XAudit : A Learning-Theoretic Look at Auditing with Explanations

Anonymous authors
Paper under double-blind review

Abstract

Responsible use of machine learning requires models to be audited for undesirable properties. While a body of work has proposed using explanations for auditing, how to do so and why has remained relatively ill-understood. This work formalizes the role of explanations in auditing using inspirations from active learning and investigates if and how model explanations can help audits. As an instantiation of our framework, we propose explanation-based algorithms for auditing linear classifiers and decision trees for 'feature sensitivity'. Our results illustrate that Counterfactual explanations are extremely helpful for auditing, even in the worst-case. While Anchor explanations and decision paths may not be as beneficial in the worst-case, in the average-case they do aid significantly.

1 Introduction

The recent success of machine learning (ML) has opened up exciting possibilities for potential societal applications Bojarski et al. (2016); Aini et al. (2020); Castiglioni et al. (2021). However, this kind of widespread usage requires us to be able to audit these models extensively and verify that they possess certain desirable properties related to safety, robustness and fairness.

So far, due to complexities in the ML pipeline and the black box nature of most models, auditing has been primarily performed in a somewhat ad-hoc manner, usually by creating a separate audit testing set. This, however, leaves open the question of what an audit really signifies. A recent body of work has also posited that local explanations might be helpful for auditing Bhatt et al. (2020); Oala et al. (2020); Adebayo and Gorelick (2017); Watson and Floridi (2021); Zhang et al. (2022); Poland (2022); Hamelers (2021); Carvalho et al. (2019); however, in the absence of a formal framework, how or why precisely this is the case is not understood.

In recent work, Yan and Zhang (2022) has made one of the first attempts at providing a formal framework for auditing and proposed principled auditors for auditing demographic parity with rigorous guarantees. However a limitation of their work is that they do not incorporate explanations into their framework. We take this line of work forward by formalizing the role of explanations in auditing.

In the context of auditing, the model-to-be-audited is held by a data scientist and not revealed to the auditor due to confidentiality reasons. This hidden aspect is what makes the task of auditing challenging. Consequently, the auditor's objective is to acquire insights into the concealed model. To achieve this, the auditor queries data points to the data scientist. In turn the data scientist responds with labels and (local) explanations. This paper aims to measure whether the supplementary information derived from explanations, beyond mere labels, can effectively diminish the query complexity of auditing.

As an example, we consider auditing feature sensitivity for two hypothesis classes with different local explanation methods – linear classifiers with counterfactual and anchor explanations and decision trees with decision path explanations. We provide auditing algorithms and theoretical guarantees for these cases. Our results illustrate that explanation methods differ greatly in their audit efficacy – while worst-case anchors do not provide any additional information than predictions themselves and with decision paths total queries

scale linearly with number of nodes in the tree in the worst-case, surprisingly counterfactuals greatly bring down the query complexity of auditing to a single query.

Could there be a more general auditing strategy in our framework? We next show that this is possible. By drawing on a connection between auditing and membership query active learning, we provide strategies for two critical aspects of any auditor – picking the next query and stopping criteria. We show that our general strategy leads to successful auditing when the property to be audited is *testable* – in the sense that it can be inferred based on making queries to the model.

We conclude by empirically evaluating our proposed auditors on standard datasets. Our experiments show that unlike the worst case, 'typical' anchors significantly reduce the number of queries needed to audit linear classifiers. Additionally, our proposed Anchor Augmentation technique helps reduce the query complexity over a no-anchor approach. Similarly, our experiments on decision trees demonstrate that the average number of queries to audit is considerably lower than the number of nodes in the tree.

To summarize, our contributions are as follows.

1. We formalize the role of explanations in auditing.
2. We propose a general strategy for auditing by drawing a connection between auditing and active learning.
3. We propose algorithms for using explanations to audit feature sensitivity in linear classifiers and decision trees.
4. We empirically illustrate that unlike the worst case, in the average case, explanations help reduce the query complexity of auditing significantly.
5. Finally, we discuss ways of dealing with an untruthful data scientist and the privacy concerns arising from auditing.

2 Preliminaries

The four key components involved in an ML audit are : 1) Data Scientist (DS), 2) Auditor, 3) Model to be audited and 4) Auditing Property (AP). DS is the one who holds a model which it cannot reveal for confidentiality reasons, while an Auditor is an external entity who wants to verify certain properties of the hidden model. We assume that the auditor has query access to this model through the DS. The hidden model, $\bar{h} : \mathcal{X} \times \{-1, 1\}$, belongs to a hypothesis class $\mathcal{H}$. We assume that the hypothesis class is known to the auditor. Let the instance space $\mathcal{X} = \mathcal{R}^d$.

The auditor will audit the model held by the DS for a specific auditing property that we measure quantitatively by a score function $s : \mathcal{H} \to [0, 1]$. This function has a value of zero when the property is absent in the model and a high value when the model exhibits the property to a large extent. Auditor knows the exact form of the score function but does not know its value for $\bar{h}$ (since $\bar{h}$ is hidden).

We expect an auditing property to satisfy the notion of 'testability', defined below. Testability ensures that the property of a hypothesis can be determined by making queries to instances $x \in \mathcal{X}$. Absence of testability makes it possible to have two hypotheses that agree on *every* input in the instance space, yet may have very different values of the property. Auditing this kind of a property would require more information than simply queries.

Definition 1 (Testable Auditing Property)**.** *An auditing property defined by score function $s : \mathcal{H} \to [0, 1]$ is testable using an instance space $\mathcal{X}$ if the following holds: for any $h_1, h_2 \in \mathcal{H}$ if $h_1(x) = h_2(x)$ $\forall x \in \mathcal{X}$, then $s(h_1) = s(h_2)$.*

When a query is asked to the DS, it returns the corresponding label and a local explanation. Both the DS and auditor agree upon a specific explanation method before auditing begins. Let $e_h : \mathcal{X} \to \mathcal{E}$ denote this explanation method where $\mathcal{E}$ represents the codomain of e_h and $h \in \mathcal{H}$. We assume that the DS is truthful, meaning the labels and explanations returned for x are $y = \bar{h}(x)$ and $e_{\bar{h}}(x)$ respectively. In §6 we discuss what happens when this assumption does not hold.

At the end of the auditing process, the auditor responds with an answer Y_a which takes values $\{\texttt{Yes}, \texttt{No}\}$. This is a random variable since both the DS and auditor can be randomized algorithms.

Next we conceptualize the auditing process as an interaction between the DS and auditor. Our protocol is as follows.

At each time step $t = 1, 2, \ldots$

1. Auditor picks a new query $x_t \in \mathcal{X}$ and supplies it to the DS.
2. DS returns a label $y_t \in \mathcal{Y}$ and an explanation $e_t \in \mathcal{E}$ to the auditor.
3. Auditor decides whether or not to stop. If auditor decides to stop, it returns a decision $Y_a \in \{\texttt{Yes}, \texttt{No}\}$, otherwise it continues to the next time step.

Furthermore, we formally define what makes a successful auditor at the end of the auditing process.

Definition 2 ((ϵ, δ)-auditor)**.** *An auditor is an (ϵ, δ)-auditor for $\epsilon, \delta \in [0, 1]$, hypothesis class $\mathcal{H}$ and a score function $s(\cdot)$ if $\forall h \in \mathcal{H}$ the following conditions hold : 1) if $s(h) > \epsilon$, $\Pr(Y_a = \texttt{Yes}) \geq 1 - \delta$ and 2) if $s(h) = 0$, $\Pr(Y_a = \texttt{No}) = 1$.*

The first condition pertaining to *soundness* implies that a successful auditor should return a `Yes` with high probability when AP is followed by the model to a large extent. The second condition measures *completeness*, and requires the auditor to say `No` when AP is not followed at all. Observe that when $0 < s(h) \leq \epsilon$, we cannot place a guarantee on the auditor's decision with a finite query budget.

Query complexity T of an (ϵ, δ)-auditor is the total number of queries it asks the DS before stopping. Efficient auditing requires that T be small.

3 An Example of Auditing Through 'Feature Sensitivity'

A popular auditing property in the literature has been demographic parity. But this notion applies exclusively to fairness and is distribution dependent. In contrast, we focus on 'feature sensitivity'.

A feature is termed sensitive if changing its value in the input leads to a different prediction. Identifying such features can lead to insights into a model's working and uncover spurious correlations and harmful biases that it may have learnt. For instance, while auditing a model that predicts the presence of lung cancer based on a radiology image, a human auditor may suspect a feature in the input, such as the presence of pen markings on X-rays, to be spuriously correlated to the output. Another example is when an auditor might suspect that an input feature such as home zipcode, which might act as a proxy for race (not included in the data), may be correlated with loan rejections. In both cases, the goal is to audit if a model is sensitive to a specific input feature. Note that our presented techniques can easily extend to multiple sensitive features by repeating the same procedure for different features.

Let the feature that we wish to audit the sensitivity of be called a feature-of-interest (FoI). Let *pair* p_{ij} denote a pair of inputs $x^i, x^j \in \mathcal{X}$ which are same in all but the feature-of-interest. The pair p_{ij} is called a *responsive* pair if the predictions for x^i and x^j are distinct. Unless mentioned, predictions are made by the model $\bar{h}$. Lastly, our feature-of-interest is a sensitive feature for the model if one or more responsive pairs exist.

The score function for feature sensitivity is the probability that a randomly drawn pair from the set of all pairs P is a responsive pair and is given as $s(\cdot) = \Pr_{p_{ij} \in P}(p_{ij} \text{ is a responsive pair})$. When P is finite, the score function can be interpreted as the fraction of responsive pairs. Usually when the score function is probabilistic, knowledge of the underlying distribution is required in order to estimate its value. However, in the following sections, we give an auditing decision without computing the value of score function explicitly, by using the structure of the class and explanation methods. We next prove that feature sensitivity is a testable property.

Theorem 3.1. *Feature Sensitivity, given by $s(\cdot) = \Pr_{p_{ij} \in P}(p_{ij}$ is a responsive pair$)$, is testable for a hypothesis class $\mathcal{H}$ using the instance space $\mathcal{X} = \mathcal{R}^d$.*

Proof. Fix two hypothesis $h_1, h_2 \in \mathcal{H}$ such that $\forall x \in \mathcal{X}$ it holds that $h_1(x) = h_2(x)$. Fix a pair $(x_i, x_j) \in \mathcal{X}$. This pair is responsive according to h_1 if and only if it is responsive according to h_2, i.e., $h_1(x_i) \neq h_1(x_j) \iff h_2(x_i) \neq h_2(x_j)$. Thus $s(h_1) = s(h_2)$ and it is testable for $\mathcal{H}$. □

A Simple Baseline: Random Testing. An easy way to detect feature sensitivity for *any* hypothesis class is to query a large number of pairs at random from the DS. The feature is sensitive if any responsive pair exists. This algorithm would need to query $\mathrm{O}(\frac{1}{\epsilon}\log\frac{1}{\delta})$ pairs to be an (ϵ, δ)-auditor.

Note that this baseline is independent of the hypothesis class. It does not need to find out what the hidden $\bar{h}$ is, even partially, in order to audit and it doesn't use explanations either. However, a lot of pairs have to be queried for a small ϵ. Next we discuss cases where using explanations and exploiting the structure of hypothesis class leads to more efficient auditing.

3.1 Linear Classifiers

Our first hypothesis class is *Linear Classifiers*, defined as $\mathcal{H}_{LC} = \{h_{w,b}\}_{w \in R^d, b \in R}$ where $h_{w,b}(x) = sign(\langle w, x\rangle + b)$. Responsive pairs exist with respect to $h \in \mathcal{H}_{LC}$ and for a feature of interest x_i if and only if weight w_i is non-zero.

3.1.1 Counterfactual Explanations

Given an input x and a model h, a counterfactual explanation Laugel et al. (2017); Deutch and Frost (2019); Karimi et al. (2020) returns the closest instance x' in L_2 distance such that h labels x' differently from x. More precisely, $x' = \mathrm{argmin}_{x':h(x')\neq h(x)} \|x - x'\|_2$.

We observe that for linear classifiers the difference $x - x'$ is parallel to w. This fact can be exploited by an auditor, which returns a decision by simply checking if $x - x'$ is zero in the feature-of-interest. We use these insights to design an auditor that uses a single random query to audit, denoted by $\texttt{AlgLC}_c$ 1.

Algorithm 1 $\texttt{AlgLC}_c$: Auditing Linear Classifiers using Counterfactuals

```
Query any point x from the DS
Auditor receives label y and explanation x' from the DS
ŵ ← x − x'
if ŵ_d = 0 then
    return No
else
    return Yes
end if
```

Theorem 3.2. *For any $\epsilon \in [0, 1]$, auditor $\texttt{AlgLC}_c$ is an $(\epsilon,0)$-auditor for feature sensitivity and hypothesis class $\mathcal{H}_{LC}$ with $T = 1$ query.*

Proof for the theorem can be found in the Appendix §A.2.

Counterfactual explanations enable the auditor to *partially* learn the hidden $\bar{h}$ via a scaled (not exact) version of the weights, which it then utilizes to make a decision. This partial-learning-based-auditing is way efficient than completely learning a linear classifier without explanations, which requires $\mathrm{O}(d\log\frac{1}{\epsilon})$ queries in the active learning setting §4. Also note that we allow the explanation to be found from the instance space rather than the training set. This is so because an example in the training data may be too far to be considered a good counterfactual explanation and if a counterfactual with some property (for instance an actionable counterfactual) is desired then the training data may not contain a relevant counterfactual example Karimi et al. (2020); Deutch and Frost (2019).

3.1.2 Anchor Explanations

Given a model h, an instance x, a distribution D, and a precision parameter τ, an anchor explanation Ribeiro et al. (2018) returns a hyperrectangle A_x such that (a) A_x contains x (b) at least τ fraction of the points in A_x under D have the same label $h(x)$ Ribeiro et al. (2018); Dasgupta et al. (2022). Specifically, $\tau = \Pr_{x' \in_{\mathcal{D}} A_x}(h(x) = h(x'))$. Here, the precision parameter τ measures the quality of the anchor explanation. To describe the quality of an anchor explanation, we also use a coverage parameter, c – which is the probability that a point sampled according to D lies in the A_x, $c = \Pr_{x' \sim D}(x' \in A_x)$. The distribution D is used by the DS to create anchor explanations.

We consider homogeneous linear classifiers ($b = 0$) in this section; however our techniques can be easily extended to non-homogeneous linear classifiers by considering $d + 1$ dimensions and concatenating 1 to x and b to w. For simplicity, we assume our anchor explanations have perfect precision.

We propose an auditor with anchors inspired by the active learning algorithm of Alabdulmohsin et al. (2015) which does not use explanations. Their goal is to learn a hypothesis by maintaining a search space over hypotheses and narrowing it down actively through label queries. They construct an ellipsoidal approximation of the search space at each step, $\varepsilon^\star = (\mu^\star, \Sigma^\star)$ where $\mu^\star$ is the center and $\Sigma^\star$ is the covariance matrix of the ellipsoid. Next they extract the top eigenvector of a matrix comprising of the covariance matrix of the ellipsoid. This eigenvector serves as the next synthesized query to the oracle. Our auditor uses this skeleton to audit by maintaining a similar search space and narrowing it down for $\bar{h}$.

Our contribution to the aforementioned algorithm is a new method for incorporating anchors into it – a procedure that we call *Anchor Augmentation* – for potentially higher auditing efficiency. The main idea is that anchor explanations give a region of space around a point x where labels are the same as that of x; using this fact more synthetic *already labeled* examples can be generated for free without actually querying the DS and fed into the algorithm. Through these insights we design an auditor $\texttt{AlgLC}_a$ §2 for linear classifiers using anchors. This auditor first learns the hidden $\bar{h}$ and then checks the weight of the feature of interest to return a decision.

Notation used in $\texttt{AlgLC}_a$ In $\texttt{AlgLC}_a$, $\varepsilon^{t\star} = (\mu^{t\star}, \Sigma^{t\star})$ denotes the largest ellipsoid that approximates the search space (corresponds to search space in our case) at time t where $\mu^{t\star}$ is the center and $\Sigma^{t\star}$ is the covariance matrix of the ellipsoid at time t. N^t is the orthonormal basis of the orthogonal complement of $\mu^{t\star}$ and $N^{t'}$ is its transpose. $\alpha^{t\star}$ is the top eigenvector of the matrix $N^{t'}\Sigma^{t\star}N^t$. In the implementation by Alabdulmohsin et al. (2015), some warm-up labeled points are supplied by the user, we denote this set as W. Let $W(t)$ denote the t-th element of this set. Let the dth feature be the feature of interest without loss of generality.

Worst Case Query Complexity. Anchor explanations returned by the DS can be absolutely consistent with the input and truthful, yet be worst-case. For instance, in the 1D case, where linear classifiers are thresholds, all the anchor points can lie on one side of the input point away from the threshold – this maintains truthfulness and consistency of the explanation – but do not help in identifying the threshold anymore than the input point itself. Generalizing this to higher dimensions, worst-case would mean that the shrinkage of the search space over hypotheses with and without explanations is the same. For the aforementioned algorithm, worst-case anchor explanations for a point x is $\{\lambda x | \lambda \in \mathcal{R}^+\}$ such that all anchor points lie on a ray in the direction of point x.

Lemma 1. *Given input x, a worst-case anchor for $\texttt{AlgLC}_a$ is of the form $A_x = \{\lambda x | \lambda \in \mathcal{R}^+\}$ with precision parameter $\tau = 1$.*

Proof. Consider that given x, DS returns label y and anchor explanation $A_x = \{\lambda x | \lambda \in \mathcal{R}^+\}$ with $\tau = 1$.

Let the set of all classifiers consistent with the label be $H = \{w : y\langle w, x\rangle > 0\}$.

From the definition of anchors and $\tau = 1$, the label of all points in anchor A_x is also y. Then, the set of all classifiers consistent with A_x is, $H' = \{w : y\langle w, \lambda x\rangle > 0\} = \{w : \lambda y\langle w, x\rangle > 0\} = H$. Hence the set of consistent classifiers remains the same despite anchors. Therefore A_x qualifies as a worst-case anchor. □

Algorithm 2 `AlgLC`$_a$: Auditing Linear Classifiers using Anchors

1: **Input:** T, augmentation size s, set of warm-up labeled points W
2: set of queried points $Q := \emptyset$, l :=size(W)
3: **for** $t = 1, 2, 3, \ldots, T + l$ **do**
4: **if** $t <= l$ **then**
5: $(x, y) := W(t)$
6: $Q \leftarrow Q \cup (x, y)$
7: goto step 15
8: **else**
9: Query point $x^t := N^t \alpha^{t\star}$ from the DS
10: Auditor receives label y and explanation A_x from the DS
11: $Q \leftarrow Q \cup (x^t, y)$
12: Sample randomly q points $x^{t1} \ldots x^{tq}$ from A_x } Anchor Augmentation
13: $Q \leftarrow Q \cup \{(x^{t1}, y) \ldots (x^{tq}, y)\}$
14: **end if**
15: $\varepsilon^{(t+1)*} = \left(\mu^{(t+1)\star}, \Sigma^{(t+1)\star}\right) :=$ estimate_ellipsoid(Q)
16: $N^{t+1} :=$ update_N($\mu^{(t+1)\star}$)
17: $\alpha^{(t+1)\star} :=$ update_alpha($N^{t+1}, \Sigma^{(t+1)\star}$)
18: **end for**
19: $\hat{w} = \mu^{T+1}$
20: **if** $|\hat{w}_d| \leq \Delta$ **then**
21: return `No`
22: **else**
23: return `Yes`
24: **end if**

The query complexity of our auditor using anchors `AlgLC`$_a$ is presented below.

Theorem 3.3. *For every dimension d, there exists $c > 0$ such that for any $\epsilon \in (0, 1)$, auditor* `AlgLC`$_a$ *is an $(\epsilon, 0)$-auditor for feature sensitivity and $\mathcal{H}_{LC}$ with $T = O\left(d \log \frac{2c}{\epsilon}\right)$ queries.*

This bound is comparable to actively learning a linear classifier without explanations $\mathrm{O}(d \log \frac{1}{\epsilon})$ Alabdulmohsin et al. (2015); Balcan and Long (2013) and might discredit the utility of explanations. However, we show the silver lining in §5.1. Proof and more details about the algorithm can be found in Appendix §A.4.

3.2 Decision Trees

Now we move beyond the linear hypothesis class to a non-linear one – decision trees and show how explanations can help in auditing them. A natural explanation for a decision tree prediction is the path traversed in the tree from root to the predicted leaf Audemard et al. (2021); Boer et al. (2020); Dasgupta et al. (2022).

We propose an auditor `AlgDT` based on an explanation-based breadth-first tree search algorithm, it works as follows. If the feature-of-interest is not present in the explanation path, auditor randomly picks a node in the path and perturbs its value while keeping the other features fixed such that it can explore the other branch of the perturbed node. Incase after perturbation the point satisfies one of the previous paths received as an explanation, it randomly picks a new query. Otherwise, the feature-of-interest is a node in the explanation, and therefore is a sensitive feature since it is used in the tree. `AlgDT` can be found in 3.

While we are not aware of an active learning algorithm that learns a decision tree with continuous features, Kushilevitz and Mansour (1991) propose a non-explanation membership query algorithm for binary features that exactly learns the tree in time poly(2^{depth}, d). Our auditor with explanations, which basically learns the entire tree in the worst-case, has a linear worst-case complexity on the order of number of nodes in the tree. Proof is in the App. Sec. A.5.

Algorithm 3 `AlgDT` : Auditing Decision Trees using decision path explanations

Given : Query Count threshold $qT \geq 1$
Initialize : Query count $qcount := 0$; set of received paths, $P := \emptyset$
Query any point x from the DS
Auditor receives label y and explanation path p from the DS
$P \leftarrow P \cup p$
$qcount := qcount + 1$
if foi is a node in p **then**
  return `Yes`
end if
while True **do**
  if $qcount == qT$ **then**
    return `No`
  end if
  $x :=$ perturb(x, p) {In App. Sec.A.5 }
  while findpath(x, P) {In App. Sec.A.5 } **do**
    Pick a random query as x
  end while
  Query x from the DS
  Auditor receives label y' and explanation path p' from the DS
  $P \leftarrow P \cup p'$
  $qcount := qcount + 1$
  if foi is a node in p' **then**
    return `Yes`
  end if
end while

Theorem 3.4. *For any $\epsilon \in [0, 1]$, auditor* `AlgDT` *is an (ϵ,0)-auditor for feature sensitivity and hypothesis class $\mathcal{H}_{DT}$ with $T = O(V)$ queries where V is the number of nodes in the decision tree.*

4 Auditor through connections with Active Learning

In the previous section we designed specific auditors based on hypothesis class and explanation type. This naturally begs the question if a general auditing strategy exists, we answer in the affirmative. We start by taking a closer look at two critical steps – strategy to pick the next query and the stopping condition – of our interactive auditing protocol mentioned in §2 and then propose a general recipe to create an auditor.

Table 1: Worst-Case Query Complexity of auditing feature sensitivity

Auditor	Query Complexity
Baseline	$O(\frac{1}{\epsilon}\log\frac{1}{\delta})$
`AlgLC`$_c$	1
`AlgLC`$_a$	$O(d\log(\frac{2C}{\epsilon}))$
`AlgDT`	$O(V)$

Picking next query The next query should be picked such that the auditor gets closer to the hidden $\bar{h}$, thereby reducing its uncertainty in making a decision. To achieve this the auditor can maintain a search space over hypotheses beginning with $\mathcal{S}_0 = \mathcal{H}$ and narrowing it through queries made to the DS at each subsequent step until the stopping condition is reached.

Let $Z_{t-1} \subseteq Z = \mathcal{X} \times \mathcal{Y}$ and $E_{t-1} \subseteq \mathcal{X} \times \mathcal{E}$ be the set of labeled examples and explanations acquired from the DS till $t-1$. Search space at t, $\mathcal{S}_t$, is the subset of hypotheses that are consistent with all the labels *and explanations* in Z_{t-1} and E_{t-1} respectively. As an example for counterfactual explanations, the hypothesis h is consistent if $\forall (x, \bar{h}(x)) \in Z_{t-1}$ and their counterfactuals $(x, x' = e_{\bar{h}}(x)) \in E_{t-1}$, $h(x) = \bar{h}(x)$ and $h(x') = \bar{h}(x')$.

For efficient pruning of the search space, the auditor chooses a query point that reduces the measure of the search space the most, no matter what explanation and label are returned by the DS. Formally, $x_t =$

$\text{argmax}_{x\in\mathcal{X}}(\min_{\mathcal{E}\times\mathcal{Y}} {}^{|\mathcal{S}_t|}/_{|\mathcal{S}_{t+1}|}) = \text{argmax}_{x\in\mathcal{X}}\ \text{value}_x$ where $|\mathcal{S}_t|$ and $|\mathcal{S}_{t+1}|$ denote the measure of the search spaces $\mathcal{S}_t$ and $\mathcal{S}_{t+1}$ and value_x corresponds to the reduction by worst case label and explanation for a point x. For finite hypothesis classes, the measure of the search space corresponds to the number of elements in it, while for infinite classes it is more complex; in §3.1.2 the search space is approximated by an ellipsoid at each time and measure is volume of the ellipsoid.

Stopping Condition decides query complexity of the auditor. For finite hypothesis classes, since $\bar{h}$ is always guaranteed to be in the search space at all times, if $s(h) > \epsilon$ for all h in S_t then the auditor can safely conclude that $\bar{h}$ has the desired property and stop with a `Yes` decision. Similarly, if $s(h) \leq \epsilon$ for all h in S_t the auditor will stop with a `No`. Observe that the auditor will ultimately arrive at one of these two cases – if none of the two stopping conditions hold, the next query will cause the search space to shrink – and ultimately our search space will consist of a single hypothesis.

For *infinite* hypothesis classes, we propose exploiting the structure of the hypothesis class and explanation methods to reduce the search space to a single hypothesis or to a set where value of the score function is same for each of its elements, in which case similar stopping conditions as the finite case can be used. Our proposed auditors for feature sensitivity lie in this setting.

Using the above query picking and stopping strategies, we formally outline an auditor for finite hypothesis classes presented in Alg.4 in the Appendix Sec.A.1. These strategies lead to an (ϵ, δ)-auditor, which follows directly from $\bar{h}$ belonging to the search space at all times and the stopping conditions as proved below.

Theorem 4.1. *Algorithm 4 is an (ϵ, δ)-auditor for a finite hypothesis class $\mathcal{H}$.*

Proof. Firstly, note that $\bar{h}$ is in the search space S_t at all times t. This is because the search space is reduced based on labels and explanations w.r.t. $\bar{h}$ (provided by the data scientist). Next, we assume that value of function s can be computed to arbitrary precision for any hypothesis h. Then the proposed stopping conditions satisfy the soundness and completeness properties of the (ϵ, δ)-auditor from definition. □

This general auditor is similar to the Optimal Deterministic Algorithm of Yan and Zhang (2022) in the sense that it is inspired from active learning. However, our general auditor has different stopping condition, also uses explanations and is not manipulation-proof (although all of our auditors in §3 are manipulation-proof). Due to difference in stopping condition and the fact that our auditor returns a binary decision, our auditor might be faster in reality. See Appendix §A.6 for details on manipulation-proofness.

Connections to Active Learning. For readers who are acquainted with active learning, our auditor may appear familiar. In active learning Dasgupta (2005), the goal is to learn a classifier in an interactive manner by querying highly informative unlabeled data points for labels. A specific variant of active learning is Membership Query Active Learning (MQAL) Angluin (1988); Feldman (2009) where the learner synthesizes queries rather than sampling from the data distribution or selecting from a pool.

Our auditor is essentially an active *partial*-learning algorithm in the MQAL setting, with a *modified* oracle that returns explanations in addition to labels. Here, the DS is the oracle and the auditor is the learner. The algorithm does *partial* learning – since it stops when the auditing goal is complete and before the classifier $\bar{h}$ is fully learnt.

We observe that learning is a harder task than auditing – since the goal in learning is to find the classifier generating the labels, while in auditing we aim to decide if this classifier has a certain property. This implies that if there exists an algorithm that can learn a classifier, it can also audit it as long as the score function can be computed/estimated, and a membership query active learning algorithm can be used as a fallback auditing algorithm. This leads to the following fallback guarantee.

Theorem 4.2. *If there exists a membership query active learner that can learn $\bar{h}$ exactly in T queries, then the active learner can also audit in T queries.*

Proof. Once $\bar{h}$ is known, $s(\bar{h})$ returns the auditing decision. If $s(\bar{h}) = 0$, the decision is `No` and if $s(\bar{h}) > \epsilon$, the decision is `Yes`. □

To summarize, while our auditor is connected to active learning, the key differences are – 1) the goals as mentioned above, 2) usage of explanations and 3) unlike active learning, partial learning of the hypothesis can be sufficient for auditing. For example, if all hypotheses in the search space have a zero score value, auditor can return a decision without learning $\bar{h}$ exactly; or learning only some parameters of $\bar{h}$ may be enough to audit. Since partial learning can be sufficient for auditing, an auditing algorithm is not necessarily an active learning algorithm.

5 Experiments

In this section we conduct experiments on standard datasets to test some aspects of feature sensitivity auditing. Specifically, we ask the following questions:

1. Does augmentation of 'typical' anchors reduce query complexity for linear classifiers?
2. How many queries are needed on average to audit decision trees of various depths?

The datasets used in our experiments are - Adult Income Becker and Kohavi (1996), Covertype Blackard (1998) and Credit Default Yeh (2016). The features of interest are gender, wilderness area type and sex for Adult, Covertype and Credit datasets respectively. All the datasets have a mix of categorical and continuous features. Categorical features are processed such that each category corresponds to a binary feature in itself. We remove all rows with missing values in any columns. We use an 80-20 split to create the train-test sets.

We learn both our linear and tree classifiers using scikit-learn. To run the anchor experiments, we use the matlab code provided by Alabdulmohsin et al. (2015). All of our experiments on a CPU varying from a minute to 2-3 hours. Additional details about the experiments can be found in the appendix §A.7.

5.1 Anchor Augmentation of Typical Anchors

As discussed in §3.1.2, augmentation of worst-case anchors does not help in reducing the query complexity of auditing. In other words, augmenting worst-case anchors is equivalent to not using anchor explanations. A natural question then is - do typical anchors help? Since auditing with anchors using $\texttt{AlgLC}_a$ means learning the DS's model under the hood, reduction in query complexity of learning implies reduction in that of auditing. Hence, we check experimentally if faster learning is achieved through anchor augmentation of typical anchor points.

Methodology We learn a linear classifier with weights w for each dataset; these correspond to the DS's model. Then the weights are estimated during auditing using Algorithm 2. We consider two different augmentations 1) worst-case anchor points which is equivalent to not using anchors 2) typical anchor points. We set the augmentation size (number of anchor points augmented) to a maximum of 30. Our anchors are hyperrectangles of a fixed volume surrounding the query point. We sample points with the same label as the query point from this hyperrectangle and augment to the set of queries in the typical case.

Results The results are shown in Figure 1. We see that the anchor augmentation of typical anchors drastically reduces the query complexity to achieve the same estimation error between weights as compared to not using anchors (or equivalently using worst-case anchors). This saving directly translates to efficiency in auditing. For example, in the adults dataset, less than 50% of the queries are needed to achieve a lower error with typical anchors than without them. This illustrates that anchor explanations can be helpful for auditing, suggesting an application for these explanations.

5.2 Average Queries for Decision Tree Auditing

In §3.2 we discussed that the worst-case query complexity of auditing is on the order of number of nodes in the decision tree. However on average the auditor might ask way lesser queries than the worst-case bound. We check experimentally if this is the case.

Methodology We learn decision trees for each of the datasets using scikit-learn which implements CART Breiman (2017) to construct the tree. We vary tree depth by fixing the 'max-depth' hyperparameter. Then we freeze the tree and run `AlgDT` a 1000 times. We report an average of the total queries required to audit

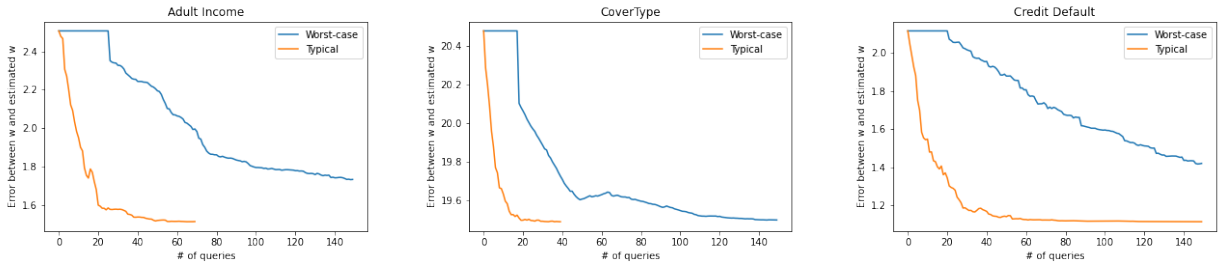

Figure 1: Number of queries required to learn a linear classifier with anchor augmentation of worst-case anchor points (in blue) and typical anchor points (in orange). For the latter, a clear reduction in query complexity is observed.

across the 1000 runs. We also run our random testing baseline from §3 until we detect the sensitive feature. We do this a 1000 times and report the average number of queries over those runs.

Results Our results are displayed in Table 2. As can be seen, the average number of queries needed to audit is consistently way lesser than the number of nodes, which highlights the utility of explanations in the average-case. There is a complex relationship between the total number of nodes in the tree and the number of feature-of-interest nodes in the tree which jointly determines the number of queries.

As the depth of the tree increases, the number of nodes increase exponentially while the feature-of-interest nodes and therefore average queries increase at a much slower pace. The CovType dataset has the lowest query complexity which might be a result of the feature-of-interest having four categories rather than two (male/female) in other datasets. There is a huge disparity between the average number of queries our auditor needs (low) and the average random testing queries (high). This demonstrates the efficacy of our auditor over the baseline.

Table 2: Decision Tree Auditing : Avg. number of queries required to audit over 1000 runs of `AlgDT` across different depths of decision trees. #Avg. queries are rounded to the nearest integer and reported with 95% confidence intervals. #Avg. queries are consistently lower than #Nodes and #Random Testing Queries.

Dataset	Depth	Test Acc.	#Avg. Queries	#Nodes	#FoI Nodes	#Random Testing Queries
Adult	9	85.09	11 ± 0.48	188	3	248 ± 15.4
	12	85.36	9 ± 0.4	552	18	299 ± 18.65
	15	84.59	10 ± 0.41	1158	31	160 ± 9.8
CovType	7	77.92	2 ± 0.10	119	9	17 ± 1.03
	9	80.06	2 ± 0.07	383	14	13 ± 0.77
	12	83.80	2 ± 0.08	1625	34	11 ± 0.67
	15	87.40	2 ± 0.08	4196	57	9 ± 0.5
Credit	9	81.07	17 ± 0.63	234	3	176 ± 10.71
	12	80.58	36 ± 1.2	629	6	276 ± 17.59
	15	81.65	54 ± 1.87	1030	20	905 ± 54.31

To summarize, we observe that worst-case estimates are generally very pessimistic and project explanations in a bad light. In the average case, which happens a lot on average, explanations bring down the number of queries for auditing significantly.

6 Discussion

Untruthful Data Scientist A natural question to ask is what happens when a DS is not entirely truthful in the auditing process, as we assumed in §2. What kind of auditing is possible in this case?

Suppose the DS returns all labels and explanations from an entirely different model than $\bar{h}$. In this case, there is no way for an auditor to detect it; but perhaps this can be dealt with in a procedural manner – like the DS hands its model to a trusted third party who answers the auditor's queries.

What if the DS returns the correct labels but incorrect explanations? DS might be forced to return correct labels when the auditor has some labeled samples already and hence can catch the DS if it lies with labels. This scenario motivates verification of explanations. Next, we discuss schemes to verify anchors and counterfactuals for this scenario.

Anchors. For anchors, verification is possible if we have samples from the underlying distribution D. For a query x, DS returns an unverified anchor A_x with precision τ_x and coverage c_x. The correctness of τ_x can be detected as follows. First, get an estimate for the true value of the precision parameter by sampling n points from anchor A_x according to D. Let the true and estimated values of precision parameter be τ and $\hat{\tau}$ respectively. $\hat{\tau}$ approaches τ with sufficiently large number of points as mentioned in lemma 2. Second, compare τ_x with $\hat{\tau}$. A large difference between τ_x and $\hat{\tau}$ implies false anchors.

Lemma 2. *For any $\Delta > 0$ and integer n,* $\Pr(|\tau - \hat{\tau}| \geq \Delta) \leq 2\exp^{-2\Delta^2 n}$.

Lemma 2 is immediate from Hoeffding's Inequality. Notice that the number of samples n required to verify precision changes with $1/\Delta^2$. If the fraction of responsive pairs ϵ equals Δ, our baseline in section 3 can audit with $O(1/\Delta)$ samples without using explanations. Hence, in adversarial conditions where the probability of a lying DS is high, it is better to audit with our explanations-free baseline than auditing with anchors and verifying them.

Since coverage is the probability that a point sampled from D belongs to A_x, it can be easily checked 1) by calculating the volume of A_x when all dimensions of x are bounded or 2) by sampling points from D when the features are unbounded.

Counterfactuals. Given x, let x' be the unverified counterfactual explanation returned by the DS. There are two aspects to a counterfactual explanation – its label which should be different from x and it should be the closest such point to x.

The first aspect can be easily verified by querying x' from the DS. For the second aspect, we observe that finding the counterfactual is equivalent to finding the closest adversarial point. Deriving from adversarial robustness literature, verifying the closeness aspect can be a computationally hard problem for some hypothesis classes like discussed in Weng et al. (2018). However, we propose a sampling based algorithm to estimate the true counterfactual, assuming that the DS is lying. Firstly, sample points from the ball $B(x, d(x, x'))$. For x' to be the true counterfactual, all points within the ball $B(x, d(x, x'))$ should have the same label as x. If a point x'' with a different label is sampled, select this point as an estimate of the true counterfactual and repeat the scheme with the new ball $B(x, d(x, x''))$. By following this procedure iteratively, we get closer to the correct counterfactual explanation as the radius of the ball reduces at each iteration. There are cases where our algorithm may not work well. We leave designing better algorithms for verifying closeness to future work.

Upon verification, if auditor finds that the DS is untruthful, it can choose to 1) stop auditing and declare that the DS is lying, 2) audit with estimated explanations or 3) audit with explanations-free baseline algorithm (§3) since option 2 is computationally intensive.

Privacy Auditing also raises a legitimate privacy concern. On one hand, model is hidden from the auditor for confidentiality reasons while on the other hand, to be able to give a correct auditing decision efficiently, the auditor has to extract/learn the hidden model, albeit partially. This concern can be resolved by using crytographic tools like Zero-Knowledge proofs Singh et al. (2021); Liu et al. (2021). A future direction of our work is modifying our framework to work with cryptographic tools.

7 Related Work

Auditing has been used to uncover undesirable behavior in ML models in the past Buolamwini and Gebru (2018); Koenecke et al. (2020); Tatman and Kasten (2017); Tatman (2017). However, these works were based

on creating a audit dataset in an ad-hoc manner. Recently, there have been efforts towards streamlining and developing a structured process for auditing. For instance, Gebru et al. (2018); Mitchell et al. (2019) introduce datasheets for datasets and model cards and Raji et al. (2020) introduce an end-to-end internal auditing process. In contrast with these works, we theoretically formalize auditing with explanations which allows us to determine what kind of properties can be audited and with how many queries.

Auditing with Formal Guarantees The work most related to ours is concurrent work due to Yan and Zhang (2022), who present a formal framework as well as algorithms for auditing fairness in ML models by checking if a model has a certain demographic parity on a data distribution. Their algorithms are motivated by connections to active learning as well as machine teaching Goldman and Kearns (1995). In contrast, we look at auditing *with explanations*. Our instantiation – auditing feature sensitivity – is different from demographic parity, which, together with the use of explanations, leads to very different algorithms.

Another work on auditing with guarantees is Goldwasser et al. (2021), who audit the accuracy of ML models. They use an interactive protocol between the verifier and the prover like our work, and their algorithm is related to a notion of property testing due to Balcan et al. (2011). The auditing task used in our paper, feature sensitivity, is different from accuracy used in their paper. We also utilize explanations.

Auditing and Explanations Explanations are supposed to help in auditing as suggested in Bhatt et al. (2020); Oala et al. (2020); Adebayo and Gorelick (2017); Watson and Floridi (2021); Zhang et al. (2022); Poland (2022); Hamelers (2021); Carvalho et al. (2019), but exactly how and under what conditions was not understood. In a concurrent work by Mougan et al. (2023), authors show that explanations lead to more sensitive audits than using labels and demonstrate the case for demographic parity using shapley values Lundberg et al. (2019; 2020). On the other hand, our focus is to demonstrate how explanations can reduce the query complexity of auditing and formalizing the auditing process by connecting it to active learning. The notion of feature sensitivity and the kind of explanations used in our paper are also different from theirs.

Other Auditing Methods Specific statistical methods have been proposed by Jagielski et al. (2020); Ye et al. (2021) to audit privacy and by Liu and Tsaftaris (2020); Huang et al. (2021) to audit data deletion. These are properties of the *learning algorithm* rather than the *model*. In our paper we focus on the latter and hence our algorithms are not applicable to auditing privacy and data deletion. Extending our framework to broader settings is an important direction of future work.

8 Conclusions and Future Work

We formalize the role of explanations in auditing and find their theoretical efficacy in reducing query complexity to vastly vary across different methods. Structure of the hypothesis class and explanation method together determine the efficacy. While worst-case complexities are pessimistic, our experiments herald optimism – in the average-case we see a significant reduction in the number of queries required to audit across all explanation types.

We believe that this work is a first step towards understanding certified auditing with explanations. A major research direction is auditing with approximate or incorrect explanations. Other future directions include incorporating other kinds of explanations, investigating different hypothesis classes, auditing properties and theorising the average-case complexity with explanations.

Ethics Statement. This work uses feature sensitivity more broadly than fairness (as described in §3)and only to demonstrate an instantiation of our framework. We agree that feature sensitivity, though linked to fairness, is not 'the' perfect measure of fairness and we doubt if such a metric can exist.

We believe that auditing is a complex process. Auditing can only be done for vulnerabilities that we know can exist and hence unknown vulnerabilities might still exist in a model despite being audited. There are real-life scenarios like limited budget (financial, time) which might prevent an auditor from *uncovering* vulnerabilites, even though they exist. Our work is a step in understanding the auditing process better.

References

Julius Adebayo and Micha Gorelick. Fairml: Auditing black-box predictive models. *Blog. Fastforwardlabs. Com*, 2017.

Gulimila Aini et al. A summary of the research on the judicial application of artificial intelligence. *Chinese Studies*, 9(01):14, 2020.

Ibrahim Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Efficient active learning of halfspaces via query synthesis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

Gilles Audemard, Steve Bellart, Louenas Bounia, Frédéric Koriche, Jean-Marie Lagniez, and Pierre Marquis. On the explanatory power of decision trees. *arXiv preprint arXiv:2108.05266*, 2021.

Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316. PMLR, 2013.

Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. *arXiv preprint arXiv:1111.0897*, 2011.

Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 648–657, 2020.

Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

Naama Boer, Daniel Deutch, Nave Frost, and Tova Milo. Personal insights for altering decisions of tree-based ensembles over time. *Proceedings of the VLDB Endowment*, 13(6):798–811, 2020.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

Diogo V Carvalho, Eduardo M Pereira, and Jaime S Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832, 2019.

Isabella Castiglioni, Leonardo Rundo, Marina Codari, Giovanni Di Leo, Christian Salvatore, Matteo Interlenghi, Francesca Gallivanone, Andrea Cozzi, Natascha Claudia D'Amico, and Francesco Sardanelli. Ai applications to medical images: From machine learning to deep learning. *Physica Medica*, 83:9–24, 2021.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. *Advances in neural information processing systems*, 18, 2005.

Sanjoy Dasgupta, Nave Frost, and Michal Moshkovitz. Framework for evaluating faithfulness of local explanations. *arXiv preprint arXiv:2202.00734*, 2022.

Daniel Deutch and Nave Frost. Constraints-based explanations of classifications. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 530–541. IEEE, 2019.

Vitaly Feldman. On the power of membership queries in agnostic learning. *The Journal of Machine Learning Research*, 10:163–182, 2009.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *arXiv preprint arXiv:1803.09010*, 2018.

Sally A Goldman and Michael J Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

Shafi Goldwasser, Guy N Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

LH Hamelers. Detecting and explaining potential financial fraud cases in invoice data with machine learning. Master's thesis, University of Twente, 2021.

Yangsibo Huang, Xiaoxiao Li, and Kai Li. Ema: Auditing data removal from trained models. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 793–803. Springer, 2021.

Matthew Jagielski, Jonathan Ullman, and Alina Oprea. Auditing differentially private machine learning: How private is private sgd? *Advances in Neural Information Processing Systems*, 33:22205–22216, 2020.

Amir-Hossein Karimi, Gilles Barthe, Borja Balle, and Isabel Valera. Model-agnostic counterfactual explanations for consequential decisions. In *International Conference on Artificial Intelligence and Statistics*, pages 895–905. PMLR, 2020.

Allison Koenecke, Andrew Nam, Emily Lake, Joe Nudell, Minnie Quartey, Zion Mengesha, Connor Toups, John R Rickford, Dan Jurafsky, and Sharad Goel. Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, 117(14):7684–7689, 2020.

Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the fourier spectrum. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 455–464, 1991.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. Inverse classification for comparison-based interpretability in machine learning. *arXiv preprint arXiv:1712.08443*, 2017.

Tianyi Liu, Xiang Xie, and Yupeng Zhang. Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

Xiao Liu and Sotirios A Tsaftaris. Have you forgotten? a method to assess if machine learning models have forgotten data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 95–105. Springer, 2020.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. Explainable ai for trees: From local explanations to global understanding. *arXiv preprint arXiv:1905.04610*, 2019.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

Carlos Mougan, Laura State, Antonio Ferrara, Salvatore Ruggieri, and Steffen Staab. Demographic parity inspector: Fairness audits via the explanation space. *arXiv preprint arXiv:2303.08040*, 2023.

Luis Oala, Jana Fehr, Luca Gilli, Pradeep Balachandran, Alixandro Werneck Leite, Saul Calderon-Ramirez, Danny Xie Li, Gabriel Nobis, Erick Alejandro Muñoz Alvarado, Giovanna Jaramillo-Gutierrez, et al. Ml4h auditing: From paper to practice. In *Machine learning for health*, pages 280–317. PMLR, 2020.

Cherie M Poland. The right tool for the job: Open-source auditing tools in machine learning. *arXiv preprint arXiv:2206.10613*, 2022.

Inioluwa Deborah Raji, Andrew Smart, Rebecca N White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the ai accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 33–44, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Nitin Singh, Pankaj Dayama, and Vinayaka Pandit. Zero knowledge proofs towards verifiable decentralized ai pipelines. *Cryptology ePrint Archive*, 2021.

Rachael Tatman. Gender and dialect bias in youtube's automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, 2017.

Rachael Tatman and Conner Kasten. Effects of talker dialect, gender & race on accuracy of bing speech and youtube automatic captions. In *Interspeech*, pages 934–938, 2017.

David S Watson and Luciano Floridi. The explanation game: a formal framework for interpretable machine learning. In *Ethics, Governance, and Policies in Artificial Intelligence*, pages 185–219. Springer, 2021.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.

Tom Yan and Chicheng Zhang. Active fairness auditing. In *International Conference on Machine Learning*, pages 24929–24962. PMLR, 2022.

Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, and Reza Shokri. Privacy auditing of machine learning using membership inference attacks. ″″, 2021.

I-Cheng Yeh. default of credit card clients. UCI Machine Learning Repository, 2016. DOI: https://doi.org/10.24432/C55S3H.

Chanyuan Abigail Zhang, Soohyun Cho, and Miklos Vasarhelyi. Explainable artificial intelligence (xai) in auditing. *International Journal of Accounting Information Systems*, page 100572, 2022.