Leakage of Dataset Properties in Multi-Party Machine Learning

Wanrong Zhang[§]
Georgia Institute of Technology

Shruti Tople
Microsoft Research

Olga Ohrimenko[§]
The University of Melbourne

Abstract

Secure multi-party machine learning allows several parties to build a model on their pooled data to increase utility while not explicitly sharing data with each other. We show that such multi-party computation can cause leakage of global dataset properties between the parties even when parties obtain only black-box access to the final model. In particular, a "curious" party can infer the distribution of sensitive attributes in other parties' data with high accuracy. This raises concerns regarding the confidentiality of properties pertaining to the whole dataset as opposed to individual data records. We show that our attack can leak *population-level* properties in datasets of different types, including tabular, text, and graph data. To understand and measure the source of leakage, we consider several models of correlation between a sensitive attribute and the rest of the data. Using multiple machine learning models, we show that leakage occurs even if the sensitive attribute is *not* included in the training data and has a low correlation with other attributes or the target variable.

1 Introduction

Modern machine learning models have been shown to memorize information about their training data, leading to privacy concerns regarding their use and release in practice. Leakage of sensitive information about the data has been shown via membership attacks [47, 50], attribute inference attacks [17, 53], extraction of text [8] and data used in model updates [46, 55]. These attacks focus on leakage of information about an individual record in the training data, with several recent exceptions [18, 39] pointing out that leakage of global properties about a dataset can also lead to confidentiality and privacy breaches.

In this paper, we study the problem of leakage of dataset properties at the *population-level*. Attacks on leakage of global properties about the data are concerned with learning information about the *data owner* as opposed to individuals whose privacy may be violated via membership or attribute inference attacks. The global properties of a dataset are confidential when they are related to the proprietary information or IP that the data contains, and its owner is not willing to share. As an example, consider the advantage one can gain from learning demographic information of customers or sales distribution across competitor's products.

Our primary focus is on inferring dataset properties in the *centralized multi-party machine learning setting*. This setting allows multiple parties to increase utility of their data since the model they obtain is trained on a larger data sample than available to them individually. Benefits of computing on combined data have been identified in multiple sectors including drug discovery, health services, manufacturing and finance. For example, anti-money laundering served as a use case for secure data sharing and computation during the TechSprint organized by the Financial Conduct Authority, UK in 2019 [5]. A potential machine learning task in this setting is to create a system that identifies a suspicious activity based on financial transactions and demographic information of an entity (e.g., a bank customer). Since multiple financial institutions have separate views of the activities, such a system can be used to detect common patterns.

Deployments and availability of secure computation methods [2, 29, 32, 48] can enable multi-party machine-learning by alleviating immediate privacy concerns of the parties. In particular, secure multi-party machine learning provides parties with a *black-box* access to a model

[§]Work done in part while at Microsoft.

trained on their pooled data without requiring the parties to share plaintext data with each other. Unfortunately, as we show in this paper, this is insufficient to address all privacy implications of collaborative machine learning. In particular, we demonstrate that *global properties* about one party's sensitive attributes can be inferred by the second party, even when only black-box access to the model is available. Consider implications of our attacks in the use case above. An attacker party (e.g., one of the banks) can learn distribution of demographic features pertaining to the customer population in the other bank (e.g., whether the other bank has more female than other customers or what percentage of customers has income over a certain threshold) that it can use in the future when developing a marketing campaign to attract new customers.

Analysis of our attacks shows that leakage of population-level properties is possible even in cases where sensitive attribute is irrelevant to the task, i.e., it has ≈ 0 correlation with the task in hand. Though removing sensitive attributes may seem like a viable solution, it is not provably secure due to correlations that are present in the data. Indeed, we show that in many cases, information is still leaked regardless of whether training data contained the sensitive attribute or not. We argue that this is possible due to correlation between sensitive attributes and other attributes that exists in the data. For example, datasets we use indicate that there is correlation between sets of attributes including gender, occupation and working hours per week, as well as income, occupation and age. Such customer attributes are often recorded by financial institutions, as a result indicating potential leakage if institutions were to collaborate towards detection of financial crime as described above.

Threat model. We consider the setting where the model is securely trained on the joined data of the honest party and of an *honest-but-curious* party. Honest-but-curious adversary considers a realistic setting where the malicious party (1) will not alter its own data — if it does, the model may not perform well and, if detected, could undermine the trust from the other party in the partnership — and (2) will not change the machine learning code — both parties may wish to observe the code to be run on the data to ensure its quality and security.

The attacker is interested in learning global properties about a sensitive attribute at the dataset level, that is, how values of this attribute are distributed in the other party's dataset. It may be interested in learning which attribute value is dominant (e.g., whether there are more females) or what the precise ratio of attribute values is (e.g., 90% females vs. 70% females).

Attack technique. We show that dataset property can be leaked merely from the black-box access to the model. In particular, the attacker does not require access to the training process of the model (e.g., via gradients [39]) or to model parameters (aka white-box attack [7, 18]). Following other attacks in the space, the attacker also uses shadow models and a meta classifier. However, individual predictions from the model are not sufficient to extract global information about a dataset. To this end, we introduce an attack vector based on a set of queries and use them in combination in order to infer a dataset property. In contrast to previous work on property leakage, the attack requires less information and assumptions on the attacker (see Table 1 and Section 8 for more details).

Methodology. To understand what causes information leakage about a property we consider several correlation relationships between the sensitive attribute A, the rest of the attributes X, and the target variable Y that the machine learning model aims to learn. Surprisingly, we show that dataset-level properties about A can be leaked in the setting where A has low or no correlation with Y. We demonstrate this with experiments on real data and experiments with a synthetic attribute where we control its influence on X and Y. The attack persists across different model types such as logistic regression, multi-layer perceptrons (MLPs), Long Short Term Memory networks (LSTMs), and Graphical Convolution Networks (GCNs) models and for different dataset types such as tabular, text, and graph data. The attack is efficient as it requires 100 shadow models and fewer than 1000 queries.

Machine learning settings. In addition to the multi-party setting, our property leakage attack can be carried out in the following two settings. (1) single-party setting where an owner of a dataset releases query interface of their model; (2) in the model update setting, one can infer how the distribution of a sensitive property has changed since the previous release of the model. The second attack also applies to multi-party machine learning, showing that the party that joins last exposes its data distribution more than parties who were already collaborating.

Contributions. Our contributions are as follows:

- *Problem Formulation:* We study leakage of properties about a dataset used to train a machine learning model when only black-box access to the model is available to the attacker.

	Attacker's knowledge	Single-party	Multi-party	Datasets
Melis *et al.* [39]	training gradients		✓	tabular, text, images
Ganju *et al.* [18]	model parameters (white-box)	✓		tabular, images
Ateniese *et al.* [7]	model parameters (white-box)	✓		tabular, speech
This work	model predictions (black-box)	✓	✓	tabular, text, graphs

Table 1: Comparison of attacks on leakage of dataset properties.

- *Attack Technique:* We propose an effective attack strategy that requires only a few hundred inference queries to the model (black-box access) and relies on a simple attack architecture that even a computationally bound attacker can use.
- *Attack Setting:* We show that leakage of dataset properties is an issue for an owner of a dataset when the owner releases a model trained on their data (single-party setting); when the owner participates in multi-party machine learning, and when the owner contributes data to update an already trained model (e.g., either because it joins other parties or because it has acquired new data).
- *Empirical Results:* We show that distribution of a sensitive attribute can be inferred with high accuracy for several types of datasets (tabular, text, graph) and models, even if the sensitive attribute is dropped from the training dataset and has low correlation with the target variable.

Finally, we note that secure multi-party computation, based on cryptographic techniques or secure hardware, [13, 19, 20, 26, 27, 34, 40, 41, 42, 54] guarantees that nothing except the output of the computation is revealed to the individual parties. However, it is not concerned with what this final output can reveal about the input data of each party. On the other hand, defenses, such as differential privacy, are concerned with individual record privacy and not dataset property privacy considered in this paper. We discuss this further in Section 7. In summary, we believe this work identifies a potential gap in multi-party machine learning research in terms of techniques that parties can deploy to protect global properties about their dataset.

2 Preliminaries

We assume that there is an underlying data distribution $\mathcal{D}$ determined by variables X, A, Y where X models a set of features, A models a feature that is deemed private (or sensitive) and Y is the target variable, i.e., either a label or a real value (e.g., if using regression models). We consider a supervised setting where the goal is to train a model f such that $f(X,A)$ predicts Y.

Secure multi-party computation (MPC). MPC lets parties obtain a result of a computation on their combined datasets without requiring them to share plaintext data with each other or anyone else. Methods that instantiate it include homomorphic encryption, secret sharing, secure hardware and garbled circuits [12, 14, 25, 43, 45]. These methods vary in terms of their security guarantees (e.g., availability of a trusted processor vs. non-colluding servers) and efficiency. We abstract MPC using an ideal functionality [43]: a trusted third entity accepts inputs from the parties, computes the desired function on the combined data, and returns the output of the computation to each party. Security of protocols implementing this functionality is often captured by proving the existence of a simulator that can simulate adversary's view in the protocol based only on adversary's input and the output of the computation. Hence, an MPC protocol guarantees that an adversarial party learns only the output of the computation but does not learn the content of the inputs of other parties beyond what it can infer based on its own data and the output. Since our attacks are oblivious to the exact technique used for secure computation, we assume ideal MPC functionality and specify additional information available to the adversary in the next section.

Multi-party machine learning. Let D_{honest} and D_{adv} be the datasets corresponding to the data of the victim parties and D_{adv} be the data that belongs to the parties whose data is known to the adversary. For simplicity, we model it using two parties P_{honest} and P_{adv} who own D_{honest} and D_{adv}, respectively. Both D_{honest} and D_{adv} are sampled from $\mathcal{D}$ but may have a different distribution of A, conditional on some latent variable, for example, a party identifier. Importantly, distribution of A in D_{honest} is secret and unknown to P_{adv}. Parties are interested in increasing the utility of their model through collaboration with each other. To this end, they agree on an algorithm to train a machine learning model, f, using their combined datasets D_{honest} and D_{adv}.

The parties use secure multi-party computation to train f, as they are not willing to share it either due to

privacy concerns or regulations. Once the target model is trained using MPC, it can be released to the parties either as a white- or black-box. In the former, f is sent to the parties, and, in the latter, the model is available to the parties through an inference interface (e.g., the model stays encrypted at the server such that inferences are made either using secure hardware or cryptographic techniques [28]). We assume that f is trained faithfully and, hence, P_{adv} cannot tamper with how f is trained (e.g., this avoids attacks where a malicious algorithm can encode training data in model weights [51]).

MPC guarantees that parties learn nothing about the computation besides the output, i.e., they learn no other information about each other's data besides what is revealed from their access to f. The goal of this paper is to show that even by having black-box access to f one party can infer information about other parties' data.

3 Data Modeling

To reason about leakage of A's distribution in $\mathcal{D}$, we consider different relationships between X, Y, A based on their correlation. We use $\sim$ to indicate that there is a correlation between random variables and $\perp$ if not. We consider four possible relationships between Y, X and the sensitive attribute A.

$\underline{Y \perp A}$: If Y is independent of A, and if f is faithfully modeling the underlying distribution, A should not be leaked. That is, information about A that an adversary acquires from $f(X,A)$ and $f'(X)$ should be the same for models f and f' trained to predict Y. Two scenarios arise depending on whether the rest of the features are correlated with A or not: $(X \perp A, Y \perp A)$ and $(X \sim A, Y \perp A)$. We argue that leakage in the latter case is possible due to how machine learning models are trained. Below we describe why it is theoretically feasible and experimentally validate this in Section 6.

A machine learning model is trying to learn the conditional probability distribution $\Pr(Y = y|X = x)$ where X are the attributes and Y is the target variable. Suppose there is a latent variable Z, and the observed X is modeled by $X = h(Z,A)$ where h is a function capturing the relationship between the variables. Even if the target variable Y only depends on Z through a random function g: $Y = g(Z)$, the conditional distribution $\Pr(Y = y|X = x)$ still depends on A. Thus, machine learning models will capture information about A. For example, consider a task of predicting education level (Y) based on data that contains gender (A) and income (X). Suppose income can be modeled by a function of latent variables skill and occupation, and education level is only associated with the skill. Though gender is not correlated with education level ($Y \perp A$), it could be associated with occupation and thus correlated with income (X).

The $(X \sim A, Y \perp A)$ scenario was also noted by Locatello *et al.* [38] when studying fair representations. The authors indicated that even if the original data may not have a bias (i.e., when the target variable and the protected variable are independent) using the protected attribute in training can introduce bias.

To model $(X \sim A, Y \perp A)$ scenario in the experiments, we use correlation coefficients to determine the split of dataset attributes into X and A. To have a more controlled experiment, we also carry out experiments where we introduce a synthetic variable and inject correlations between it and a subset of attributes in X.

$\underline{Y \sim A}$: We also consider two cases where there is a correlation between the target variable Y and the sensitive attribute A: $(X \perp A, Y \sim A)$ and $(X \sim A, Y \sim A)$. In the setting of $(X \perp A, Y \sim A)$, attribute A and a set of attributes X may be relevant in predicting Y, while being uncorrelated with each other. For example, a reaction of an individual to a new drug (Y) could depend on the age and weight of an adult, while age and weight may be regarded as independent between each other.

The final setting of $(X \sim A, Y \sim A)$ is the most likely scenario to happen in practice where the true distribution and dependence between variables maybe unknown. For example, consider a task of predicting whether a financial transaction by an individual is suspicious or not (Y) based on customer information (e.g., occupation, age, gender) and their transaction history (X), where their income is the sensitive attribute A. The correlation between attributes could either belong to cases $(X \sim A, Y \perp A)$ or to $(X \sim A, Y \sim A)$ since attributes such as occupation and age are likely to be correlated with income (as also suggested by the correlations in the datasets we use in our experimental evaluation in Appendix A).

4 Threat Model and Attack

The goal of the adversarial party P_{adv} is to learn population-level properties about the rest of the dataset used in the multi-party machine learning setting (e.g., in the two-party setting this corresponds to learning properties of the other party's dataset). Since P_{adv} is one of the parties, it has black-box access to the joint model f trained (e.g., via MPC) on the data of all the parties (i.e., D_{honest} and D_{adv}). Given this query interface to f, the attacker wants to infer how sensitive attribute A is distributed in honest parties' dataset D_{honest}. Throughout the paper, we use attribute and feature interchangeably.

We model dataset property leakage as follows. Let $\mathbf{a}_{\mathsf{honest}}$ denote attribute values of A for all records in D_{honest} (for example, if the sensitive attribute is gender, then $\mathbf{a}_{\mathsf{honest}}$ is a vector of gender values of all records in P_{honest} data). We define $p(\mathbf{a}_{\mathsf{honest}})$ to be the property or information about $\mathbf{a}_{\mathsf{honest}}$ that the adversary is trying to infer. For example, the property could be related to determining whether there is a higher presence of female patients in the dataset D_{honest} or learn the exact ratio of female patients.

The attacker, besides knowing its own dataset D_{adv} and having black-box access to the model f, is assumed to have auxiliary dataset D_{aux} that is distributed according to $\mathcal{D}$. Similar to [50], an auxiliary dataset can be generated either via (1) model-based synthesis approach — feeding synthetic data to f and using its output to guide the search towards data samples on which the model returns predictions with high confidence, (2) statistics-based synthesis that uses information about marginal distribution of the attributes, or (3) using a (publicly available) dataset of similar distribution. The attacker can use approach (1) by merely using f, while D_{adv} provides it with statistics for (2). The availability of a dataset that follows similar distribution to $\mathcal{D}$ depends on the setting. Consider the anti-money laundering use case in the introduction. A party may have access to billions of financial transactions that it can use either for approach (2) since record-level marginal distribution between demographic features, income, education level is likely to be similar between the parties, or for approach (3) by dividing its dataset into D_{aux} and D_{adv}.

The attack follows the shadow model training approach [7, 50]. However, we modify the attack vector to measure the signal about the distribution of a sensitive attribute in a whole dataset. Our attack strategy is described below; Figure 1 shows graphical representation of how the attack model is trained and Figure 2 shows the execution of an attack on target model f.

We make an observation that to infer global properties about training data, the attacker needs to combine information from multiple inferences made by f. To this end, the attacker measures how f performs on a sequence of k records, called D_{attack}, as opposed to a single record used in work on attribute and membership inference. We obtain the "attack feature" sequence $\mathcal{F}$ by setting it to the posterior probability vector across classes returned by f on D_{attack}. Hence, if f is a classification model over l classes $\mathcal{F}$ consists of $k \times l$ values. In the experiments, we construct D_{attack} by sampling from D_{aux} at random. We leave open a question of whether more sophisticated methods of constructing D_{attack} can lead to better attacks.

Shadow models and attack meta-classifier. The attacker relies on shadow models in order to determine whether $\mathcal{F}$ is generated from f trained on a dataset with property p or not. To this end, the attacker trains n "shadow" models that resemble f. In particular, it generates training datasets D^i_{shadow}, half of them exhibiting the property and half not, labeled as p and $\bar{p}$ accordingly. These datasets could be obtained by resampling from D_{aux}. Each shadow model f^i_{shadow} is trained on a dataset $D^i_{\mathsf{shadow}} \cup D_{\mathsf{adv}}$ using the same way as the target central model f. Once f^i_{shadow} is trained, the attacker queries it using D_{attack} and combines inference results to form a feature vector $\mathcal{F}_i$ associated with p or $\bar{p}$, depending on its training data.

After training all shadow models, the adversary has a set of features $\mathcal{F}_i$ with the corresponding property label $p_i \in \{p, \bar{p}\}$. The adversary then trains a meta-classifier on the pairs $\{(\mathcal{F}_i, p_i)\}_i$ using any binary classification algorithm. For example, logistic regression is sufficient for attacks in our experimental evaluation.

The attacker carries out its attack as follows. Once the target model f is trained on the joined data of the attacker and honest party, the attacker queries the model using D_{attack} to obtains the feature representation of the target model, $\mathcal{F}$. It then feeds $\mathcal{F}$ to its meta-classifier and obtains a prediction for the sensitive property $p(\mathbf{a}_{\mathsf{honest}})$.

Single-party attack. We explained the attack strategy for the multi-party case since this is the primary focus of this work. However, we can easily adapt the attack to the single-party case: the only change that has to be made to the attack description above is by setting D_{adv} to an empty set. As highlighted in Table 1, besides being the first attack on property leakage in the centralized multi-party setting, our attack is also the first to show that dataset properties can be leaked in the black-box setting.

Fine-grained attack. The above attack shows how an adversary can learn whether some property is present in a dataset or not. The attacker can extend this binary property attack and distinguish between multiple properties $P = \{p^1, p^2, \ldots\}$. It simply generates shadow training datasets for each property and then trains a meta-classifier to predict one of the properties in P based on attack vector $\mathcal{F}$. For example, P can be a set of possible ratios of females to other values, and the attack meta-classifier will try to distinguish whether it is 10:90, 50:50 or 90:10 split. In the experimental evaluation, we show that this

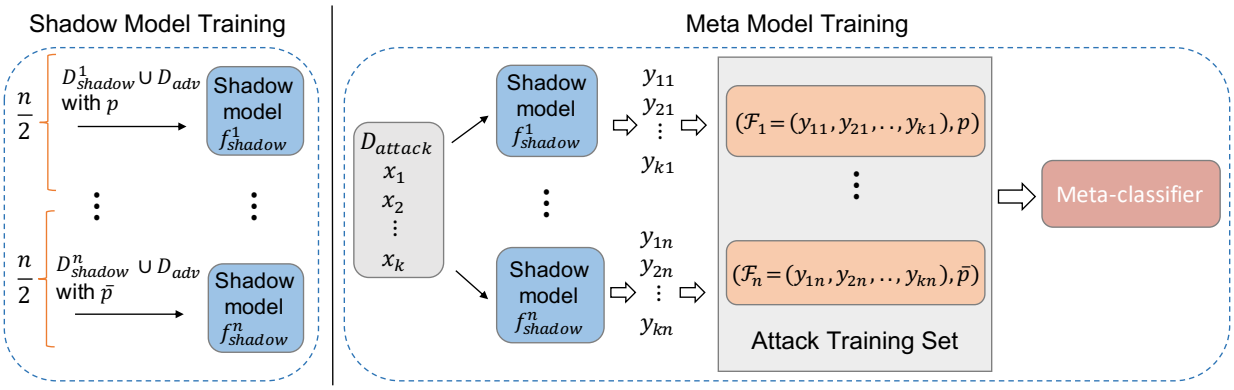

Figure 1: Attack model pipeline. Half of shadow models are trained with the property p that the attacker is trying to learn and half without it. Each shadow model f^i_{shadow} is queried on a dataset D_{attack}. Output probability vectors are concatenated to form a vector $\mathcal{F}_i$. Finally, the meta-classifier is trained on feature-label tuples of the form $\{(\mathcal{F}_i, p_i)\}_i$.

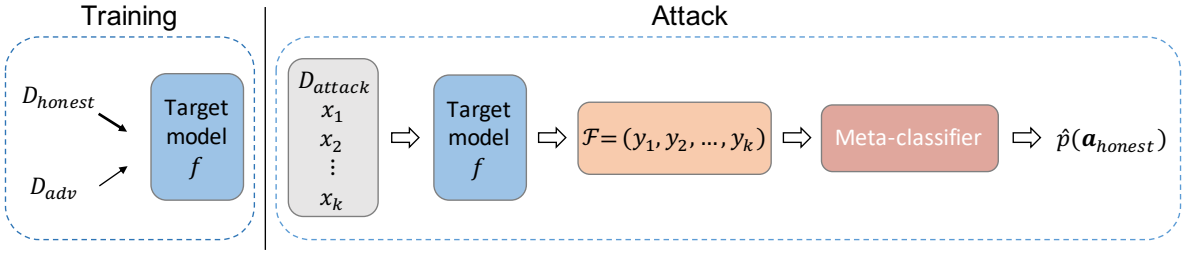

Figure 2: Execution of the attack on the target model to learn the prediction of the property $p(\mathbf{a}_{\text{honest}})$ in D_{honest}, $\hat{p}$.

attack is effective in learning fine-grained distribution of sensitive attributes as well as identifying how the distribution of a sensitive attribute has changed after the model was updated with new data.

Scope. This work focuses on understanding the leakage of population-level properties of the training dataset. Since our threat model is similar to that of the attacker who is able to infer individual record-level attributes [17, 51, 53], our setting allows for record-level leakage as well. Albeit, the attack strategy needs to be changed in order to train shadow models that capture the difference between inputs with different attribute values. Importantly, for both the record-level and population-level attribute inference attack, the attacker — here and in [51, 53] — is assumed to know the domain of an attribute it is trying to infer (e.g., `Gender` taking values male, female, or other). Hence, similar to prior work [18, 39], our attack cannot infer a sensitive attribute with a large, potentially unbounded, domain (e.g., such as `Name` for which the attacker may not be able to enumerate all possible values).

5 Experimental Setup

The goal of our experiments is to evaluate the efficacy of the attack in Section 4 to learn population-level properties about a sensitive attribute in the multi-party and single-party machine learning setting. We then aim to understand how the difference in machine learning models (e.g., logistic regression and neural network models), dataset type (e.g., tabular data, text or graph), access to the model through its weights or inference interface, and attribute correlation influence attack accuracy.

5.1 Benchmark Datasets

We evaluate our attack on five datasets described below. The datasets, sensitive attributes, machine learning model tasks, and the type of correlations between the sensitive attribute, other attributes, and the final task are summarized in Table 3.

Health [3] The Health dataset (Heritage Health Prize) contains medical records of over 55 000 patients. Similar to the winners of the Kaggle competition, we use 141 features with `MemberID` and `Year` removed. We

group the DaysInHospital attribute into two classes. The task, Y, is to predict if a patient will be discharged, DaysInHospital $= 0$, or will stay in the hospital, DaysInHospital > 0. We consider two sensitive attributes to perform our attack on learning their distribution in the dataset of the benign party: Gender and the number of medical claims ClaimsTruncated.

Adult [33, 37] The Adult dataset contains US census information including race, gender, income, and education level. The training dataset contains 32 561 records with 14 attributes. We group the education level into four classes: 'Low', 'Medium-Low', 'Medium-High', 'High'. We use 12 features with Education and Fnlwgt removed. The task is to predict the class of the EducationLevel (i.e., variable Y for this dataset). We again consider two sensitive features whose distribution the attacker is trying to infer: Gender and Income.

Communities and Crime [37] The Communities and Crime dataset contains 1 994 records with 122 features relevant to per capita violent crime rates in the United States, which was also used for evaluating fairness with respect to protected variables [11]. We remove the attributes that have missing data, resulting in 100 attributes. The classification task is to predict the crime rate, i.e., the Y variable is CrimesPerPop. We group the crime rate into three classes based on ranges: '< 0.15', '$[0.15, 0.5]$' and '> 0.5', and the task is the multi-class prediction for the crime rate. We consider total percentage of divorce TotalPctDiv and Income as sensitive features.

Yelp-Health [4] The Yelp dataset contains 5 million reviews of businesses tagged with numerical ratings (1-5) and attributes such as business type and location. We extract a healthcare-related subset that has 2 384 reviews for pediatricians and 1 731 reviews for ophthalmologists. The classification task is to predict whether the review is positive (rating > 3) or negative (rating ≤ 3). The attack aims to predict the dominant value of the doctor Specialty of the benign party.

Amazon [1, 35] The Amazon product co-purchasing network dataset contains product metadata and reviews information about 548 552 different products such as books and music CDs. For each product, the following information is available: the similar products that get co-purchased, product type, and product reviews. We use a subset of 20 000 products and construct a product co-purchasing network, where each node represents a product and the edge represents if there is at least one reviewer who rated both products, indicating that products are bought by the same user [36]. Each node is associated with one of 4 product types and an average review score from 0 to 5, including half-score reviews (i.e., 11 possible scores in total). The classification task (for a recommendation system) is to predict the average review score of the node given the co-purchasing network and the product types. Depending on the classification task, we split reviewer scores into 2 classes: positive vs. negative review, 6 classes: rounded integer review between 0,1.., 5 and 11 classes: the original review score. The attack aims to predict whether the dominant value of the attribute ProductType of the benign party is "books".

5.2 Evaluation Methodology

Target model f. We train different target models depending on the dataset type. For tabular data, i.e., Adult, Health, and Crime, we train multinomial logistic regression and fully-connected multi-layer perceptron neural networks (MLP). For the Adult and Crime datasets, we use an MLP network with one hidden layer of size 12 and the last layer with 4 and 3 output classes, respectively. For the Health dataset, we use an MLP network with one hidden layer of size 20 and binary output. In later sections, a neural network model for tabular datasets always refers to an MLP network. In training our target models, we use the Adam [30] optimizer, ReLu as the activation function, a learning rate of 0.01, and a weight decay of 0.0001. For the Yelp-Health dataset, we use the pre-trained glove embedding of dimension 50, a bidirectional LSTM layer of dimension 50. We then use one hidden layer of size 50 and dropout regularization with parameter 0.1 between the last hidden layer and the binary output. For the Amazon dataset, we train the target model using the Graph Convolutional Networks (GCN) [31] with 1 hidden layer of 16 units, Adam as the optimizer, ReLu as the activation function, a learning rate of 0.01, and a weight decay of 0.0005. Each experiment is repeated 100 times, and all attack accuracies are averaged over these runs. As noted in Section 2, our attacks are oblivious to how f is trained, hence, in the experiments training is done in the clear.

Dataset split. In the multi-party setting, we consider two parties that contribute data for training the target model where one of the parties is trying to learn information about the data of the other party. For Adult and Health datasets, each party contributes 2 000 samples. We use 10 000 or 4 000 samples as D_{aux} to train the shadow models and the attacker uses 1 000 samples in D_{attack} to query the model and obtain the attack vector for the meta-classifier. Table 2 summarizes the splits for all other datasets. In Section 6.4 we show that a small number of

Datasets	$\#D_{\text{adv}}$, $\#D_{\text{honest}}$	$\#D_{\text{aux}}$	$\#D_{\text{attack}}$
Health [3]	2 000	10 000 / 4000	1 000
Adult [33, 37]	2 000	10 000 /4000	1 000
Crime [37]	200	1 500 / 400	94
Yelp-Health [4]	1 000	1 200	200
Amazon [35]	5 000	10 000	1 000

Table 2: Dataset split during the attack where $\#D_{\text{attack}}$ is the number of inference queries the attacker makes to the model.

samples in D_{attack} can lead to high attack accuracy as well (e.g., 200 vs. 1 000 for the Amazon dataset).

The distribution of the values of the sensitive attribute A in datasets is determined as follows. We consider the default split of 33:67 in the attacker's data D_{adv} (e.g., 33% of records are books). The attack is evaluated against several D_{honest} datasets for each possible split. For example, we evaluate our attack on 100 D_{honest} datasets: half with 33:67 split and half with 67:33 split in Sections 6.1 and 6.2. Throughout all experiments, the D_{aux} always has 50:50 split.

Attack setting. We report our main results on attack in the black-box setting; white-box results are deferred to Appendix B. We use two different meta-classifiers depending on the target model. For multinomial logistic regression, LSTM and GCN, the meta-classifier model is a binary logistic regression model. For MLP as the target model, we use a two-layer network with 20 and 8 hidden units and a learning rate of 0.001. The meta-classifier models are trained using Adam optimizer.

We perform the attack when the model is trained with the sensitive variable (A) and without it ($\bar{A}$). For the $\bar{A}$ setting, the attribute A is omitted from the machine learning pipeline, including the shadow model training and construction of D_{attack}. This setting allows us to understand the risk of leaking a sensitive attribute, even when that attribute is censored during training. For Yelp-Health, we report only $\bar{A}$ results as LSTM takes the text data, and A would be an additional feature.

Types of experiments. We study how correlations between attributes affect the attack. We show that information is leaked even when A is not correlated with the final task. We demonstrate our attack on attribute correlation as present in real dataset distributions (shown in Table 3) as well as artificially injected correlation using a synthetic sensitive variable. The latter allows us to control the correlation between the variables.

Real Data. For the experiments where all features are from the real data, including the sensitive variable, we set different variables as sensitive (A) for each dataset and perform a black-box attack using a default split of 33:67 for the sensitive attribute in the attacker's data (D_{adv}).

We compute the pairwise correlation among all the variables using Pearson correlation coefficient [44] for numerical-numerical variables, Cramer's V [10] for categorical-categorical variables, point-biserial correlation coefficient [49] for binary categorical-numerical variables, and ANOVA for multi-level categorical-numerical variables. Based on the observed correlations, for each dataset, we identify the case among those introduced in Section 3. Most scenarios correspond to $X \sim A, Y \sim A$. Details on correlation factors for all datasets are deferred to Appendix A.

Synthetic Data. For synthetic experiments, we create a new synthetic attribute as our sensitive variable A for the Adult and Health datasets. We add a correlation of A to a subset of variables in the dataset, denoted as $X' \subseteq X$, and the target variable Y, depending on the cases outlined in Section 3. We introduce the correlation by replacing attribute values in X' and/or Y for each record with values that have an injected correlation with A. For Adult dataset, X' is `Income`, for Health dataset, $X' = \{\texttt{DrugCountAve}, \texttt{LabCountAve}, \texttt{ClaimsTruncated}\}$. The variable A takes values < 5 or > 5 that are split using 33:67 ratio in the adversarial party's dataset. The honest party has two possible splits: 33:67 ratio and 67:33 ratio. The attacker's goal is to guess the distribution of A in the data of P_{honest}.

6 Attack Results

We evaluate for attribute leakage in the following settings: the single-party case where an attacker learns the distribution of an attribute in the training set and the multi-party case where an attacker learns the distribution of an attribute in the data of the honest party. Apart from inferring the dominant attribute (e.g., there are more females than males in a dataset), we perform a fine-grained attack that learns a precise distribution of the two attribute values (e.g., 70% of the dataset are females). We further use this fine-grained attack to infer the change in the attribute distribution in a model update scenario where the model is updated either due to a new party joining or new data arriving. Attack accuracy higher than the probability of a random correct guess is considered successful as this indicates that confidential property (i.e., information about P_{honest}'s data) will be leaked to the attacker in the majority of cases.

We report our attack results in the stronger black-box

Datasets	Sensitive attribute A	Task Y	Correlation
Health [3]	`Gender` `ClaimsTruncated`	`DaysInHospital`	$X \sim A, Y \perp A$
Adult [33, 37]	`Gender` `Income`	`EducationLevel`	$X \sim A, Y \perp A$ $X \sim A, Y \sim A$
Crime [37]	`TotalPctDivorce` `Income`	`CrimesPerPop`	$X \sim A, Y \sim A$
Yelp-Health [4]	`Specialty`	`ReviewRating`	$X \sim A, Y \perp A$
Amazon [35]	`ProductType`	`ReviewScore`	$X \sim A, Y \sim A$

Table 3: Datasets, tasks and attribute-label correlation where $\sim$ and $\perp$ indicate correlation and no correlation, respectively.

setting for real, synthetic, and fine-grained experiments. We evaluate the white-box attack, where the attacker has access to model parameters, only on the synthetic data. We summarize our key findings below:

- Leakage of sensitive dataset properties in honest party's data is possible even when the sensitive attribute itself is *dropped* during training and has low or no correlation with the final task. We show that the attack accuracy drops only by a few percent when A is not present in many cases.
- An adversary can learn the attribute properties of the honest party's data irrespective of whether it contributes data (multi-party) or not (single-party) to the training dataset.
- For the models and datasets considered in this paper, our property leakage attack is dataset and model-agnostic and works on tabular, text, or graph data.
- Fine-grained attacks can be used to predict a precise distribution of the attribute as well as learn the change in data distribution during model updates.

6.1 Multi-Party Setting

Real Data. Table 4 shows the attack accuracy for correlations observed in the real distribution of datasets, with the larger size of D_{aux} as listed in Table 1. The attack accuracy with the smaller size of D_{aux} is deferred to Table 12 in Appendix B. We see that the attack accuracy is always better than a random guess in all experiments, regardless of whether the sensitive attribute is included in the training data or not.

We make the following observations. The attack accuracy for Adult data with `Income` as the sensitive attribute is the highest with 98% and 96% when the target model is trained with and without A, respectively. Overall, the attack accuracy ranges between 61-98% when trained with sensitive variable (A) and 59-96% without ($\bar{A}$), respectively. The results for $\bar{A}$ are always lower than with A but are, however, above the random guess baseline of 50%. For the Amazon dataset, we observe that attack accuracy is higher for fewer output classes. We confirm this observation later in Figure 4. We also note that the attack accuracy decreases as the size of D_{aux} decreases as shown in Appendix B.

To understand how the correlation between A and other features influences the attack, we determine which attributes $X' \subseteq X$ are correlated with A. We set X' to variables based on their correlation factors. Details on how X' of each dataset was determined based on correlation factors is deferred to Appendix A. In Table 4, # X' denotes the number of attributes correlated with the sensitive attribute A. We note that simultaneously controlling the number of correlated attributes and their correlation strength is hard on real data, so we also use synthetic datasets. We observe that, for the same dataset, the attack accuracy increases with a higher number of correlated attributes X' and the sensitive attribute A.

We show the accuracies for both the pooled model and the honest party's local model in Table 11 in Appendix B. Across all these experiments, we observe a utility increase ranging from 0.58% and 5.90% for the honest party, which motivates the honest party to collaborate and train a joint target model with the other party.

Synthetic Data. Table 5 shows our results with a synthetic variable A introduced in the Adult and Health dataset for the multi-party setting. Here, we train the same dataset using both logistic regression and the neural network model (MLP). Recall that the synthetic attribute is introduced to imitate a sensitive variable to control its correlation with other variables. To this end, we create

Datasets (Output Classes)	Model Type	Attack Accuracy		A	# X'
		A	$\bar{A}$		
Health (2)	Multi-layer Perceptron	.61	.59	`Gender`	24/139
		.75	.71	`ClaimsTruncated`	54/139
Adult (4)	Logistic Regression	.83	.81	`Gender`	5/11
		.98	.96	`Income`	9/11
Crime (3)	Multi-layer Perceptron	.61	.59	`TotalPctDivorce`	26/98
		.78	.60	`Income`	38/98
Yelp-Health (2)	LSTM	-	.74	`Specialty`	review text
Amazon (2)	GCN	.86	.72	`ProductType`	graph
Amazon (6)	GCN	.62	.63	`ProductType`	graph
Amazon (11)	GCN	.67	.61	`ProductType`	graph

Table 4: Multi-Party Setting: Black-box attack accuracy for predicting the value of the distribution of sensitive variable A in the dataset of P_{honest}. The attacker tries to guess whether values of A are split as 33:67 or 67:33 in D_{honest} when its own data D_{adv} has 33:67 split. Columns A and $\bar{A}$ report the accuracy when the sensitive variable is used for training and not, respectively. X' indicates with which features in the dataset and with how many of them A is correlated. Since attack accuracy based on a random guess is 0.5, the attacker is always successful in determining the correct distribution.

datasets for different correlation criteria among the sensitive variable A, the output Y, and the remaining variables X. We report two findings.

First, logistic regression models appear to be at a higher risk, with average attack accuracy being higher as compared to neural network models: 84.5% vs. 71.3% for Adult and 80.2% vs. 70.8% for Health datasets. We suspect that this is mainly due to their simple architecture, which is easy to learn using a meta-classifier.

Second, the attack works well (greater than 74%) when the sensitive variable A is correlated with the target variable Y irrespective of its relation with X, i.e., cases where $Y \sim A$. The attack accuracy is almost equal to a random guess when $Y \perp A$. Recall that in the case of $X \sim A$, not all features used for training are correlated with A but only those in a subset of X, X'. To understand this scenario further, we reduced the number of features used during training to 3 (we refer to this setting as `R` in the tables). As the number of training features decreases, the correlation signal between A and X' becomes stronger, and the logistic regression model can capture that.

Our experiments for the case when both X and Y are independent of the sensitive variable A exhibit attack accuracy that is close to a random guess. This is expected as the variable has no correlation that the model can memorize, and hence we exclude them from Table 5.

6.2 Single-Party Setting

In addition to our motivating scenario of the multi-party setting, we evaluate the efficacy of our attack in the single-party setting where the attacker does not contribute towards the training data. For example, this corresponds to a scenario where a model is trained on data from only one hospital and is offered as an inference service for other hospitals. Table 6 shows the result for our attack using synthetic data for the Adult and Health dataset when the model is trained using both logistic regression and neural networks. We see that the attack in the single party setting is stronger since the adversary does not provide its own data, which may dilute the signal from the other party. For the case where $Y \sim A$, the attack accuracy is higher than 90%, even if the attribute itself is not used during training. This shows that our attack is highly successful even when the attacker does not participate in the training process.

6.3 Fine-grained Attack

Information leaked about attribute values can be either in terms of a binary signal, i.e., which attribute value is dominant in the dataset or an exact distribution. The results above show the leakage of the former. To learn

Model	Logistic Regression				Neural Network			
Datasets	Adult		Health		Adult		Health	
Synthetic Variable	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$
$X \sim A, Y \sim A$	1.00	1.00	1.00	1.00	.90	.84	.79	.95
$X \perp A, Y \sim A$	1.00	1.00	.99	1.00	.98	.98	.74	.98
$X \sim A, Y \perp A$	.65	.57	.52	.41	.52	.52	.52	.51
$X \sim A, Y \perp A$ (R)	.79	.75	.78	.72	.51	.45	.54	.63

Table 5: Multi-party setting: Black-box attack accuracy for predicting whether the values of (sensitive) synthetic variable A in the data of the honest party are predominantly < 5 or > 5. The attack accuracy is evaluated on 100 D_{honest} datasets: half with 33:67 and half with 67:33 split. A synthetic correlation with A is added to the variables X and Y depending on the specific case. R corresponds to the setting where only 3 attributes are used for training instead of all data. Attack accuracy based on a random guess is 0.5.

Model	Logistic Regression				Neural Network			
Datasets	Adult		Health		Adult		Health	
Synthetic Variable	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$
$X \sim A, Y \sim A$	1.00	1.00	.98	1.00	.98	.99	.92	.95
$X \perp A, Y \sim A$	1.00	1.00	.98	1.00	.99	1.00	.89	.98
$X \sim A, Y \perp A$	.67	.60	.48	.53	.56	.52	.52	.49
$X \sim A, Y \perp A$ (R)	.86	.74	.61	.62	.68	.66	.54	.61

Table 6: Single-party setting: Black-box attack accuracy with synthetic data.

information about the exact distribution, we present a variation of our main attack called the fine-grained attack. For this attack, we train a 5-class meta-classifier model that outputs whether a particular value of the sensitive attribute appears in 10%, 30%, 50%, 70%, or 90% of the dataset. Note that we train only one meta-classifier model with 5 output classes, but the attacker can perform a more systematic binary search over the distribution by training multiple meta-classifier models. We apply this attack in two settings.

Leakage of Attribute Distribution. We evaluate on the Adult dataset using a synthetic variable A as well as the gender variable. Table 7 shows the results for our fine-grained attack for predicting the precise distribution of the sensitive variable. The row 30 : 70 corresponds to the setting where 30% of records in D_{honest} have the value of the sensitive attribute A less than 5. Here, the attacker tries to guess the split of 30 : 70 among five possible splits of 10 : 90, 30 : 70, etc. The baseline accuracy is 20% because the attacker wishes to distinguish between 5 splits. Since the attack accuracy is always higher than the random guess, the attacker can successfully find the correct ratio by training a meta-classifier that distinguishes between different splits of the sensitive attribute values. Similar to the observation in Section 6.1, we observe that logistic regression has higher attack accuracy than neural networks. The attack accuracy for the real data with gender as the sensitive attribute is consistently greater than the 20% baseline for random guessing for all the distributions.

Distribution of A in D_{honest}:	LR Synthetic A		NN Synthetic A		LR A: Gender
	A	$\bar{A}$	A	$\bar{A}$	$\bar{A}$
10 : 90	.994	.998	.84	.89	.44
30 : 70	.993	.991	.79	.79	.59
50 : 50	.999	.997	.79	.73	.50
70 : 30	.997	.989	.73	.71	.46
90 : 10	.993	.998	.72	.77	.53

Table 7: Fine-grained attack accuracy for predicting the precise distribution of sensitive variable A in D_{honest} in the synthetic setting $X \perp A, Y \sim A$, and real data setting when A is Gender on the Adult dataset. Attack accuracy based on a random guess is 0.2.

Model Update Setting. We apply the fine-grained attack to learn the change in the distribution of an attribute value given access to an updated version of a model. In this attack, the malicious party initially obtains a model that is jointly trained on D_{honest1} and D_{adv}. Later, another honest party D_{honest2} joins, and a new model is trained on the three parties' data. The attacker tries to infer the dominant value of the sensitive attribute of P_{honest2} given the original and the updated model. It uses a fine-grained attack against both models, as result learning a dominant value in D_{honest1} and $D_{\text{honest1}} \cup D_{\text{honest2}}$. It then compares the two and infers how D_{honest2} has affected the distribution. If the split is dominated by the same attribute value in both models, the attacker uses this attribute value distribution as its guess. Otherwise, the attacker makes a guess that the other attribute value is dominated in D_{honest2}. Table 8 shows the results for our attack in the model update setting using synthetic data for the Adult dataset. The attack accuracy is almost close to 100% for the synthetic case and ranges from 63% to 86% for the Gender variable which is higher than a random guess of 50%.

6.4 Attack Parameters

We perform ablation experiments to understand the effect of varying the number of queries, distribution of

Distribution of A in D_{honest1}:	Distribution of A in D_{honest2}:	LR Synthetic A	LR A: `Gender`
30:70	30:70	1.00	.87
	70:30	.99	.72
70:30	30:70	.99	.63
	70:30	1.00	.85

Table 8: Model update setting: attack accuracy for predicting the dominant value of sensitive variable A in D_{honest2} in the synthetic setting $X \perp A, Y \sim A$ and real data setting when A is `Gender` on Adult dataset when A is removed from the training data. D_{adv} has 50:50 split. Attack accuracy based on a random guess is 0.5.

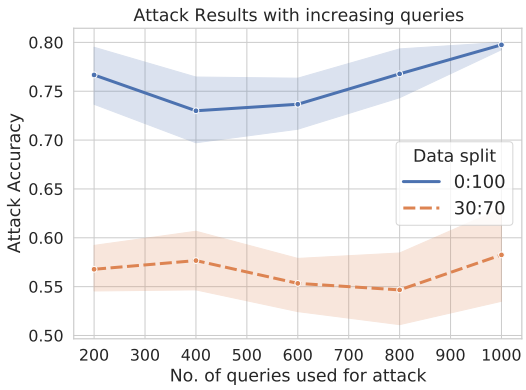

Figure 3: Attack accuracy for leaking sensitive attribute `ProductType` on the Amazon graph data (11 output classes) as the number of queries to the model increases.

the sensitive attribute and the number of output classes on the attack accuracy. We use the Amazon graph data for these experiments where, as before, `ProductType` is the sensitive attribute, and `ReviewScore` is the target.

Number of queries. We compute the attack accuracy for two different splits of values of the sensitive attribute, 0:100 (all books) and 30:70 (70% books, 30% of other products), and train the model to predict one of 11 review scores averaged over 10 runs. Figure 3 shows the effect of increasing the number of queries on the attack accuracy. Note that the number of queries also correspond to the input features of our attacker classifier. We observe that changing queries does not significantly impact the attack accuracy. With 1000 queries, attack accuracy is up to 80% for the 0:100 split and ≈59% for 30:70 split.

Attribute distribution and number of output classes. Figure 4 shows the results for the GCN trained on the Amazon dataset for 2, 6 and 11 output classes for the

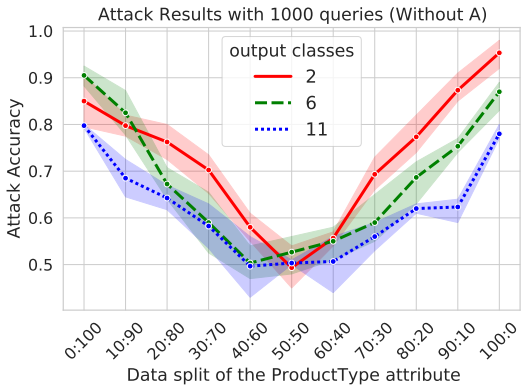

Figure 4: Attack accuracy for the Amazon graph data when the sensitive attribute `ProductType` is not used during training for different numbers of output classes across different distributions (splits).

review score. We evaluate for all the splits between 0:100 to 100:0. First, we observe that the attack accuracy drops as the ratio of the sensitive attribute values changes from 0:100 to 50:50 and increases again gradually from 50:50 to 100:0. This is because our primary attack is designed to identify the dominant attribute value. For inferring the distribution in the balanced range, the attacker can perform our fine-grained attack discussed in Section 6.3. Next, we observe that the attack accuracy is lower for a higher number of output classes such as 6 and 11 as compared to 2. This could be due to lower number of input features that are given to the attack classifier when there are lower number of output classes — the classifier is able to learn the attribute distribution better when the information is divided among fewer features thus resulting in a lower dimension input. Similar trends are observed in Figure 5 in Appendix when A is used during training.

7 Defenses

In the previous section, we saw that removing the sensitive attribute from the dataset is not an effective solution due to the correlations that exist between the attributes. Disentangling data representation through variational-auto-encoders [11, 23, 56] allows one to obtain mutually independent variables for representing the data. Intuitively, the removal of this variable before decoding the record for further down-stream tasks would lead to better censorship. Similarly, adversarial learning has also been proposed for learning a privacy-preserving data filter in a multi-party setting [21] and a privacy-preserving record representation [16]. Unfortunately, such techniques do not have provable worst-case guarantees and have been shown ineffective in the privacy context [53].

Differential privacy [15] guarantees record-level privacy, that is, whether a particular record is in their dataset or not. However, differential privacy does not protect population-level properties of a dataset [9, 15]. In fact, a differentially private algorithm with high utility aims to learn population properties without sacrificing individual privacy. Group differential privacy is an extension of differential privacy that considers the privacy of a group of k correlated records, as a result one way of achieving it is to increase, for example, Laplace noise, proportional to k. Though it can be applied to preserve the privacy of all records in each party's dataset by setting k to the size of each party's data, depending on the setting, it can effect utility as even with $k = 1$ accuracy of models have been shown to drop [6, 52].

In settings with more than two parties, where the attacker controls only one party, the signal weakens as it is harder for the adversary to identify the mapping between a property and a party whose data exhibits it. This was also noted by Melis *et al.* [39] in the federated learning setting with a small number of parties.

8 Related work

Membership attacks on machine learning models aim to determine whether a certain record was part of a training dataset or not [47, 50]. These attacks train shadow models that are similar to the target model and use their output (e.g., posterior probabilities over all classes) to build a meta-classifier that classifies records as members of the training data or not based on inference results of the target model on the record in question. A recent link stealing attack on graphs can be seen as a type of a membership attack that tries to infer whether two nodes have a link between them in the training graph [22].

Attribute inference attacks [17, 53], on the other hand, aim to determine the value of a sensitive attribute for a single record. For example, the authors of [53] study leakage of a sensitive value from a latent representation of a record in the model (i.e., a feature extractor layer); an attacker can obtain such intermediate record representations from having access to model parameters. They show that an attribute of a record, even if censored using adversarial learning, can be leaked. Hitaj *et al* [24] show that a malicious party can construct class representatives from a model trained in federated learning setting.

The work by Ganju *et al.* [18] and Ateniese *et al.* [7] are closest to ours as they also consider leakage of *dataset properties*. Different from this work, their attack is set in a single-party setting and requires a *white-box access* to the model, i.e., its parameters, that may not always be possible (e.g., when the model access is via cloud-hosted interface). Since the number of model parameters in neural networks can be very large (several million), approaches that are based on sophisticated methods for reducing network representation are required [18]. We show that attacks based on a combination of inferences and logistic regression as a meta-classifier are sufficient to learn attribute distribution.

Property leakage in a multi-party learning has been demonstrated only in federated setting [39]. In this setting an attacker obtains a gradient computed on a small batch of records (e.g., 32) and tries to learn how a sensitive feature is distributed in the batch. This setting is arguably easier from the attacker point of view: an attacker gains access to a much more granular computation on the data compared to the access to a query interface of the final model trained on the whole dataset, as considered in this paper. Moreover, previous work on dataset property leakage [7, 18, 39] did not consider the case when the sensitive attribute is *removed* from the data and the effect it has on the success of their attacks.

Recently, Zanella-Béguelin *et al.* [55] have demonstrated leakage of text and general trends in the data used to update next word prediction model. Salem *et al.* [46], on the other hand, consider granular leakage about records used to update the model: record labels and their features. Similar to our work, Salem *et al.* use a probing dataset to query the models to obtain the posterior difference. This output is then given to an encoder-decoder framework to reconstruct the meaning of the difference between posteriors of the initial and updated models. Our model update attack, in comparison, is about identifying the distribution of a sensitive feature in the dataset used to update the model and requires a simple machine learning architecture.

9 Conclusion

We demonstrate an attack, set in the centralized multi-party machine learning, that lets one of the parties learn sensitive properties about other parties' data. The attack requires only black-box access to the model and can extract the distribution of a sensitive attribute with small number of inference queries. We show that trivial defenses such as excluding a sensitive attribute from training are insufficient to prevent leakage. Our attack works on models for tabular, text, and graph data and datasets that exhibit various correlation relationships among attributes and class labels. Finally, we note that existing techniques for secure computation and differential privacy are either not directly applicable to protect leakage of population-level properties or do so at a high cost.

Acknowledgements

We thank Marcella Hastings and the anonymous reviewers for their valuable comments on the paper.

References

[1] Amazon product co-purchasing network metadata. http://snap.stanford.edu/data/amazon-meta.html.

[2] Azure confidential computing, Microsoft Azure. https://azure.microsoft.com/en-au/solutions/confidential-compute.

[3] Kaggle health dataset. https://www.kaggle.com/c/hhp.

[4] Yelp open dataset. https://www.yelp.com/dataset.

[5] Global AML and Financial Crime TechSprint. https://www.fca.org.uk/events/techsprints/2019-global-aml-and-financial-crime-techsprint, 2019. [Online; accessed 25-Jan-2021].

[6] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *ACM Conference on Computer and Communications Security (CCS)*. ACM, 2016.

[7] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici. Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers. *Int. J. Secur. Netw.*, 2015.

[8] N. Carlini, C. Liu, U. Erlingsson, J. Kos, and D. Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium*, 2019.

[9] G. Cormode. Personal privacy vs population privacy: Learning to attack anonymization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2011.

[10] H. Cramér. *Mathematical methods of statistics*, volume 43. Princeton university press, 1999.

[11] E. Creager, D. Madras, J.-H. Jacobsen, M. Weis, K. Swersky, T. Pitassi, and R. Zemel. Flexibly fair representation learning by disentanglement. In K. Chaudhuri and R. Salakhutdinov, editors, *International Conference on Machine Learning (ICML)*, volume 97, pages 1436–1445, 2019.

[12] I. Damgård, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology—CRYPTO*, pages 643–662, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker. Practical secure decision tree learning in a teletreatment application. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security*, 2014.

[14] S. Dov Gordon, F.-H. Liu, and E. Shi. Constant-round mpc with fairness and guarantee of output delivery. In *Advances in Cryptology—CRYPTO*, pages 63–82, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[15] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 2014.

[16] H. Edwards and A. Storkey. Censoring representations with an adversary. In *International Conference on Learning Representations (ICLR)*, 2 2016.

[17] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *ACM Conference on Computer and Communications Security (CCS)*, pages 1322–1333, 2015.

[18] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov. Property inference attacks on fully connected neural networks using permutation invariant representations. In *ACM Conference on Computer and Communications Security (CCS)*, 2018.

[19] R. Gilad-Bachrach, K. Laine, K. Lauter, P. Rindal, and M. Rosulek. Secure data exchange: A marketplace in the cloud. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*, CCSW'19, page 117–128, New York, NY, USA, 2019. Association for Computing Machinery.

[20] T. Graepel, K. Lauter, and M. Naehrig. Ml confidential: Machine learning on encrypted data. In T. Kwon, M.-K. Lee, and D. Kwon, editors, *Information Security and Cryptology – ICISC 2012*, 2013.

[21] J. Hamm. Preserving privacy of continuous high-dimensional data with minimax filters. In *Artificial Intelligence and Statistics Conference (AISTATS)*, 2015.

[22] X. He, J.-Y. Jia, M. Backes, N. Gong, and Y. Zhang. Stealing links from graph neural networks. In *USENIX Security Symposium*, 2020.

[23] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations (ICLR)*, 2017.

[24] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[25] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *USENIX Security Symposium*, SEC'11, page 35, USA, 2011.

[26] T. Hunt, C. Song, R. Shokri, V. Shmatikov, and E. Witchel. Chiron: Privacy-preserving machine learning as a service. *CoRR*, abs/1803.05961, 2018.

[27] N. Hynes, R. Cheng, and D. Song. Efficient deep learning on multi-source private data. *CoRR*, abs/1807.06689, 2018.

[28] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, 2018.

[29] P. Karnati. Data-in-use protection on IBM cloud using Intel SGX, 2018. https://www.ibm.com/cloud/blog/data-use-protection-ibm-cloud-using-intel-sgx.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[32] B. Knott, S. Venkataraman, A. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten. Crypten: Secure multi-party computation meets machine learning. In *Proceedings of the NeurIPS Workshop on Privacy-Preserving Machine Learning*, 2020.

[33] R. Kohavi. Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 202–207. AAAI Press, 1996.

[34] S. Laur, H. Lipmaa, and T. Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[35] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5–es, 2007.

[36] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 2009.

[37] M. Lichman et al. UCI machine learning repository, 2013.

[38] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, and O. Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*, pages 14584–14597, 2019.

[39] L. Melis, C. Song, E. D. Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In *IEEE Symposium on Security and Privacy (S&P)*, 2019.

[40] P. Mohassel and Y. Zhang. SecureML: a system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy (S&P)*, pages 19–38, 2017.

[41] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *ACM Conference on Computer and Communications Security (CCS)*, page 801–812, New York, NY, USA, 2013. Association for Computing Machinery.

[42] O. Ohrimenko, F. Schuster, C. Fournet, A. Mehta, S. Nowozin, K. Vaswani, and M. Costa. Oblivious

multi-party machine learning on trusted processors. In *USENIX Security Symposium*, 2016.

[43] R. Pass, E. Shi, and F. Tramèr. Formal abstractions for attested execution secure processors. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2017.

[44] K. Pearson. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242, 1895.

[45] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *ACM Symposium on Theory of Computing (STOC)*, STOC '89, pages 73–85, New York, NY, USA, 1989. Association for Computing Machinery.

[46] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang. Updates-leak: Data set inference and reconstruction attacks in online learning. In S. Capkun and F. Roesner, editors, *USENIX Security Symposium*, 2020.

[47] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes. ML-leaks: Model and data independent membership inference attacks and defenses on machine learning models. In *Symposium on Network and Distributed System Security (NDSS)*, 2019.

[48] Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL, Nov. 2020. Microsoft Research, Redmond, WA.

[49] D. J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2020.

[50] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[51] C. Song, T. Ristenpart, and V. Shmatikov. Machine learning models that remember too much. In *ACM Conference on Computer and Communications Security (CCS)*, New York, NY, USA, 2017. Association for Computing Machinery.

[52] C. Song and V. Shmatikov. Auditing data provenance in text-generation models. In *International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 196–206. ACM, 2019.

[53] C. Song and V. Shmatikov. Overlearning reveals sensitive attributes. In *International Conference on Learning Representations (ICLR)*, 2020.

[54] F. Tramer and D. Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations (ICLR)*, 2019.

[55] S. Zanella-Béguelin, L. Wutschitz, S. Tople, V. Rühle, A. Paverd, O. Ohrimenko, B. Köpf, and M. Brockschmidt. Analyzing information leakage of updates to natural language models. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[56] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International Conference on Machine Learning (ICML)*, 2013.

A Attribute Correlation in Datasets

This section provides information on correlation cases for the datasets and attributes in Table 3.

Health Dataset. We measure the correlations between `Gender` or `ClaimsTruncated` and the 133 categorical attributes and 6 numerical attributes by Cramer's V scores and point biserial correlation coefficients, respectively. With `Gender` as the sensitive attribute, we identify 22 categorical attributes that have Cramer's V scores greater than 0.15 and 2 numerical attributes that have point biserial correlation (absolute value) greater than 0.1. The attributes that have the highest Cramer's V are `sp10` (0.218), `noSpecialities` (0.212), `noProviders` (0.208), `noVendors` (0.201). To give a overview of correlations including weak correlation with other attributes, we identify 17 attributes that have Cramer's V scores within the range $[0.1, 0.15]$ and 37 attributes Cramer's V scores within the range $[0.5, 0.1]$. The Cramer's V score between DaysInHospital and `Gender` is 0.09, and thus, we deem them as uncorrelated. With `ClaimsTruncated` as the sensitive attribute, we identify 50 categorical attributes (e.g., `sp1` (0.42), `sp2` (0.51), `pcg1` (0.41), etc.) that have Cramer's V scores greater than 0.15, and 4 numerical attributes that have point biserial correlation (absolute value) greater than 0.1. The score between `DaysInHospital` and `ClaimsTruncated` is 0.13, and we set them uncorrelated.

Adult Dataset. We measure the correlations between `Gender` or `Income` and the 7 categorical attributes and 4

Attributes	Gender	Income
Cramer's V scores		
EducationLevel	0.042	0.326
MaritalStatus	0.466	0.448
Occupation	0.435	0.329
Relationship	0.650	0.454
Race	0.119	0.099
NativeCountry	0.059	0.096
Income	0.217	-
Gender	-	0.217
point biserial correlation coefficients		
Age	0.082	0.229
CapitalGain	0.049	0.221
CapitalLoss	0.047	0.150
HoursPerWeek	0.231	0.230

Table 9: Correlation factors for the Adult dataset.

Range	TotalPctDivorce	Income
$[0.5,1]$	15	34
$[0.4,0.5)$	11	4
$[0.3,0.4)$	31	22
$[0.2,0.3)$	4	12
$[0.1,0.2)$	14	19

Table 10: Correlation factors for Crime dataset.

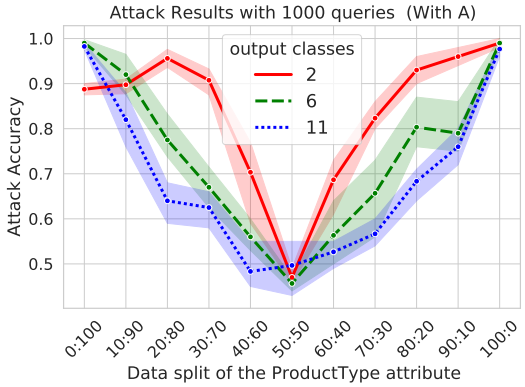

Figure 5: Attack accuracy for the Amazon data with the sensitive attribute ProductType used during training.

numerical attributes by Cramer's V scores and point biserial correlation coefficients, respectively. We list all the correlation factors in Table 9, as X only has 11 attributes. For Gender, we identify 4 categorical attributes that have Cramer's V scores above 0.15 and 1 numerical attribute that has point biserial correlation coefficients above 0.1. The sensitive attribute Income has a high Cramer's V score with 5 categorical attributes and the target variable EducationLevel, as well as high point biserial correlation coefficients with 4 numerical attributes.

Crime Dataset. Since all features are numerical, we measure the Pearson correlation coefficients. Table 10 shows the number of attributes that have the coefficients within a certain range. We use 0.4 as the threshold to determine X'. The target variable CrimesPerPop is correlated with both TotalPctDivorce and Income, with correlation coefficients 0.553 and -0.424, respectively.

Yelp-Health and Amazon Datasets. For Yelp-Health dataset, the point biserial correlation coefficients between Specialty and ReviewRating is 0.009, hence, the scenario corresponds $X \sim A, Y \perp A$. The review text is clearly correlated with the doctor specialty as in Table 4 in [39]. For the Amazon dataset, since the ProductType has 4 levels, we use the ANOVA to test whether the differences between the means of ReviewScore across different product types are statistically significant. The ANOVA p-value is $7.6e-83$. We conjecture that the co-purchasing graph X is also correlated with the ProductType. and hence $X \sim A, Y \sim A$.

B Additional Results

We present attack results for Health, Adult and Crime dataset with smaller size of D_{aux} from Table 2. We show accuracies for both pooled model and the honest party's local model and the utility increase in Table 11. Figure 5 complements results in Section 6.4 on Amazon dataset trained with the sensitive attribute A.

White-box Attack Results Additionally, we performed experiments where the attacker has access to the model parameters, i.e., the white box setting. As for the meta-classifier model, we use a two-layer network with 200 and 50 hidden units and learning rate 0.001. Each meta-classifier is trained based on 100 shadow models using Adam optimizer. Here, the meta-classifier takes as input model parameters as opposed to model inferences. Table 13 shows the results. For logistic regression, the results are similar to those in Table 5 for the black-box setting. However, the attack accuracy for neural networks

(MLP) reduces significantly. This was noted in the work by [18]. One reason is that it is hard for a naive meta-classifier to learn the structure of equivalent symmetrical weights of neural networks. Indeed, one of their contributions is a technique for identifying this symmetry.

Datasets (Output Classes)	Sensitive Attribute	Pooled Accuracy	Local Accuracy	Utility Increase
Health (2)	Gender	85.22%	84.64%	.58%
	ClaimsTruncated	76.63%	73.56%	3.07%
Adult (4)	Gender	73.23%	72.46%	.76%
	Income	71.14%	70.43%	.71%
Crime (3)	TotalPctDivorce	74.52%	72.35%	2.17%
	Income	72.81%	71.30%	1.51%
Yelp-Health (2)	Specialty	86.28%	80.38%	5.90%
Amazon (2)	ProductType	76.80%	76.28%	.62%
Amazon (6)	ProductType	45.92%	42.50%	3.42%
Amazon (11)	ProductType	27.94%	26.09%	1.85%

Table 11: Test accuracies of the model trained on pooled dataset and the model trained only on honest party's data. The split in the honest party is 33 : 67 based on the sensitive attribute.

Datasets (Classes) Model Type	Attack Accuracy A	Attack Accuracy $\bar{A}$	A / $\bar{A}$	# X'
Health (2) MLP	.59	.55	Gender	24/139
	. 67	. 56	ClaimsTruncated	54/139
Adult (4) LR	.73	.76	Gender	5/11
	.84	.91	Income	9/11
Crime (3) MLP	.60	.56	TotalPctDivorce	26/98
	.62	.60	Income	38/98

Table 12: Multi-Party Setting: Black-box attack accuracy for predicting the value of the distribution of sensitive variable A in the dataset of P_{honest}. We use smaller size of D_{aux} listed in Table 2, while all other settings are the same as in Table 4.

Model	Logistic Regression				Neural Network			
Datasets	Adult		Health		Adult		Health	
Synthetic Variable	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$
$X \sim A, Y \sim A$	.90	.94	.85	.97	.54	.49	.65	.61
$X \perp A, Y \sim A$	.95	.93	.81	.80	.57	.53	.63	.56
$X \sim A, Y \perp A$	.54	.53	.50	.53	.56	.53	.54	.51
$X \sim A, Y \perp A$ (R)	.75	.63	.76	.68	.55	.50	.55	.45

Table 13: White-box attack accuracy for predicting whether the values of sensitive variable A in D_{honest}, the data of the honest party, are predominantly < 5 or > 5. The attack accuracy is evaluated on 100 D_{honest} datasets: half with 33:67 and half with 67:33 split and D_{adv} has 33:67 split. A synthetic correlation with A is added to the variables X and Y depending on the specific case. R corresponds to the setting where only 3 attributes are used for training instead of all data. Attack accuracy based on a random guess is 0.5.