
13: Return (T)
```

3.3 The TREPAC Algorithm

TREPAC is a tree induction algorithm that extracts decision trees from binary classifiers, seen as oracles. The original motivation behind the development of TREPAC is to approximate a neural network by means of a symbolic structure that is more interpretable than a neural network classification model while giving PAC guarantees. This approach has also its roots in the context of a wider interest in knowledge extraction from neural networks (see Introduction). TREPAC differs from conventional inductive learning algorithms such as CART and C4.5 because it uses the PAC framework to estimate the amount of training data and internal nodes in the resulting decision tree. It uses a membership oracle to classify examples used for training. Here we consider the binary entropy as the splitting criterion, that is, $G(q) = -q\log(q) - (1-q)\log(1-q)$ and $q \in [0, 1]$. The *best (binary) split*, denoted best_split, is the candidate split with the lowest entropy w.r.t a set of samples $\mathbb{S}$.

The pseudo-code for TREPAC is shown in Algorithm 2. In more details, the TREPAC algorithm works as follows. It takes 5 inputs, namely, an oracle $\mathsf{MQ}_{\mathbb{C},T_\star}$ for creating the sample, the maximal number of internal nodes (called size_limit), an upper bound k on the number of misclassified examples, a set of candidate splits Φ, and the size of the training set (called training_size). At the beginning, the algorithm creates an empty node and pushes it to the queue (Line 3). The queue is used to decide which nodes should be evaluated (Line 6). The algorithm calls the oracle $\mathsf{MQ}_{\mathbb{C},T_\star}$ to create a training set (Line 4).

TREPAC stops the tree extraction process when one of three criteria is met: no more nodes need to be further expanded (the queue is empty), a predefined limit of the tree size is reached, or the training error is below a predefined bound (Line 5). In Line 10, by the "misclassification of a node ni" we mean the number of classified examples in $\mathbb{S}$ that reach n, but whose class differs from i.

Corollary 17. Let $\mathsf{MQ}_{\mathbb{C},T_\star}$ be the membership oracle for the concept class $\mathbb{C}$ and the target $T_\star$ and let Φ be the set of candidate splits induced by $\mathbb{C}$. Let T be the output of a run of TREPAC ($\mathsf{MQ}_{\mathbb{C},T_\star}$, size_limit, k, Φ, training_size) and let $\epsilon, \delta \in (0, 1)$. With probability greater than $1 - \delta$, we have that $\mathsf{error}(T, T_\star, \mathcal{D})$ is smaller than ϵ, where $\mathcal{D}$ is the probability distribution used to generate the examples given as input to the membership oracle $\mathsf{MQ}_{\mathbb{C},T_\star}$ if (i) training_size is as in Theorem 8 and the number of misclassified examples of T w.r.t. the training set is bounded by k; or (ii) size_limit (the number of internal nodes) and training_size are as in Theorem 1 by (Kearns and Mansour 1999) and $k = 0$.

Algorithm 2: TREPAC($\mathsf{MQ}_{\mathbb{C},T_\star}$, size_limit, k, Φ, training_size)

```
1: Queue Q ← ∅
2: Decision tree T ← {(∅, ∅)}
3: Push the root node ∅ into Q
4: Let S be a set created with training_size calls to MQ_{C,T_⋆}
5: while Q ≠ ∅, size_of(T) < size_limit, and
                                  error_S(T) > k/training_size
   do
6:     Pop n from Q
7:     Use S to find best_split ∈ Φ based on binary entropy
8:     T := (T \ {(n, ∅)}) ∪ {(n, best_split), (n0, ∅), (n1, ∅)}
9:     for i ∈ {0, 1} do
10:        if ni misclassification is > k/(size_limit + 1) then
11:            Push ni into Q
12:        end if
13:    end for
14: end while
15: Return (T)
```

Proof. This result is a consequence of Theorem 8 and in Theorem 1 by Kearns and Mansour (1999)). □

4 Experiments

We now describe our experiments using TREPAC. In order to showcase the performance of TREPAC as a tool for explaining the behaviour black box models, we consider the recent study case presented by Blum et al. (2023), where the authors extract rules indicating occupational gender biases in BERT-based language models. In the mentioned reference, the authors use an adaptation of Angluin's Horn algorithm to extract rules. The authors *do not* provide PAC guarantees for their results (although this could have been obtained in their work by running the algorithm without limiting the number of simulated equivalence queries). Here we extract decision trees and analyse the results in light of the PAC framework. Although BERT-based models (Devlin et al. 2019; Liu et al. 2019) are not binary classifiers, the study case focuses on pronoun prediction ('she' or 'he').[2] We describe the study case by Blum et al. (2023) (Section 4.1) and also our experimental results (Section 4.2).

4.1 The Occupational Gender Bias Study Case

In the study case by Blum et al. (2023), the authors consider 4 language models: BERT-base-cased, BERT-large-cased (Devlin et al. 2019), RoBERTa-base, and RoBERTa-large (Liu et al. 2019). These language models are trained to fill up a blank part (called 'mask') of a sentence with a

[2]The non-binary pronoun 'they' did not receive significant prediction values by the models, see Table 2 in (Blum et al. 2023).

c	n	$m : k = 0$	$m : k = 5$	$m : k = 10$	$m : k = 15$
0.04	3	132	187	235	282
0.06	6	209	263	311	358
0.08	10	265	319	367	414
0.1	18	329	384	432	479

Table 1: Sample size m calculated c.f. Theorem 8 with $\delta = 0.1$, $\epsilon = 0.2$, and at most k misclassified examples. Values are rounded and calculated with the size of the hypothesis space being the number of all decision trees with n internal nodes. n is estimated c.f. Theorem 1 (Kearns and Mansour 1999) with constant $c = [0.04, 0.06, 0.08, 0.1]$ and $\gamma = 0.5 - \epsilon$.

token. The authors of the study case use the following kinds of features to create sentences[3]:

- **birth period:** before 1875, between 1875 and 1925, between 1925 and 1951, between 1951 and 1970, after 1970;
- **location:** North America, Africa, Europe, Asia, South America, Oceania, Eurasia, Americas, Australia;
- **occupation:** nurse, fashion designer, dancer, footballer, industrialist, boxer, singer, violinist.

In total, this gives 22 features which can be used to create sentences with one value for each kind of feature. The sentences are of the form: `<mask> was born [birth period] in [location] and is a/an [occupation]`, where the task of a BERT-based model is to predict `<mask>`. For instance, one possible sentence in this template would be: `<mask> was born after 1970 in Africa and is a singer.` Since there are 5 birth periods, 9 locations, and 8 occupations in the study, one can form $5 \cdot 9 \cdot 8 = 360$ sentences using the template. Even though the model could fill the mask with any token, the authors indicate that the most likely completions are one of the two pronouns 'she' or 'he', which works as a binary classification task. Each sentence is mapped to a binary vector using a lookup table (see at Ozaki et al. (2024)). The binary vector is a one-hot encoding of the features. Our sentence in the template above would be mapped to the binary vector:

$$[0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]$$

If the BERT-based model predicts the pronoun 'she' then this would correspond to classifying the example with 0 and if it would predict 'he' then the classification would be 1.

4.2 Results

We now present our experimental results. The idea behind our experiments is to (i) check the *feasibility* of the study case (Blum et al. 2023) in our setting, which gives PAC guarantees, (ii) check whether we obtain results that are *consistent* with their results, and also (iii) check whether the *format* of decision trees provides additional useful information. Experiments were run on a MacBook Pro M2 16GB. Each experiment was run 10 times, we present average values.

[3]They also include 'unknown value' features, relevant to the Horn algorithm but unnecessary for our work.

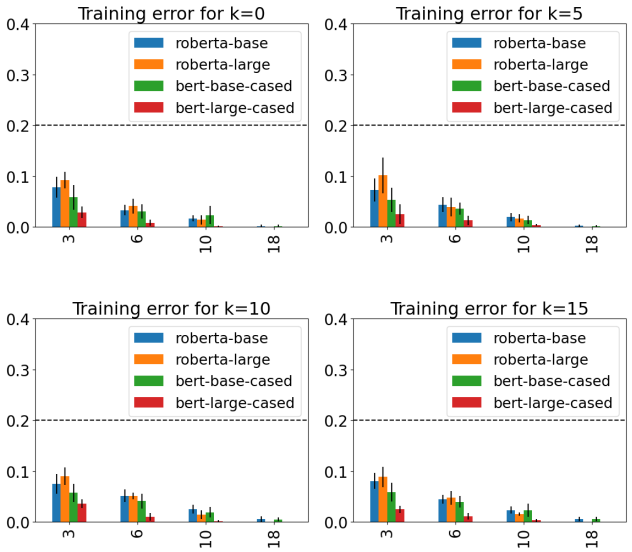

Figure 1: TREPAC training error for $k = [0, 5, 10, 15]$ and $n = [3, 6, 10, 18]$. Increasing the number of internal nodes reduces the training error. The horizontal dotted line correspond to $\epsilon = 0.2$. See also Table 1.

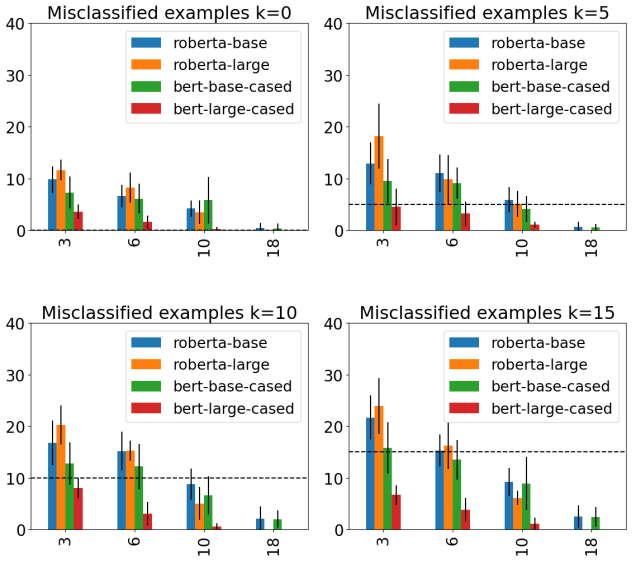

Figure 2: TREPAC misclassified training examples for $k = [0, 5, 10, 15]$ and $n = [3, 6, 10, 18]$. The number of misclassified examples is at most k (horizontal dotted line) when an appropriate tree size is chosen.

Feasibility We ran several experiments to check the practical feasibility of the approach using the PAC framework. The binary data representation is suitable for the case of learning surrogate decision trees from the BERT-based language models. We use Theorem 8 to calculate m considering the hypothesis space to be the set of all decision trees with

n internal nodes. The size of the hypothesis space was constrained to the number of all decision trees with n internal nodes, in particular $|n|^{|F|}$, where $|F|$ is number of features which in our case is 22. The corresponding sample sizes (m) for values of $k = [0, 5, 10, 15]$ and $n = [3, 6, 10, 18]$ are shown in Table 1. The range values for k and n were chosen so that the number of training examples is low in comparison with the total number of examples. When the number of accepted errors k is larger, the number of hypotheses which can be used increases. This also raises the chance of selecting a bad hypothesis. To compensate for this, a larger number of consistent examples is required. For each language model we extracted 16 decision trees varying n and k according to Table 1. For each decision tree we measured the training error (Figure 1), as well as the number of misclassified training examples (Figure 2).

As expected, both the training error and the number of misclassified training examples decrease whenever the size of the tree increases. Figure 1 shows the training error of the BERT-based language models. For all models, this error is less than the upper bound $\epsilon = 0.2$, chosen for determining the number of examples needed for PAC learnability based on k. This result is consistent with our Theorem 8. Figure 2 shows the sensitivity of the results w.r.t. the number of accepted errors k. The numbers of samples m calculated in Table 1 ensure that the number of misclassified training examples is at most $k = 0$ when the tree size is at least $n = 18$ for all models. For other values $k = [5, 10, 15]$, m ensures that the number of misclassified training examples is at most k when an appropriate tree size is chosen. To have at most $k = 5$ misclassified examples, the tree size needs to have at least $n = 10$ internal nodes. For higher values of k, trees with fewer nodes suffice. These results support the practical applicability of Theorem 8. In our experiments, the training errors of the decision trees extracted from RoBERTa-base and RoBERTa-large are higher than those from BERT-base and BERT-large. This suggests that models trained on larger corpora (which is the case for the RoBERTa models) may be more complex, which translates into more internal nodes. We provide additional supporting information regarding the error at Ozaki et al. (2024). Crucially, the time for running the algorithm is negligible. The time for generating training sets using BERT-based models varied from approximately 85 to 510 seconds (more at Ozaki et al. (2024)).

Consistency and Format Our results are consistent with those obtained by Blum et al. (2023), indicating occupational gender bias in these models (see Fig. 3). Regarding the format, the decision trees facilitate the visualization of which features are most relevant for pronoun prediction, with the occupations being the most relevant ones. The decision trees can also be used to extract rules. We report the frequency of common rules in Table 2. The 'nurse' occupation appears as the most biased occupation towards female.

5 Conclusion and Future Work

We motivate and study the PAC guarantees for decision trees extracted from black box binary classifiers. Since the training error is rarely zero, we provide a non-trivial proof for estimating the sample size for finite hypothesis when the training error is not zero. Our proof is applicable to any concept class with finite hypothesis space. We then formalize the relevant definitions for decision trees. Finally, we perform experimental results with decision trees on BERT-based models and analyse the results in light of the PAC framework.

In future work, it would be interesting to extend the experiments with other study cases and consider multi-classification tasks. Also, one could study a sampling strategy that satisfies the constraints of the leaves, as in Trepan (Craven and Shavlik 1995), within the PAC framework. This seems promising from a practical point of view. However, the analysis would be complex because randomly selecting examples that satisfy the leaves would change the sample space, necessitating an investigation into how the PAC framework could be adapted for this scenario.

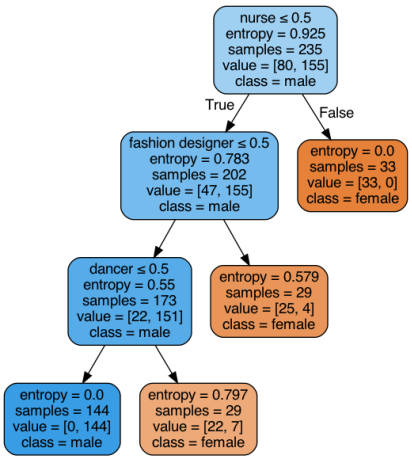

Figure 3: Surrogate tree for RoBERTa-base. The tree is extracted with $n = 3$ and $m = 235$ ($k = 10$), see Table 1.

Rule	Frequency
nurse → female	0.50
industrialist → male	0.18
footballer → male	0.16
boxer → male	0.12
fashion_designer → female	0.02
dancer → female	0.02

Table 2: Most frequent rules extracted from the surrogate decision trees of the BERT-based models.

Acknowledgments

We thank the anonymous reviewers for their comments. Ozaki is supported by the Research Council of Norway, project (316022). This work was also supported by the Research Council of Norway, Integreat - Norwegian Centre for knowledge-driven machine learning (332645). Confalonieri acknowledges funding support from the 'NeuroXAI' project (BIRD231830).

References

Awudu, K.; and Zhou, S. 2015. X-TREPAN: a multi class regression and adapted extraction of comprehensible decision tree in artificial neural networks. *ArXiv*, abs/1508.07551.

Aytekin, Ç. 2022. Neural Networks are Decision Trees. *CoRR*, abs/2210.05189.

Baesens, B.; Setiono, R.; Mues, C.; and Vanthienen, J. 2003. Using Neural Network Rule Extraction and Decision Tables for Credit - Risk Evaluation. *Manag. Sci.*, 49(3): 312–329.

Blum, S.; Koudijs, R.; Ozaki, A.; and Touileb, S. 2023. Learning Horn envelopes via queries from language models. *International Journal of Approximate Reasoning*, 109026.

Bologna, G.; and Hayashi, Y. 2018. A Comparison Study on Rule Extraction from Neural Network Ensembles, Boosted Shallow Trees, and SVMs. *Appl. Comput. Intell. Soft Comput.*, 2018: 4084850:1–4084850:20.

Boz, O. 2002. Extracting decision trees from trained neural networks. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, 456–461. ACM.

Burkhardt, S.; Brugger, J.; Wagner, N.; Ahmadi, Z.; Kersting, K.; and Kramer, S. 2021. Rule Extraction From Binary Neural Networks With Convolutional Rules for Model Validation. *Frontiers Artif. Intell.*, 4: 642263.

Chorowski, J.; and Zurada, J. M. 2011. Extracting Rules From Neural Networks as Decision Diagrams. *IEEE Trans. Neural Networks*, 22(12): 2435–2446.

Confalonieri, R.; Weyde, T.; Besold, T. R.; and del Prado Martín, F. M. 2020. Trepan Reloaded: A Knowledge-driven Approach to Explaining Black-box Models. In *Proceedings of ECAI 2020*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, 2457–2464. IOS press.

Craven, M. W.; and Shavlik, J. W. 1995. Extracting Tree-Structured Representations of Trained Networks. In Touretzky, D. S.; Mozer, M.; and Hasselmo, M. E., eds., *NeurIPS*, 24–30. MIT Press.

Dancey, D.; Bandar, Z.; and McLean, D. 2010. Rule extraction from neural networks for medical domains. In *International Joint Conference on Neural Networks, IJCNN 2010, Barcelona, Spain, 18-23 July, 2010*, 1–8. IEEE.

Dancey, D.; McLean, D.; and Bandar, Z. 2004. Decision Tree Extraction from Trained Neural Networks. In Barr, V.; and Markov, Z., eds., *Proceedings of the 17th Int. Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, 515–519. AAAI Press.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Burstein, J.; Doran, C.; and Solorio, T., eds., *NAACL-HLT*, 4171–4186. Association for Computational Linguistics.

Fanta, M.; Pulc, P.; and Holena, M. 2019. Rules Extraction from Neural Networks Trained on Multimedia Data. In Barancíková, P.; Holena, M.; Horváth, T.; Pleva, M.; and Rosa, R., eds., *ITAT*, volume 2473 of *CEUR Workshop Proceedings*, 26–35. CEUR-WS.org.

Frosst, N.; and Hinton, G. E. 2017. Distilling a Neural Network Into a Soft Decision Tree. In Besold, T. R.; and Kutz, O., eds., *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML*, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org.

II, W. A. Y.; Weckman, G. R.; Rangwala, M. H.; II, H. S. W.; Paschold, H. W.; Snow, A. H.; and Mourning, C. 2011. An investigation of TREPAN utilising a continuous oracle model. *Int. J. Data Anal. Tech. Strateg.*, 3(4): 325–352.

Kearns, M. J.; and Mansour, Y. 1999. On the Boosting Ability of Top-Down Decision Tree Learning Algorithms. *J. Comput. Syst. Sci.*, 58(1): 109–128.

Krishnan, R.; Sivakumar, G.; and Bhattacharya, P. 1999. Extracting decision trees from trained neural networks. *Pattern Recognit.*, 32(12): 1999–2009.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2018. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2 edition. ISBN 978-0-262-03940-6.

Nanfack, G.; Temple, P.; and Frénay, B. 2021. Global explanations with decision rules: a co-learning approach. In de Campos, C. P.; Maathuis, M. H.; and Quaeghebeur, E., eds., *UAI*, volume 161 of *Proceedings of Machine Learning Research*, 589–599. AUAI Press.

Ozaki, A. 2020. Learning Description Logic Ontologies: Five Approaches. Where Do They Stand? *Künstliche Intell.*, 34(3): 317–327.

Ozaki, A.; Confalonieri, R.; Guimarães, R.; and Imenes, A. 2024. Extracting PAC Decision Trees from Black Box Binary Classifiers: The Gender Bias Study Case on BERT-based Language Models. arXiv:2412.10513.

Schmitz, G. P. J.; Aldrich, C.; and Gouws, F. S. 1999. ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Trans. Neural Networks*, 10(6): 1392–1401.

Setiono, R.; and Liu, H. 1995. Understanding Neural Networks via Rule Extraction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95, Montréal Québec, Canada, August 20-25 1995, 2 Volumes*, 480–487. Morgan Kaufmann.

Shalev-Shwartz, S.; and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms*. USA: Cambridge University Press. ISBN 1107057132.

Shih, A.; Darwiche, A.; and Choi, A. 2019. Verifying Binarized Neural Networks by Angluin-Style Learning. In Janota, M.; and Lynce, I., eds., *Theory and Applications of Satisfiability Testing - SAT*, volume 11628 of *Lecture Notes in Computer Science*, 354–370. Springer.

Valiant, L. G. 1984. A Theory of the Learnable. In DeMillo, R. A., ed., *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, 436–445. ACM.

Vapnik, V. N.; and Chervonenkis, A. Y. 1971. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2): 264–280.

Vasilev, N.; Mincheva, Z.; and Nikolov, V. 2020. Decision Tree Extraction using Trained Neural Network. In Klein, C.; and Helfert, M., eds., *Proceedings of the 9th International Conference on Smart Cities and Green ICT Systems, SMARTGREENS 2020, Prague, Czech Republic, May 2-4, 2020*, 194–200. SCITEPRESS.

Wei, C.-C.; Chen, L.; and Hsu, H.-H. 2012. *Recent Advances in Computer Science and Information Engineering: Volume 1*, chapter Neural-Based Decision Trees Classification Techniques: A Case Study in Water Resources Management, 377–382. Berlin, Heidelberg: Springer Berlin Heidelberg.

Zhu, Y.; Yin, X.; and Chen, C. 2023. Extracting Decision Tree From Trained Deep Reinforcement Learning in Traffic Signal Control. *IEEE Trans. Comput. Soc. Syst.*, 10(4): 1997–2007.