Data Privacy and Computation Integrity in Machine Learning Scenarios: Some Issues and Approaches

Sabrina De Capitani di Vimercati[1], Sara Foresti[1], Stefano Paraboschi[2], Pierangela Samarati[1]
[1]Università degli Studi di Milano, Italy – Email: *firstname.lastname*@unimi.it
[2]Università degli Studi di Bergamo, Italy – Email: parabosc@unibg.it

Abstract

The increasing popularity of artificial intelligence and the wide adoption of machine learning has raised new privacy and security issues. Indeed, the massive amount of data on which AI and ML rely can include sensitive or company-confidential information. To enable the wide distribution of trained models as a service to interested final users, as well as the adoption of external parties for training models (e.g., to leverage the flexibility and economies of scale offered by cloud solutions), data need to be adequately protected. Also, the correctness and completeness of computations (e.g., for running or training models) delegated to external parties need to be guaranteed. In this paper, we consider some privacy and computation integrity issues arising when data and machine learning models or tasks are shared with external parties, together with possible solutions and research directions.

1 Introduction

The growing adoption of machine learning in our daily lives brings several advantages, such as better decision making and faster problem solving. Machine learning applications rely on the availability of massive amounts of data, together with powerful and efficient computational infrastructures and services used to train models and support increasingly complex analytics tasks. In many scenarios, these data can be provided by different owners, and often contain personal, sensitive, or company-confidential information that cannot be freely shared without proper protection. It is therefore important to ensure that both data confidentiality and privacy of data subjects (i.e., the entities to whom the data refer, such as users or organizations) are adequately protected throughout these data-sharing and analysis processes. Furthermore, privacy risks are not limited to data. When trained models are shared with external parties (e.g., for testing the performance of the model), they may also leak sensitive information about the data used for training. Although the protection of the confidentiality and privacy of data, models, and users is extremely important, this is not sufficient to guarantee that machine learning applications operate correctly, especially when the data sharing and preparation, model training, or inference phase of the machine learning life cycle is delegated to external providers.

Existing solutions for protecting confidentiality when sharing training datasets with external parties (e.g., [6, 7]) are based on cryptographic techniques such as homomorphic encryption that, together with secure multi-party computation, can be used to protect the dataset while used to train a machine learning model. More recent solutions combine cryptographic techniques with, for example, differential privacy, hardware-based trusted execution environments, or secure aggregation protocols used in federated learning (e.g., [12, 27]). Alternative solutions aim to avoid sharing real data by releasing synthetic data, that is, artificially generated data that preserve the statistical properties of the original data (e.g., [8]). While promising, these solutions based on synthetic data do not guarantee full protection (e.g., [2]). Training data can also be subject to attacks aimed at degrading the resulting model or altering its

behavior. One of the well-known threats is data poisoning that happens when an adversary intentionally injects, modifies, or manipulates data in the training dataset. Existing solutions aim at detecting and removing poisoning samples or to make learning algorithms more robust (e.g., [5, 25]). The privacy of data subjects represented in training datasets released to third parties is instead protected through the application of anonymization techniques such as k-anonymity [23] or differential privacy [14] and their variations. Although these techniques help to preserve privacy, they often reduce the accuracy of the resulting machine learning models (e.g., [24]). One of the main challenges is therefore to balance privacy and utility with respect to the analytical task for which the data will be used (e.g., [4, 9, 17]). Public or outsourced models are instead vulnerable to different types of inference attacks, such as membership inference, attribute inference, model inversion, and property inferences as well as model extraction attacks, which allow adversaries to reconstruct an equivalent model (i.e., a model that provides nearly the same results as the original model) and exploit it for further privacy violations (e.g., [22]). Periodic model updates (e.g., incremental learning) can also disclose changes in the underlying data such as the addition or removal of data subjects in unlearning settings. In addition to protecting the privacy of data subjects in the training dataset, also the sensitive information that users provide when using the model need to be protected. Indeed, a privacy-friendly model should not require users to disclose (either releasing directly or exposing indirectly) sensitive information. All these privacy and confidentiality issues become particularly significant in emerging collaborative learning scenarios (e.g., federated learning), where data are not centralized but remain distributed among multiple clients (e.g., [1, 15, 26]). In this context, the clients exchange only model updates (e.g., the gradients or parameter modifications computed during the training performed by the participants). These updates, however, can be exploited for inferring, for example, information about the underlying data (e.g., [12, 27]).

While one may assume an overall proper behavior, the use of external providers in the machine learning life cycle is clearly vulnerable to possible misbehavior by them, which can either be sloppy in their operation, or - even worse - intentionally misbehave, and therefore opportunistic in their responses. Since users as well as organizations are increasing their dependency on data and on the results of machine learning and data analytics tasks for their daily operations, data and computation integrity is paramount. Guaranteeing integrity means that techniques should be adopted to discover possible misbehavior that tampered with data or results of computations. When training datasets are stored and managed by external providers, it is necessary to ensure that no unauthorized modification occur (e.g., data poisoning). When prediction computations are outsourced, users should be able to verify that the results returned by the external provider are both correct and complete. Incorrect or incomplete results may arise from laziness, misconfiguration, or malicious tampering. Ensuring integrity therefore requires solutions that allow data owners and users to assess the integrity of every computation performed by the service provider (e.g., [10]).

To concretely analyze and discuss possible solutions to the confidentiality, privacy, and integrity issues discussed above, we focus on the classification problem since it is at the basis of a wide range of real-world applications (e.g., medical diagnosis, fraud and anomaly detection in financial systems, and spam filtering, just to name a few). Data classification is the process of training a machine learning model that is used for predicting the correct label of a given input data. The model learns patterns from a labeled training dataset (i.e., a dataset where each piece of information is associated with the correct label), and then uses this model to make predictions on new, unseen data. In the remainder of this paper, we address some data privacy, confidentiality, and integrity issues that can arise in the machine learning life cycle. Specifically, at data sharing and preparation time, we focus on the problem of protecting the privacy of data subjects while preserving the utility of the data for the downstream classification task (Section 2). We describe TA_DA, a target-aware data anonymization technique that allows data owners to contribute with their data to a classification task, anonymizing their data while maintaining utility

	Gender	Age	Smoker	Disease	Risk
t_1	F	65	former	emphysema	yes
t_2	M	54	never	cirrhosis	no
t_3	F	61	current	bronchitis	yes
t_4	M	72	never	cirrhosis	no
t_5	F	35	former	arthritis	no
t_6	M	68	former	bronchitis	yes
t_7	M	64	never	diabetes	no
t_8	F	15	never	asthma	no
t_9	F	39	never	osteoporosis	no
t_{10}	M	29	never	celiac	no
t_{11}	M	33	never	arthritis	no
t_{12}	F	54	current	bronchitis	yes
t_{13}	F	75	former	bronchitis	yes

Figure 1: An example of dataset including information about a sample of people

for the downstream task. At the model training time, we focus on the problem of minimizing leakage of sensitive information from the training data used for building a model made publicly accessible as well as from data of users interacting with the model. We describe PriSM, a solution for generating a privacy-friendly classifier that requires neither sensitive information nor information correlated with it for training a high accuracy classifier (Section 3). Finally, we address the problem of ensuring data and computation integrity by enabling the detection of incorrect or incomplete results returned by external providers (Section 4). We describe sentinels and twins that span all three phases of the machine learning data life cycle. In the following, we assume that the training dataset is a relational table R characterized by a set $\{a_1, \ldots, a_n, s, l\}$ of attributes, where s is sensitive and l is the label attribute. Figure 1 illustrates a running example showing a dataset collecting information about chronic disease and smoking habits of a sample of people. In particular, the relation keeps track, for each subject, of their gender (attribute `Gender`), age (attribute `Age`), smoking habits (attribute `Smoker` with values current, former, or never), chronic disease (attribute `Disease`), and risk of suffering from a Chronic Obstructive Pulmonary Disease (binary attribute `Risk`). Attribute `Disease` is considered sensitive, while `Risk` is the label attribute.

2 Target-aware anonymization

When data owners contribute their datasets to collaborative data analytics and machine learning tasks for training more comprehensive and accurate models, their datasets could be revealed to both the party in charge of the training phase and to other contributors. To provide protection to such datasets, solutions based on noise injection (e.g., differential privacy [14]) or on generalization and suppression (e.g., k-anonymity [23]) can be used. While noise injection may significantly distort the data and can compromise the quality of the downstream analysis, generalization preserves the truthfulness of the data. However, generalization needs to be applied with care as it inevitably causes information loss, which can reduce the accuracy of downstream analytical task, especially when applied on attributes that are good predictors of the classification label. Based on this observation, TA_DA (Target-Aware Data Anonymization) [4, 9] relies on generalization for protecting (i.e., anonymizing with k-anonymity [23] and ℓ-diversity [19]) data, while preserving as much as possible the utility of the protected data for the downstream collaborative analytics.

Anonymization with k-anonymity and ℓ-diversity operates by generalizing the values of the quasi-identifier attributes (i.e., attributes that, in combination, could be used to reconstruct the identity of data subjects through linking with external data sources) in the training dataset R in such a way that

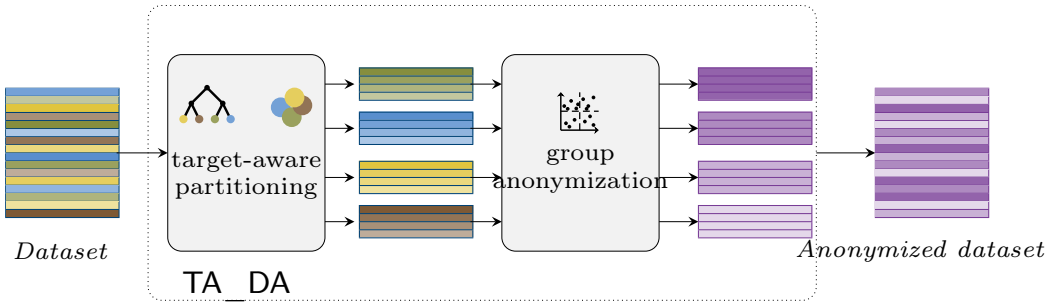

Figure 2: Overall working of TA_DA

each combination of generalized quasi-identifier values appears at least k times in the released dataset, and each group of tuples sharing the same generalized quasi-identifier contains at least ℓ well-represented (e.g., distinct) values for the sensitive attribute. Given a dataset R and privacy parameters k and ℓ, different generalization strategies can enforce k-anonymity and ℓ-diversity. Among all the solutions guaranteeing k-anonymity and ℓ-diversity, TA_DA aims at computing the one that preserves as much as possible those attributes on which the label attribute depend more, favoring generalization on the other attributes of the quasi-identifier. For instance, with reference to the example in Figure 1 where the quasi-identifier includes attributes `Gender` and `Age`, TA_DA would prefer a solution generalizing attribute `Gender` if `Risk` highly depends on `Age`. To achieve this goal, TA_DA operates in two steps (see Figure 2): *target-aware partitioning*, which partitions the tuples in R in groups according to the downstream classification task while satisfying k-anonymity and ℓ-diversity; and *group anonymization*, which anonymizes each partition generated by the previous step.

Target-aware partitioning. The goal of this phase is to partition tuples in the training dataset in such a way that each group includes tuples with similar values for the attributes in the quasi-identifier that are the best predictors of the label attribute (i.e., the attributes on which the label attribute value depends more). The intuition is that keeping in the same group tuples with similar values for an attribute limits the amount of generalization needed on it for achieving k-anonymity. The problem is that the best predictors of the label attribute are not known and must be learned from the dataset itself. TA_DA learns predictors and partitions the dataset according to their values by building a (k, ℓ)-compliant decision tree for the label attribute. A (k, ℓ)-compliant decision tree is a decision tree [16] having leaf nodes including at least k tuples with at least ℓ different values for the sensitive attribute. TA_DA builds a (k, ℓ)-compliant decision tree considering only quasi-identifying attributes for split operations. Other attributes, including the sensitive attribute are not considered. The sensitive attribute is not considered because there is the need to ensure diversity of its values in each generalized group. Other attributes are not affected by generalization, and therefore the anonymization process has no impact on them. Starting from the original training dataset, which corresponds to the root node, TA_DA recursively splits the dataset according to the values of a quasi-identifier attribute, generating children nodes. The split attribute, and its values used for partitioning tuples, is selected to maximize the quality of the decision tree (e.g., maximize information gain to have uniform labels in the leaf nodes), provided the resulting nodes guarantee that the tree is (k, ℓ)-compliant. The recursive split terminates when a stopping condition is satisfied (e.g., uniform values for the label in leaf nodes) or when any split would result in a decision tree that is not (k, ℓ)-compliant. By construction, the split attributes are those on which the label attribute depends more and each group contains tuples with similar values for these attributes, since the tuples in a node satisfy the same decision rule (i.e., the same set of conditions defined over the split attributes along the path from the root to the node). Figure 3 illustrates an

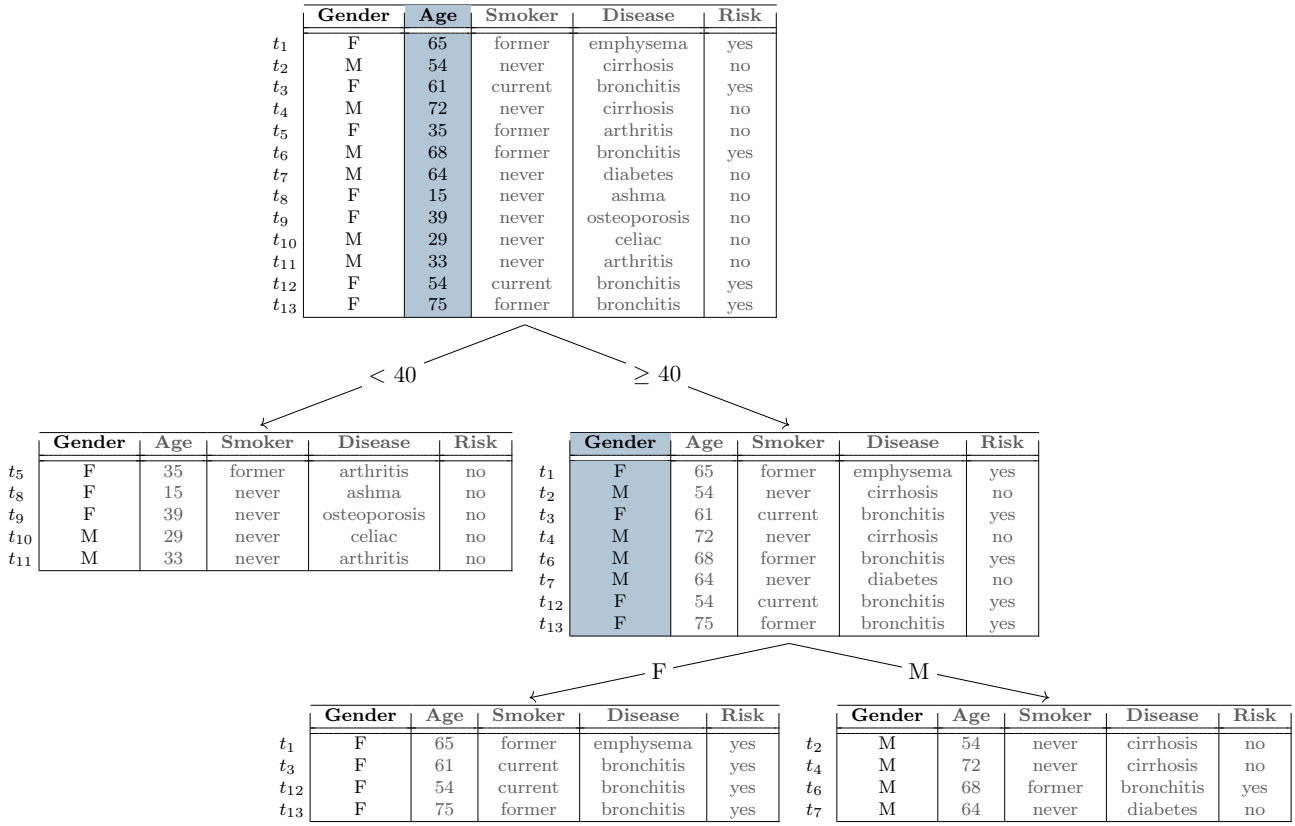

Figure 3: An example of (2,2)-compliant decision tree built over the relation in Figure 1

example of (2,2)-compliant decision tree built over the relation in Figure 1 and considering attributes `Gender` and `Age` in the quasi-identifier. The root node corresponds to the whole dataset that is split over attribute `Age` on condition < 40 (≥ 40, resp.), obtaining two partitions. For the first partition of tuples, representing patients with `Age`< 40, the process stops since all the tuples in the group have the same value for label attribute `Risk`. For the second, representing patients with age `Age`≥ 40, there is a further split on attribute `Gender`. The partitioning process then stops, since there are no other attributes in the quasi-identifier that could be used for further splitting the two leaf nodes. In the figure, attributes with gray values are those that cannot be used for splitting, and the attribute with a light blue background (gray in b/w printout) is, at each level, the one on which a further split is performed. The resulting decision tree is $(2, 2)$-compliant since each leaf node includes at least two tuples and has at least two different values for the sensitive attribute `Disease`.

Group anonymization. The goal of this phase is to independently anonymize each group of tuples corresponding to the leaf nodes of the (k, ℓ)-compliant decision tree computed in the previous phase. Intuitively, by generalizing each leaf node in a (k, ℓ)-compliant decision tree, the resulting dataset would be k-anonymous and ℓ-diverse. While any anonymization algorithm can be used for this second phase, TA_DA relies on Mondrian [18]. Mondrian has the advantage of leveraging a multi-dimensional spatial representation of the dataset similarly to the approach of construction of the (k, ℓ)-compliant decision tree. Each tuple is modeled as a point in a multi-dimensional space having a dimension for each attribute in the quasi-identifier. Mondrian recursively splits the multi-dimensional space in two partitions, in such a way that each partition includes at least k tuples with at least ℓ different values for the sensitive attribute. This process terminates when any further split would generate subspaces with less than k

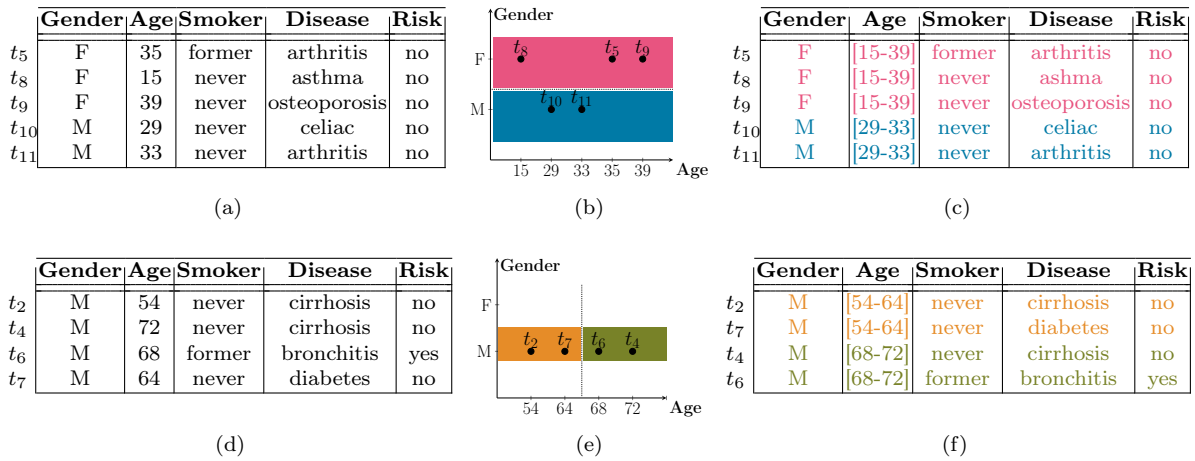

Figure 4: Anonymization of the first (a-c) and third (d-f) leaf nodes of the (2,2)-compliant decision tree shown in Figure 3

	Gender	Age	Smoker	Disease	Risk
t_1	F	[54-75]	former	emphysema	yes
t_3	F	[54-75]	current	bronchitis	yes
t_{12}	F	[54-75]	current	bronchitis	yes
t_{13}	F	[54-75]	former	bronchitis	yes
t_2	M	[54-64]	never	cirrhosis	no
t_7	M	[54-64]	never	diabetes	no
t_4	M	[68-72]	never	cirrhosis	no
t_6	M	[68-72]	former	bronchitis	yes
t_5	F	[15-39]	former	arthritis	no
t_8	F	[15-39]	never	ashma	no
t_9	F	[15-39]	never	osteoporosis	no
t_{10}	M	[29-33]	never	celiac	no
t_{11}	M	[29-33]	never	arthritis	no

Figure 5: An example of (2,2)-anonymous version of the relation in Figure 1

tuples or less than ℓ different values for the sensitive attribute. For all the tuples in each sub-space, the values of the attributes in the quasi-identifier are generalized to the same combination of values, thus guaranteeing that all the generalized tuples in the subspace are indistinguishable according to the quasi-identifier. The anonymized version of the dataset R is then obtained through the union of the anonymized groups of tuples corresponding to the leaves of the (k, l)-compliant decision tree. For instance, with reference to the (2,2)-compliant decision tree in Figure 3, the first leaf node includes five tuples and Mondrian performs a split over attribute `Gender` producing two groups on which attribute `Age` is then generalized. Similarly, for the third leaf node, Mondrian performs a split on attribute `Age`, producing two groups of tuples, in which attribute `Age` is then generalized. For the second leaf, Mondrian cannot perform a split on the attribute `Age` (which is the only attribute with different values for the tuples in the group), since the result would violate ℓ-diversity. Figure 4 illustrates the tuples in the first and third leaf nodes of the tree, their spatial representation, and the corresponding (2,2)-anonymous version. Figure 5 illustrates the resulting generalized table after the partitioning and the group anonymization described.

Summary and other issues. Combining target-aware partitioning with group anonymization, TA_DA enforces anonymization taking into consideration the downstream classification task. Indeed, TA_DA identifies and limits the amount of generalization on quasi-identifier attributes on which the label

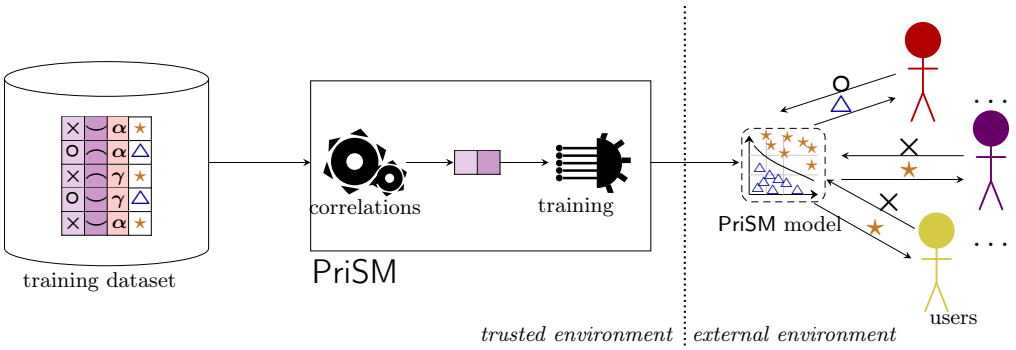

Figure 6: Overall working of PriSM

attribute depends more, thus producing a generalized table that satisfies k-anonymity and ℓ-diversity and that also preserves utility for data classification. Hence, a classifier trained over data anonymized using TA_DA can provide higher accuracy compared to a classifier trained over data anonymized with a classical (non target-aware) anonymization algorithm, as also demonstrated by the experimental analysis in [4]. Note that, besides classification, TA_DA can be used for *clustering* (a non-supervised learning task [21]) as downstream analytics. To this purpose, the target-aware partitioning phase needs to be revised to group tuples according to the same strategy adopted by clustering (e.g., minimize intra-cluster distance or similarity), while at the same time guaranteeing k-anonymity and ℓ-diversity. Besides the consideration of different downstream analytics, TA_DA could be extended to consider different privacy definitions, like differential privacy, with the goal of limiting noise and driving its injection based on the downstream machine learning task.

3 Privacy-friendly training

Allowing users (beyond the data or model owners) to rely on machine learning models trained using large and valuable data collections represents a clear advantage, especially for those who would not have the resources for collecting data and/or training models. However, this increased accessibility to machine learning models also increases the risk of exposing sensitive data in the training dataset. As a matter of fact, even if the training dataset is not released, its content might partially be exposed if the model is publicly released, or even through its simple use (e.g., observing the behavior of the model and its output). To address this challenge, we illustrate PriSM (Privacy-friendly Support vector Machine) [3], which aims at building a privacy-preserving classifier that guarantees protection of the sensitive attributes in the training dataset. For enabling final users to take advantage of machine learning models, while protecting the data used for training, PriSM removes from the training dataset not only sensitive attributes but also those (combination of) attributes that could indirectly expose them (e.g., through inferences). This strategy has a twofold advantage as it protects the sensitive information in the training dataset as well as the privacy of the users, who would not need to disclose sensitive information for obtaining predictions from the model. Intuitively, if an attribute is not considered in training, it will not be used (and therefore asked) for prediction. Given attributes (or sets thereof) that are considered sensitive, PriSM operates in two steps (Figure 6): it first identifies (sets of) other attributes that could leak the sensitive attribute values, and then trains the classifier in such a way to protect the sensitive attribute while limiting the impact on the accuracy of the model.

Sensitive correlation discovery. The goal of the first phase is to identify sets of attributes that could

leak the values of the sensitive attribute. A *sensitive correlation* is a set X of non-label attributes (i.e., $X \subset R \setminus \{l\}$, with l the label attribute) that can be used to predict the sensitive attribute $\in R$ (i.e., the values of s can be inferred from the values of the attributes in X and knowing X could lead to knowledge also of s). For instance, in the dataset in Figure 1, `Age` and `Smoker` can be used to infer (or reduce the uncertainty about) sensitive attribute `Disease`. Intuitively, a correlation between a set X of attributes and a sensitive attribute can be assessed by training a classifier that uses the sensitive attribute as target. A set X is a sensitive correlation if the classifier achieves a prediction accuracy for s higher than a predefined threshold. Since this approach is impractical in real-world scenarios, PriSM uses a correlation coefficient as a proxy for such evaluation. If the coefficient is greater than a given threshold, the correlation is considered sensitive and must be protected. Aimed at limiting information loss and at maximizing classification precision, PriSM distinguishes critical values from non-critical values in the domain of the sensitive attribute (e.g., a critical value for the attribute `Disease` could be emphysema, while a non-critical one could be flu), and considers sensitive only those correlations that can be exploited to infer critical values. As a matter of fact, not all values of the sensitive attribute are critical, and focusing only on the critical ones allows PriSM to be more precise in identifying sensitive correlations, excluding from consideration those correlations that are not considered sensitive, and hence possibly improve accuracy in classification. PriSM verifies the ability to leverage non-sensitive attributes to predict each critical value individually (assuming each value to represent a different class) as well as the set of all critical values together (assuming a binary classifier distinguishing critical from non-critical values), to better capture different sensitive correlations. Indeed, correlations discovered considering each critical value singularly taken may not be discovered when the critical values are considered together and vice versa. To identify attribute sets that could leak critical values of the sensitive attribute, PriSM leverages the natural monotonicity of sensitive correlations. If a subset X of attributes leaks the sensitive attribute, then any superset of X should be excluded from training, as it could expose the sensitive attribute. PriSM therefore examines subsets of non-sensitive (and non-label) attributes in R ordered by increasing cardinality, omitting any superset of already identified sensitive correlations. As an example, consider the dataset in Figure 1. PriSM first evaluates the correlation between each single attribute and the sensitive attribute `Disease`. Suppose that `Smoker` permits to infer critical values of `Disease`, while `Age` and `Gender` do not. In the second iteration, PriSM then checks only the correlation between the pair of attributes `Age` and `Gender` since all pairs including `Smoker`, although sensitive, are implicitly represented by the singleton sensitive correlation involving `Smoker`.

Classifier training. The goal of this second phase is to train a classifier that excludes from training the sensitive attribute as well as any sensitive correlation that could leak its critical values, while maximizing the predictive accuracy of the model. To this purpose, PriSM selects a subset of attributes in R to be used for training while ensuring that, for each sensitive correlation, at least one attribute involved in that correlation is excluded. PriSM extends classical Support Vector Machine (SVM) classifier, proposing an approach that aims at minimizing misclassification while excluding the sensitive attribute and at least one attribute from each sensitive correlation identified in the first phase. In addition, PriSM follows a parsimony principle, aimed at using at most a predefined number of attributes as predictors. This constraint has two advantages: it reduces the disclosure of unnecessary attributes and improves the efficiency of the training process.

Summary and other issues. Excluding the sensitive attribute and sensitive correlations from training, PriSM provides a privacy-preserving classifier able to protect the privacy of data subjects represented in the training dataset, as well as final users who do not need to release their potentially sensitive information for obtaining a prediction. While imposing constraints on the attributes to be used for training, PriSM maintains high accuracy in predicting the correct value for the label attribute target of the classification task, as demonstrated by experimental results on both real-world and synthetic

datasets [3]. Besides maintaining high accuracy in classification, PriSM is also characterized by a limited performance overhead in training a privacy-preserving classifier compared to a traditional classifier. It is interesting to note that, although PriSM has been specifically designed to define privacy-preserving SVM classifiers, the identification of sensitive correlations is independent of the underlying classifier. The first phase of the approach identifies sensitive correlations analyzing the dataset, independently from the specific classifier for which such dataset will be used. The first phase of the approach can then be used in combination with any classification model (e.g., non-linear or non-binary classifiers), as well as with any other machine learning model. The working of the classification/machine learning approach then needs to be revised for excluding from training the sensitive attribute as well as sensitive correlation maintaining, at the same time, high accuracy of prediction results.

4 Probabilistic integrity controls

Delegating the training or deployment of a machine learning model to an external party (e.g., a computational provider or a set of workers), for cost efficiency or to enable access to a wider set of final users, introduces potential risks to the correctness and completeness of the resulting computations. Indeed, workers may be lazy or malicious and partially or entirely omit the computation, returning an empty or randomly generated result. Lazy workers omit computations for economic reasons, with the goal of saving on computational resources elaborating a subset of the tuples in the input dataset while being payed for the elaboration of the whole dataset. Malicious workers instead intentionally misbehave, partially omitting the computation or returning incorrect result on purpose (e.g., to influence decision making processes). Integrity verification techniques verify the *correctness*, *completeness*, and *freshness* of computation results. Correctness refers to the verification that the computation has been executed in accordance with the algorithm defined by the data owner. Completeness ensures that the computation has been executed over the entire dataset, with no tuples omitted. Freshness consists of verifying that the computation has been performed over the most recent version of the dataset. Integrity verification techniques fall into two main categories: *deterministic* approaches, providing integrity guarantees with full confidence, and *probabilistic* approaches, providing such guarantees with a certain degree of confidence [11].

Deterministic techniques associate each computation result with a *Verification Object* (VO), which enables the recipient to verify the integrity of the returned result. The VO is constructed using an authenticated data structure (e.g., a Merkle hash tree or a skip list [13, 20]) built by the data owner over the dataset. The recipient of a computation then uses the VO, possibly together with information provided by the data owner, to verify the integrity of the result. Deterministic techniques provide full confidence in the integrity of computation results defined over the attribute(s) on which the authenticated data structure is built, but they do not offer guarantees for computations involving other attributes.

Probabilistic techniques provide probabilistic integrity guarantees, meaning that they provide, with a probabilistic degree of confidence, guarantee of a computation result to satisfy integrity property when passing integrity checks. These techniques are typically based on the injection of *control tuples* into the dataset. Their main advantage with respect to deterministic techniques is their broader applicability, as they are not restricted to computations defined over specific attributes. However, the integrity guarantee remains probabilistic because an integrity violation can be detected only if it affects the injected control tuples.

Probabilistic techniques are based on the injection of fake tuples (*sentinels*) with known results, or on the replication of a subset of the tuples (*twins*) in the dataset. A sentinel is a tuple generated by the client with known computation result: a result different from the expected one signals an integrity violation. Sentinels should be indistinguishable from genuine tuples, to prevent workers from selectively

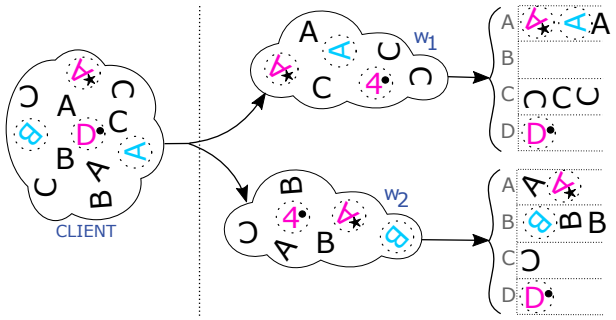

Figure 7: Sentinels and twins for integrity verification

processing them while omitting non-sentinel tuples. A twin is a tuple assigned to multiple workers: inconsistency in the results of twins signals an integrity violation. Twins are allocated to different workers to prevent them from detecting control tuples and selectively omitting their processing, while returning a coordinated result, to bypass integrity verification checks.

To assess the effectiveness of the combined adoption of sentinels and twins, several aspects must be investigated: How should sentinels be distributed among classes? Is it preferable to employ more twin pairs or more replicas of the same job? How many sentinels and twins are required to achieve a given integrity guarantee? In the following, we provide an answer to these questions, based on the analysis in [10]. Figure 7 illustrates the considered scenario, characterized by a client distributing classification jobs (i.e., the computation of the class associated with each tuple in the dataset) to a set of workers. In the figure, control jobs are circled, with twin pairs characterized by the same symbol. When omitting a classification job, a worker can return a random result (i.e., a randomly extracted a class), or opportunistically select the class that maximizes the probability of the omission to go undetected. In the following, we will refer to a worker omitting jobs as a lazy worker and, taking a safe approach, assume that lazy workers return an opportunistic answer for omitted jobs. We do not distinguish between lazy and malicious workers as the impact on integrity guarantees provided by sentinels and twins is the same for any omission, independently from the reason why the classification job has been omitted.

Distribution of sentinels in classes. To maximize the effectiveness of sentinels, it is important to properly tune their distribution in classes and their allocation to workers. The client has full control over how sentinels are distributed in the classes, since these jobs are generated ad hoc. Although the client may choose to take the distribution of the original dataset into account when distributing sentinels in classes, their effectiveness is maximized when they are uniformly distributed, that is, when injecting the same number of sentinels in each class. Otherwise, a lazy worker could exploit knowledge of the input data distribution in its opportunistic behavior. If sentinels are distributed according to genuine data distribution, by returning the most frequent class for each omitted classification job a lazy worker would correctly guess the sentinel's expected class for a high percentage (frequency of the most frequent class) of sentinels. Consider, as an example, a data distribution following Zipf's law with $\alpha = 1$ (Figure 8(a)), the injection of sentinels according to the same distribution as the data would result in 48% of them with label c_1 (see Figure 8(b), reporting sentinels added to each class on top of the corresponding bar). A lazy worker returning c_1 as a default value for omitted jobs would correctly guess almost half of the sentinels. Similarly, if the client distributes sentinels according to the normalized inverse of the input distribution, a lazy worker could simply return the least frequent class (D in the above example, see Figure 8(c)) for every omitted job, again correctly guessing the correct result of a high percentage of sentinels (48% in the example above). In contrast, when sentinels are uniformly distributed, lazy workers cannot leverage any knowledge of class frequencies to reduce the risk of the client detecting omissions.

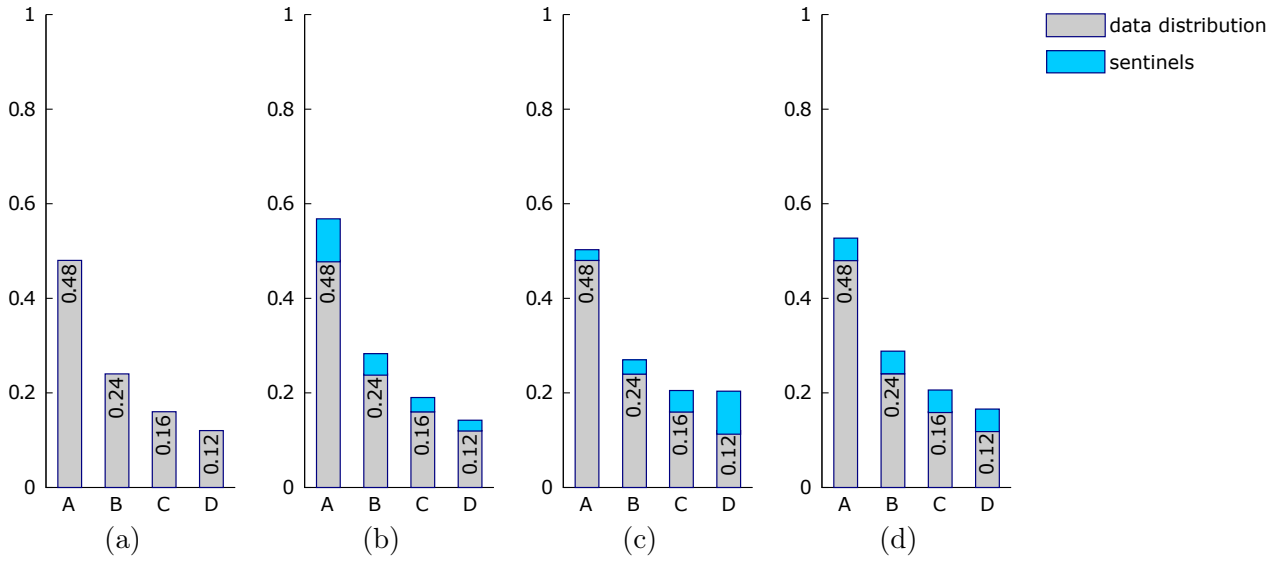

Figure 8: An example of probability mass function following a Zipf's law with $\alpha = 1$ (a) and sentinels distributed according to the data distribution (b), the normalized inverse data distribution (c), and uniformly (d)

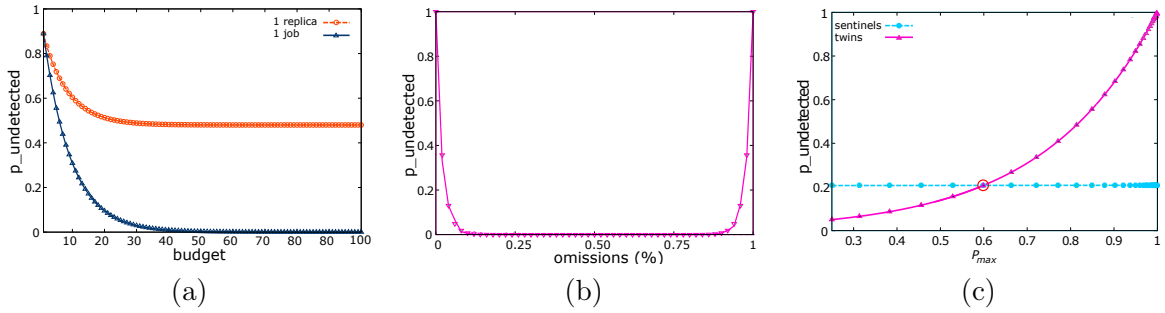

Figure 9: Probability that omissions go undetected using only twins and varying the number of twin pairs and the number of replicas (a), using only twins and varying the percentage of omissions (b), using sentinels and twins varying the frequency of the most frequent class (c)

With reference to the example above, a uniform distribution would result in distributing 25% of the sentinels in each class (Figure 8(d)), with therefore 25% probability for the lazy worker of correctly guessing omitted sentinels, independently from the data distribution.

Twin pairs vs replicas. Each job can be replicated multiple times, with each replica allocated to a different worker. Considering a fixed budget that the client can spend for integrity verification, it is worth noting that managing twins in pairs is substantially more effective, in terms of probability of detecting omission, than producing many replicas of the same classification job. In other words, it is preferable to generate a single additional copy for a larger number of different classification jobs rather than replicating one job many times. This is mainly due to the fact that the client cannot control the distribution of twins into classes, as jobs to be twinned are randomly extracted from the input dataset. Twins are naturally distributed according to the same distribution as the input dataset. Assuming lazy workers to return, for omitted jobs, the most frequent class (which minimizes classification error), the

probability of a lazy worker of correctly guessing all the replicas of a classification job is the probability of such a class. Therefore, investing all the budget replicating a single job is less effective than replicating different jobs. Indeed, especially if the original data distribution is very unbalanced, the probability for lazy workers of correctly guessing omitted replicas is very high. This behavior is confirmed in Figure 9(a), which reports the probability that an omission (50 omitted jobs by 49 workers out of 100 lazy workers) goes undetected as a function of either the number of distinct replicated jobs (continuous blue curve with triangles) or the number of replicas of a single job (dashed orange curve with circles). As visible from the figure, the probability of undetected omissions rapidly approaches zero as the number of replicated jobs increases, whereas it remains significantly higher (never going below the frequency of the most frequent class) when increasing the number of replicas for a single replicated job.

Sentinels and twins balance. Given a budget (in terms of additional jobs) for integrity verification, it is necessary to balance the adoption of sentinels and twins aiming at maximizing the probability of detecting omissions by workers. While in general twins are roughly twice as effective as sentinels (with one additional job, the client controls the behavior or two workers), a few sentinels are necessary to avoid extreme omissions to go undetected. As a matter of fact, the effectiveness of twins decreases in case of extreme omissions. Intuitively, if both the workers omitting the two replicas of a same job return the most frequent class, the omission goes undetected (the results are coherent, even if wrong). Then, if a large majority of workers omit their jobs, the probability of passing twins control is high, even when injecting a large number of twins. Figure 9(b) illustrates the probability of an omission to go undetected varying the percentage of omitted jobs, assuming to twin 5% of the jobs and that all workers are lazy. As visible from the figure, the probability quickly decreases when the percentage of omissions becomes non negligible, but it increases when the percentage of omissions becomes close to 1. Therefore, a few sentinels are always necessary to prevent extreme omissions to go undetected. The preference between the adoption of sentinels only or twins (with a few sentinels), based on their effectiveness, depends on the distribution in classes of the input dataset. More specifically, sentinels are preferred if $P_{max} > 0.5 \cdot (1 + c)$, with P_{max} the probability of the most frequent class and c the number of classes. If the data distribution is highly unbalanced, sentinels are more effective as the most effective strategy for the lazy worker (i.e., return the most frequent class for omitted jobs) implies a correct guess of the job result. Note that this threshold provides a nice and easy to use indication for the client. Once defined which between twins and sentinels is better to use, the client can size the number of control jobs based on either the client's budget or the threshold of probability of omissions going undetected. Figure 9(c) illustrates the probability of an omission to go undetected when using only sentinels (dashed blue line with bullets) and when using only twins (continuous pink line with triangles), varying the probability P_{max} of the most frequent class, assuming 10 workers, 4 of which are lazy and considering a probability of omitting jobs of 20%. As visible from the figure, the probability of omissions to go undetected is lower when using twins if P_{max} is small, it is lower when using sentinels when P_{max} grows.

Summary and other issues. While the results illustrated above permit to reason about the distribution of sentinels and twins among workers to verify their behavior, they are based on the assumption that (lazy) workers do not collude to maximize their probability of going undetected when omitting classification jobs. Colluding workers can indeed identify twin pairs assigned to them and hence return a coherent result, thus passing twin integrity check. In scenarios where workers can collude, twins need to be distributed in such a way to minimize the probability that a job and its replica are both allocated to colluding workers. Alternatively, twin jobs should be generated in such a way to make twin pairs unrecognizable as such. We also note that sentinels and twins have been primarily designed to verify precise computations (i.e., deterministic jobs, like the run of an algorithm), while real world scenarios need to account for computations characterized by an acceptable (limited) amount of errors.

5 Conclusions

We discussed some privacy and integrity issues that arise when external, potentially untrusted, parties are involved in machine learning tasks (e.g., for training or deploying a machine learning model). In fact, when releasing data to external parties for training models or releasing models for external use, it is crucial to maintain confidentiality of potentially sensitive or company-confidential information and to protect the privacy of the data subjects represented in the dataset or of the users interacting with the deployed model. Furthermore, if the party responsible for training or executing the model behaves lazily or maliciously, the resulting computation may be incomplete or incorrect, with potentially severe integrity issues. To address these concerns, we discussed solutions for constructing privacy-preserving machine learning models that preserve high utility. In addition, we presented mechanisms for verifying the correctness and completeness of results returned by potentially untrusted parties. The solutions discussed provide a foundation for designing privacy-aware and integrity-preserving machine learning techniques in (distributed) environments.

Acknowledgements

This work was supported in part by the EC under projects GLACIATION (101070141) and EdgeAI (101097300), by the Italian MUR under PRIN project POLAR (2022LA8XBH), and by project SERICS (PE00000014) under the MUR NRRP funded by the EU - NGEU. Project EdgeAI is supported by the Chips Joint Undertaking and its members including top-up funding by Austria, Belgium, France, Greece, Italy, Latvia, Netherlands, and Norway under grant agreement No. 101097300. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union, the Chips Joint Undertaking, or the Italian MUR. Neither the European Union, nor the granting authority, nor Italian MUR can be held responsible for them.

References

[1] F. Ahmed, D. Sánchez, Z. Haddi, and J. Domingo-Ferrer, "MemberShield: A framework for federated learning with membership privacy," *Neural Networks*, vol. 181, January 2025.

[2] M. Annamalai, A. Gadotti, and L. Rocher, "A linear reconstruction approach for attribute inference attacks against synthetic data," in *Proc. of USENIX SEC*, PA, USA, August 2024.

[3] M. Barbato, A. Ceselli, S. De Capitani di Vimercati, S. Foresti, and P. Samarati, "PriSM: A Privacy-friendly Support vector Machine," in *Proc. of ESORICS*, Tolouse, France, September 2025.

[4] S. Barezzani, S. De Capitani di Vimercati, S. Foresti, V. Ghirimoldi, and P. Samarati, "TA_DA: Target-Aware Data Anonymization," *IEEE Transactions on Privacy*, vol. 2, pp. 15–26, 2025.

[5] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," in *Proc. of ACM CCS*, Toronto, Canada, October 2018.

[6] H. Chabanne, A. De Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-preserving text classification on deep neural network," *Neural Processing Letters*, vol. 57, no. 2, 2025.

[7] C. Chen, L. Wei, J. Xie, and Y. Shi, "Privacy-preserving machine learning based on cryptography: A survey," *ACM TKDD*, vol. 19, no. 4, May 2025.

[8] P. Coscia, S. Ferrari, V. Piuri, and A. Salman, "Synthetic and (Un) secure: evaluating generalized membership inference attacks on image data," in *Proc. of SECRYPT*, Bilbao, Spain, June 2025.

[9] S. De Capitani di Vimercati, S. Foresti, V. Ghirimoldi, and P. Samarati, "DT-Anon: Decision tree target-driven anonymization," in *Proc. of DBSec*, San Jose, CA, USA, July 2024.

[10] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, R. Sassi, and P. Samarati, "Sentinels and twins: Effective integrity assessment for distributed computation," *IEEE TPDS*, vol. 34, no. 1, pp. 108–122, January 2023.

[11] S. De Capitani di Vimercati, S. Foresti, and P. Samarati, *Query Integrity in Smart Environments*. Springer Nature Switzerland, 2025.

[12] R. de Laage, P. Yuhala, F. Wicht, P. Felber, and C. Cachin, "Practical secure aggregation by combining cryptography and trusted execution environments," in *Proc. of DEBS*, Gothenburg, Sweden, June 2025.

[13] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine, "Authentic third-party data publication," in *Proc. of DBSec*, Schoorl, The Netherlands, August 2000.

[14] C. Dwork, "Differential privacy," in *Proc. of International colloquium on automata, languages, and programming*, Venice, Italy, July 2006.

[15] A. R. Elkordy, Y. H. Ezzeldin, S. Han, S. Sharma, C. He, S. Mehrotra, S. Avestimehr *et al.*, "Federated analytics: A survey," *APSIPA Transactions on Signal and Information Processing*, vol. 12, no. 1, 2023.

[16] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2006.

[17] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Workload-aware anonymization," in *Proc. of KDD*, Philadelphia, PA, USA, August 2006.

[18] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proc. of ICDE*, Atlanta, GE, USA, April 2006.

[19] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "ℓ-diversity: Privacy beyond k-anonymity," *ACM TKDD*, vol. 1, no. 1, March 2007.

[20] W. Pugh, "Skip lists: a probabilistic alternative to balanced trees," *Communications of ACM*, vol. 33, no. 6, pp. 668–676, 1990.

[21] X. Ran, Y. Xi, Y. Lu, X. Wang, and Z. Lu, "Comprehensive survey on hierarchical clustering algorithms and the recent developments," *Artificial Intelligence Review*, vol. 56, no. 8, pp. 8219–8264, August 2023.

[22] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," *ACM CSUR*, vol. 56, no. 4, pp. 1–34, 2023.

[23] P. Samarati, "Protecting respondents identities in microdata release," *IEEE TKDE*, vol. 13, no. 6, pp. 1010–1027, November/December 2001.

[24] N. Senavirathne and V. Torra, "On the role of data anonymization in machine learning privacy," in *Proc. of IEEE TrustCom*, Guangzhou, China, December 2020.

[25] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM CSUR*, vol. 55, no. 8, December 2022.

[26] J. Vaidya, H. Yu, and X. Jiang, "Privacy-preserving SVM classification," *Knowledge and Information Systems*, vol. 14, pp. 161–178, 2008.

[27] R. Xu, B. Li, C. Li, J. Joshi, S. Ma, and J. Li, "TAPFed: Threshold secure aggregation for privacy-preserving federated learning," *IEEE TDSC*, vol. 21, no. 5, pp. 4309–4323, September/October 2025.