Privacy-Preserving k-NN Graphs with Autoencoder-Based Representations for Sensitive Features

Gustavo Lima Oliveira[1] **, Maria da Graca Campos Pimentel**[1] **,**
Ricardo M. Marcacini[1]

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (USP)
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brazil

Abstract. *Privacy-preserving representation learning has gained significant attention for enabling secure data and model sharing by protecting sensitive information while maintaining data utility. In this paper, we present a new approach to privacy-preserving representation learning with k-NN-based graph models. This method maps the original feature space to a new space that balances feature utility, such as classification accuracy, with reducing privacy attack risks, and constructs a kNN graph from this new space. We evaluate three scenarios using real datasets to assess privacy-preserving graph representations. Experimental results show that learning a privacy-preserved representation and constructing a k-NN graph is a simple, intuitive, and competitive approach compared to other methods in the literature. Thus, this method enables graph data sharing with a lower risk of sensitive information extraction attacks.*

1. Introduction

Numerous studies have highlighted the importance of handling sensitive, private, and anonymized data in training machine learning models [Al-Rubaie and Chang 2019, Kong et al. 2020, Nasr et al. 2023]. Consequently, sharing data between organizations for model training has become a challenge. Simple data anonymization and the removal of sensitive attributes are not sufficient, as recent studies have shown that attacks can effectively use correlated attributes to identify sensitive information [Backstrom et al. 2007]. Therefore, more advanced methods have been proposed, such as encryption or some form of internal attribute encoding, especially when integrated with Federated Learning [Chen et al. 2023, Li et al. 2020b], which allows the sharing of models from different organizations to create a shared model.

A promising strategy that has been gaining attention is the sharing of graph-based models [Fu et al. 2023]. In this case, the organization's local datasets containing sensitive information are not shared. Instead, only a graph generated from the dataset's distance values—i.e., their proximity relationships—is shared . This graph is then used to train classification models using only the topological relationships and some label information. We briefly illustrate a simplified version of these approaches in Figure 1, considering three scenarios. Given a vector-space model $\mathbf{X} = (x_1, \ldots, x_m)$ representing m examples, where each example is characterized by five features $(f_1, f_2, f_3, f_4, f_5)$. We consider that features f_3 and f_5 are private features, containing sensitive information that should not be extractable through model or graph attacks. The remaining features are referred to as utility features and do not contain sensitive information. The first scenario represents a naïve approach, where a graph is constructed solely based on the k-nearest neighbors (kNN) of $\mathbf{X}$. In the second scenario, a preprocessing step is included to remove the private features before the graph construction. Finally, in the third scenario, a new data representation $(e_1, \ldots, e_m)$ is obtained using a mechanism that simulates a privacy attack within the feature learning process, with the expectation that the resulting kNN-based graph will be robust against privacy attacks.

A simple strategy, such as the one outlined in Scenario 1 of Figure 1, which involves only the construction of the graph, is known to be ineffective in preserving private attributes f_3 and f_5 in the example. This is because an attacker could retrieve this information merely by observing the adjacency matrix, which represents the principal distances. We demonstrate in the experimental section of this paper that an attacker with access to a 10% sample of leaked data can recover the remaining 90% of private information with an average accuracy of 0.91 in Scenario 1. In Scenario 2, where sensitive attributes are removed before constructing the graph, attacks are still possible because other characteristics correlated with attributes f_3 and f_5 are used to calculate the distances between examples. Thus, even without direct access to sensitive attributes, an attacker possessing the adjacency matrix could still recover sensitive information.

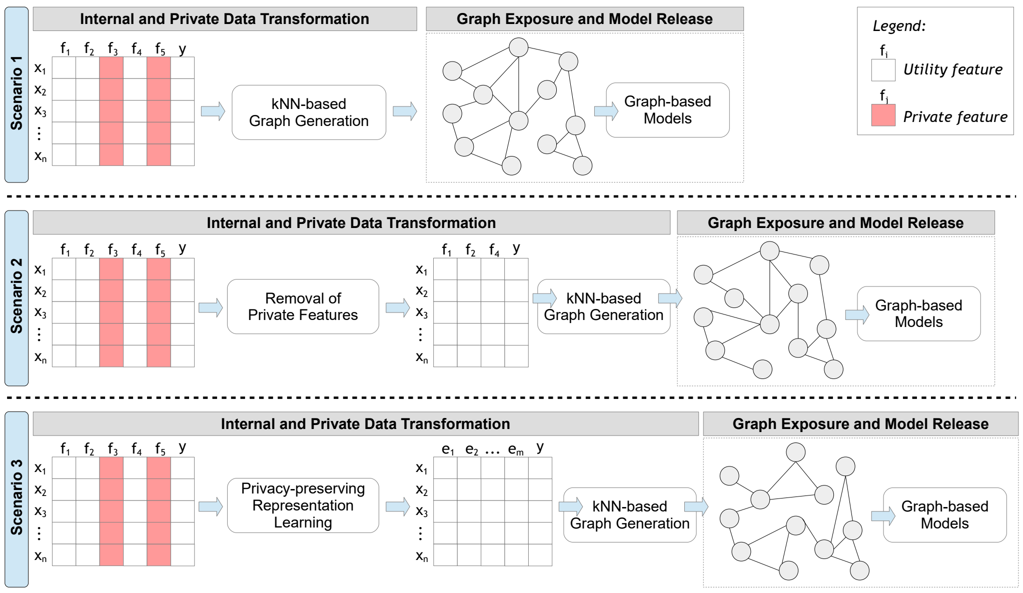

Scenario 1
Internal and Private Data Transformation
Graph Exposure and Model Release
kNN-based Graph Generation
Graph-based Models
Legend:
Utility feature
Private feature
Scenario 2
Removal of Private Features
Scenario 3
Privacy-preserving Representation Learning

Figure 1. Simplified illustration of three scenarios for privacy-preserving graph-based models. Scenario 1: A graph is constructed directly from the k-nearest neighbors (kNN) of the vector-space model X, including sensitive private features f_3 and f_5. Scenario 2: A preprocessing step removes private features f_3 and f_5 before constructing the graph. Scenario 3: A new data representation $(e_1, \ldots, e_m)$ is generated using a mechanism that simulates a privacy attack during the feature learning process, with the aim of creating a kNN-based graph robust against privacy attacks.

In this paper, we investigate an alternative strategy to address the problem of privacy preservation in graph-based representations [Ma et al. 2024]. The key idea is to explore privacy-preserving representation learning methods to learn a new feature space capable of maintaining the utility of features for a specific task, such as classification, while reducing the performance of an attack. Although similar strategies have received significant attention in deep learning-based models, their use in graph-based models has been less explored. Scenario 3 of Figure 1 illustrates the central proposal of this study and provides an overview of this strategy. In this approach, the original feature space is mapped to a new feature space using a representation learning method that simultaneously minimizes private features attack metrics and maximizes utility features classification metrics. This new space is then used to construct the graph that can be shared and used for machine learning tasks in relational data. In summary, the main contributions of this paper are threefold:

- First, we evaluate the three scenarios presented above on three real datasets used for experiments in the context of privacy-preserving data representation learning with graph-based representations, involving five classification methods. To the best of our knowledge, this is the first study to conduct an experimental evaluation comparing all scenarios. We demonstrate how easy it is to carry out the attack in Scenario 1. Additionally, we discuss the advantages and limitations of approaches based on Scenarios 2 and 3.
- Second, we extend privacy-preserving feature learning methods to graphs for Scenario 3. We explore how the method's weight and simulated attacks affect the representation learning process. We analyze three approaches: (1) prioritizing classification utility over attack resistance; (2) focusing on attack resistance and preserving sensitive features, even if it reduces utility; and (3) balancing both attack minimization and utility in the new representation.
- Third, we developed an open-source tool for k-NN graph-based privacy-preserving learning[1]. This tool maps problems from a vector-space model to a graph representation while preserving sensitive features. We hope it will be useful for projects that need to share models and data with sensitive information between organizations.

The remainder of this paper is organized as follows. Section 2 introduces some basic concepts and related works. The proposal of this paper is presented in more detail in Section 3, describing the privacy-preserving representation learning method and the construction of the k-NN graph. Section 4 presents the experimental results, followed by a discussion of limitations and future work in Section 5.

2. Background and Related Work

Graph-based machine learning methods have made significant improvements in various fields, including computer vision, natural language processing, recommendation systems, drug discovery, and fraud detection [Hoang et al. 2023]. However, with the popularization of utilizing graph machine learning techniques to systematically tackle issues across different application areas, protecting privacy remains a crucial aspect. Privacy-preserving approaches for graphs aim to share trustworthy generated graph data instead of the original sensitive data [Fu et al. 2023].

Privacy attackers in graphs can be categorized as active and passive. Active attackers insert structures into the graph before publication to identify targets in the published version. In contrast, passive attackers rely on observing the published graph and often

[1]`https://github.com/Labic-ICMC-USP/PrivacyPreservingLearning`

need minimal external information [Fu et al. 2023]. Protection strategies aim to reduce the uniqueness of nodes to mitigate two types of graph attacks: (1) Node identity disclosure [Hay et al. 2008] aims to reveal a target node, which can be used to determine whether a node belongs to a graph or to uncover specific properties of a particular node and (2) Link re-identification focus on review sensitive relationships between nodes.

We focus on passive attackers in k-NN graphs, where nodes represent instances and edges indicate connections between the k-nearest neighbors. In this context, the most common type of attack is node identification disclosure, where attackers observe node connections, such as degree distribution and neighbor connectivity, and some external information to infer attributes about node instances and identify those with similar characteristics. Existing privacy protection techniques for graphs include simple anonymization strategies. For instance, k-degree anonymization [Liu and Terzi 2008] ensures that each node in a published graph has at least k1 nodes with the same degree, reducing the likelihood of identifying the target node even with prior knowledge of degree distribution.

Recently, representation learning methods have been proposed to create more robust representations against privacy attacks in graphs. For example, [Ma et al. 2024] separates private and non-private information, allowing for a better balance between privacy and utility. It uses low-dimensional perturbations to generate anonymized graphs that protect against various inference attacks, while maintaining the usefulness of non-private information. However, these methods rely on having an initial graph with features for each node from the start of the proccess. In our case, where the input is not originally a graph but only data represented in a vector space model, our goal is to build and share a kNN graph within the context of privacy-preserving representation.

3. Method

We formally define the problem of graph-based privacy preservation with the goal of generating a k-nearest neighbors (kNN) graph that is robust to privacy attacks. Let $\mathbf{X} = \{x_1, \ldots, x_n\}$ be a set of n examples in a vector-space model, represented in an feature space $\mathcal{F} = (f_1, f_2, \ldots, f_m)$. We assume that the feature space $\mathcal{F}$ can be divided into two distinct types of features:

Utility Features: denoted as $\mathcal{F}_U = \{f_{u_1}, f_{u_2}, \ldots, f_{u_{m_u}}\}$, where $\mathcal{F}_U$ includes features that do not contain sensitive information and are used for the primary purpose of model utility and performance, for instance, graph-based classification tasks.

Private Features: denoted as $\mathcal{F}_P = \{f_{p_1}, f_{p_2}, \ldots, f_{p_{m_p}}\}$, where $\mathcal{F}_P$ includes features that contain sensitive information which should be protected from being extracted through attacks on the model or the graph. These features require protection from unauthorized access, model attacks, and data leaks. Examples include health data, financial information, location data, biometric identifiers, and legal records.

The process of constructing a privacy-preserving kNN-based graph involves two main steps. First, we preprocess the dataset $\mathbf{X}$ to protect sensitive information by applying a privacy-preserving method $M(\cdot)$, resulting in a transformed dataset $\hat{\mathbf{X}} = M(\mathbf{X}, \mathcal{F}_U, \mathcal{F}_P)$. In the second step, we construct the k-Nearest Neighbors (kNN) graph using $\hat{\mathbf{X}}$. The kNN graph is built by calculating the distance between each pair of examples in $\hat{\mathbf{X}}$ and connecting each example to its k nearest neighbors. The resulting graph is represented as $G = (V, E)$, where V denotes the set of vertices, each corresponding to an example in $\hat{\mathbf{X}}$, and E denotes the set of edges, where $E = \{(x_i, x_j) \mid x_j \text{ is one of the } k \text{ nearest neighbors of } x_i \text{ in } \hat{\mathbf{X}}\}$. At this step, access to the original features from $\mathcal{F}$ is no longer available. Instead, classification tasks is based solely on the graph structure and the connections between vertices, leveraging the proximity and relational information captured by the kNN graph and labeled nodes.

3.1. Utility metrics

In the graph-based classification process, the graph $G = (V, E)$, constructed from the dataset $\hat{\mathbf{X}}$, is employed for classification tasks. The process starts with node representation learning, where a function $\Phi(\cdot)$ is applied to the graph to transform each node v_i into a continuous vector representation (embedding) $\mathbf{e}_i = \Phi(v_i)$. These embeddings capture the topological and relational characteristics of the nodes within the graph. The learning of embeddings is performed using labeled data, with each node v_i associated with a class label or target attribute.

Once the embeddings are obtained, a classifier $C(\cdot)$ is trained to predict the class or target attribute for each node based on its embedding. The classifier is tasked with mapping the embeddings $\mathbf{e}_i$ to their respective labels or attributes. The effectiveness of this classification process is assessed using utility metrics, including accuracy, precision, recall, and F1-score.

3.2. Privacy Attacks

In practice, attackers may gain access to a portion of private features due to various reasons [Oliveira et al. 2021, Li et al. 2020a]. For instance, data breaches might occur if an organization's database is compromised, resulting in unauthorized access to sensitive information [Al-Rubaie and Chang 2019]. Insider threats could involve employees with access to private data leaking information maliciously or accidentally [Tran et al. 2024]. Additionally, incomplete or improperly secured data sharing agreements might inadvertently expose private features. Even a small fraction of leaked data can be leveraged by attackers to train a classifier, which could then be used to infer sensitive information about other data points [Al-Rubaie and Chang 2019, Li et al. 2020a, Tran et al. 2024].

To formalize the privacy attack process, consider that an attacker has access to a subset of private information that has been leaked. Let $\mathbf{F}_{P,\text{leaked}}$ denote the subset of private features from the leaked data, and $\mathbf{y}_{P,\text{leaked}}$ be the corresponding labels. The process begins by training a classifier $C_{attack}(\cdot)$ using the embeddings of nodes derived from the shared graph. Each node v_i in the graph is mapped to an embedding $\mathbf{h}_i = \Phi(v_i)$. The classifier $C_{attack}(\cdot)$ is trained with these embeddings and the labels $\mathbf{y}_{P,\text{leaked}}$ from the leaked private data. Once trained, the attacker uses the classifier $C_{attack}(\cdot)$ to predict labels for the remaining nodes in the graph. Thus, for each node v_j with embedding $\mathbf{h}_j$, the predicted label $\hat{y}_j$ is given by $\hat{y}_j = C_{attack}(\mathbf{h}_j)$.

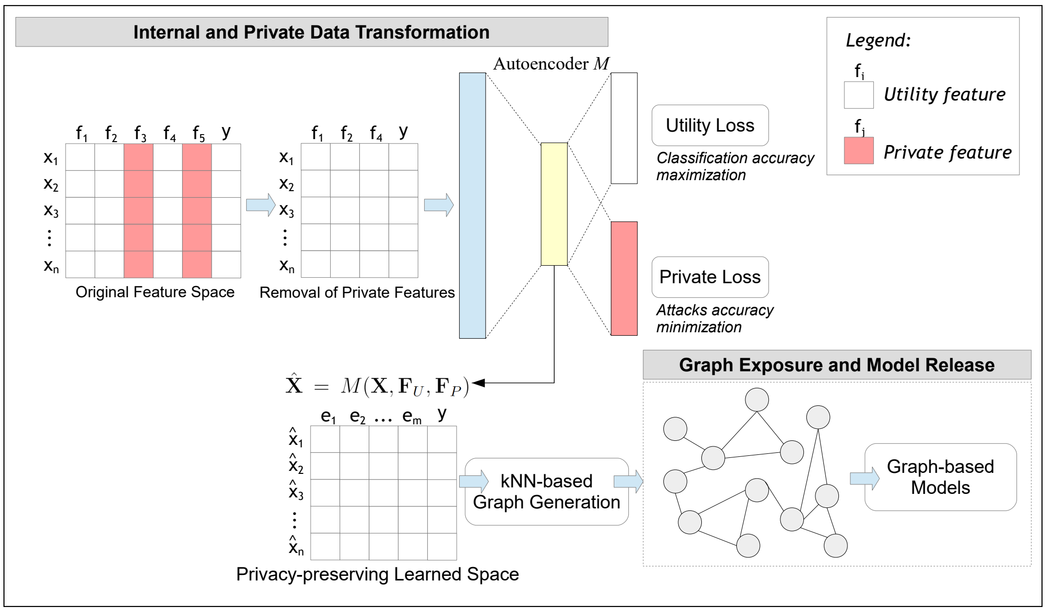
If the graph $G = (V, E)$ was not effective in preserving sensitive information, the kNN graph may still encode sufficient information for $C_{attack}(\cdot)$ to accurately predict private labels. The effectiveness of the attack can be measured by the accuracy of the classifier, as defined in Equation 1.

$$\text{Accuracy}_{attack} = \frac{1}{|V|} \sum_{v_j \in V} I(y_j = \hat{y}_j), \tag{1}$$

y_j denotes the true label, and $I(\cdot)$ is an indicator function that equals 1 if the predicted label $\hat{y}_j$ matches the true label y_j, and 0 otherwise.

In this context, $\hat{y}_j$ represents the predicted private label associated with the private feature $\mathcal{F}_P$ of node v_j, illustrating the direct connection between the private feature information $\mathcal{F}_P$ and the predicted labels. High accuracy of $C_{attack}(\cdot)$ indicates that the graph's embeddings do not effectively protect sensitive information, allowing the attacker to recover private labels with a significant degree of accuracy.

3.3. Privacy-preserving kNN-based Graph



Internal and Private Data Transformation
Autoencoder M
Legend:
f_i Utility feature
f_j Private feature
f_1 f_2 f_3 f_4 f_5 y
x_1 x_2 x_3 ⋮ x_n
Original Feature Space
f_1 f_2 f_4 y
Removal of Private Features
Utility Loss
Classification accuracy maximization
Private Loss
Attacks accuracy minimization
Graph Exposure and Model Release
$\hat{X} = M(X, F_U, F_P)$
e_1 e_2 … e_m y
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ ⋮ $\hat{x}_n$
kNN-based Graph Generation
Graph-based Models
Privacy-preserving Learned Space

Figure 2. Privacy-preserving representation learning with a modified autoencoder, which learns representations by optimizing both utility and privacy loss functions. The encoder produces (hopefully) privacy-preserved latent representations, which are then used to construct a kNN graph that can be shared by organizations for classification tasks.

In this section, we present our method for privacy-preserving representation learning using a modified autoencoder, as described in Figure 2. Our approach incorporates two distinct loss functions: one designed to maximize the utility of learned embeddings for classification tasks, and another aimed at minimizing the risk of predict private information. These loss functions guide the training process to balance effective classification with robust privacy protection. The learned autoencoder embeddings are used to construct a k-nearest neighbors (kNN) graph for subsequent relational classification tasks.

This approach, inspired by previous literature on autoencoders for privacy preservation [Han et al. 2023, Ma et al. 2024], contains significant adjustments. Unlike those approaches, we do not use a reconstruction loss, as it tends to preserve the original feature space, including private features. Moreover, while our approach is somewhat similar to adversarial attacks, the key difference is that our approach uses a more straightforward mechanism, focusing directly on minimizing privacy risks rather than generating adversarial examples. The goal is to learn a representation $\hat{\mathbf{X}} = M(\mathbf{X}, \mathbf{F}_U, \mathbf{F}_P)$ where M is the autoencoder model. The encoder maps the input $\mathbf{X}$ to a lower-dimensional latent space representation $\mathbf{z}$, and the decoder reconstructs $\mathbf{X}$ from $\mathbf{z}$. Let $\mathbf{z} = \Phi(\mathbf{X})$ denote the latent representation learned by the encoder function Φ. To guide the training of the autoencoder, two loss functions are employed:

- **Utility Loss:** This loss function aims to maximize the effectiveness of the learned representation for classification tasks. It is defined as the categorical cross-entropy loss between the predicted labels from the encoder representation and the true public labels $\mathbf{y}_U$. Formally, if $\mathbf{y}_U$ represents the true class labels, the loss function is given by:

$$\mathcal{L}_U = \text{CategoricalCrossentropy}(\mathbf{y}_U, \text{PublicPredictor}(\mathbf{z})).$$

  where $\text{PublicPredictor}(\mathbf{z})$ refers to the model that predicts the class labels from the latent representation $\mathbf{z}$.
- **Private Loss:** This loss function aims to minimize the risk of exposing sensitive information. It is defined as the negative categorical cross-entropy loss using a subset of private features $\mathbf{y}_P$ as labels for the private predictor. The negative sign in the privacy loss term is used to penalize successful attacks on private features, thereby encouraging the autoencoder to adjust its weights to learn a latent representation that minimizes the risk of sensitive information disclosure.
  If $\mathbf{y}_P$ represents the labels derived from the private features, the loss function is given by:

$$\mathcal{L}_P = -\text{CategoricalCrossentropy}(\mathbf{y}_P, \text{PrivatePredictor}(\mathbf{z})).$$

  $\text{PrivatePredictor}(\mathbf{z})$ denotes the model that predicts private features from the latent representation $\mathbf{z}$.

The total loss function for training the autoencoder is a weighted combination of the utility and private losses, with weights β and γ, respectively. The combined loss function is defined in Equation 2 and represents a trade-off between maximizing utility for classification while minimizing the risk of revealing sensitive information.

$$\mathcal{L} = \beta \cdot \mathcal{L}_U + \gamma \cdot \mathcal{L}_P. \tag{2}$$

After training, the learned privacy-preserving representation $\mathbf{z}$ is utilized to construct a k-Nearest Neighbors (kNN) graph. Formally, for each node $\mathbf{e}_i$, let $\mathcal{N}_k(i)$ denote the set of indices of the k nearest neighbors of $\mathbf{e}_i$ based on the distance function d. The kNN graph $G = (V, E)$ is then constructed, where an edge $(i, j) \in E$ is included if $j \in \mathcal{N}_k(i)$. It is important to note that only the kNN graph G is shared, without the embeddings obtained from the autoencoder, allowing organizations to use this graph for training their models while maintaining the privacy of the sensitive information.

4. Experimental Evaluation

4.1. Datasets

To evaluate the approaches discussed in this paper, we use three benchmark datasets[2] derived from real-world scenarios:

- **COMPAS Dataset:** The COMPAS dataset is widely used in criminal justice research to assess the risk of recidivism among criminal defendants. It includes a range of features related to the defendants' background and criminal history. For our study, we use a version of the COMPAS dataset that includes 8 features and 6,907 rows. The dataset is used to predict whether a defendant is likely to re-offend. In our setup, the private features are sex and race, which are sensitive attributes that could influence predictions.
- **Adult dataset:** Also known as the "Census Income" dataset, this dataset is used to predict whether an individual's annual income exceeds \$50,000. Originally, it contains 14 features and 32,561 rows. In this study, we performed random sampling to reduce the dataset to 10,000 rows while preserving the original feature distributions. For our experiments, the private features are race and sex.
- **Student Performance Dataset:** This dataset includes information on student performance and contains 395 rows with 30 features. Examples of features in this dataset are age, sex, address, family size, and various academic-related attributes such as study time, number of absences, and grades in different subjects. The target attribute is the student's grade in the first period, where we predict whether the student's grade is greater than or equal to 12. For our experiment, the private feature is sex.

4.2. Evaluation Criteria

We evaluate the effectiveness of our approach through three different scenarios, each representing a distinct method for constructing the k-Nearest Neighbors (kNN) graph, and compare these with a null model.

In **Scenario 1**, the kNN graph is constructed directly from the original dataset without any modifications to address private features. This approach serves as a baseline to understand how well the kNN graph performs when private features are not treated or removed. In **Scenario 2**, we construct the kNN graph after directly removing the private features from the dataset. Although private features are removed, this scenario does not address the possibility that other features might be correlated with the removed private features, potentially allowing attackers to infer private information. In **Scenario 3**, the kNN graph is built using the latent representations learned from our privacy-preserving autoencoder. We analyze the impact of the parameter γ on the balance between utility loss and privacy loss.

In both scenarios, we use the constructed kNN graph to train classifiers to assess both its utility and the for simulating attacks on private features. For each scenario, we generate node embeddings from the graphs using the DeepWalk algorithm. These embeddings are then used to train and evaluate five different classifiers: *K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree, Random Forest, and Multilayer*

[2] `https://ashryaagr.github.io/Fairness.jl/dev/datasets/`

Perceptron (MLP), with hyperparameters optimized using a validation set. In all scenarios, we construct graphs using Euclidean distance with $k = 7$ nearest neighbors. This value was selected as it produced the most effective graphs, maximizing the number of connected components while not significantly increasing the average vertex degree.

We assumed that the attacker had access to 10% of leaked private data to train their attack models. Although this is an overestimated value, it allows us to experimentally assess whether the privacy-preserving model can effectively handle more challenging attack scenarios. For training the classifiers and analyzing their utility for the target task, we used a typical split of 70% training data and 30% testing data.

Finally, As a baseline comparison, we use a *DummyClassifier*, which estimates how well an attacker could predict private features based solely on the data distribution of private features, i.e., the attacker can infer private information without access to sophisticated prediction methods, relying only on probabilistic estimates.

4.3. Experimental Results

We analyzed the experimental results from three aspects. First, we compared the performance of the approaches under an optimistic attack scenario. In this setup, we deliberately selected the best-performing attack models and, correspondingly, the best-performing utility classification models. This scenario aims to assess the situation where the attacker achieves significant success, thereby reflecting the highest risk to the organization. In this case, the experimental results is presented in Table 1, where each row represents the best results between all classifiers per dataset. The *Utility Acc* column indicates the accuracy of the model's performance, which we aim to maximize. The *Attack Acc* column shows the accuracy in predicting private features, which we aim to minimize.

Table 1. Comparison of *Utility Acc* and *Attack Acc* across three scenarios, including a Dummy Attack baseline.

	Scenario 1		*Scenario 2*		*Scenario 3 (ours)*		
Dataset	Utility Acc ↑	Attack Acc ↓	Utility Acc ↑	Attack Acc ↓	Utility Acc ↑	Attack Acc ↓	Dummy Attack
COMPAS	0.644	0.907	0.630	**0.807**	0.608	**0.807**	0.701
Adult	0.807	0.960	0.814	0.763	0.781	**0.673**	0.644
Student	0.663	0.885	0.638	0.634	0.630	**0.601**	0.588

It is important to note that in Scenario 1, where no prior treatment of private features is conducted before graph construction, the *Attack Acc* is notably high. An attacker achieves promising results, with an average accuracy of **0.91** across the three datasets. This type of attack significantly outperforms a Dummy Attack, which is based solely on the distribution of private feature values.

The effectiveness of attacks in Scenario 1 shows the necessity for proper treatment of private features before graph construction. Both in Scenario 2 and Scenario 3 (our approach), there is a reduction in *Attack Acc* while maintaining *Utility Acc*. Our proposed approach in Scenario 3 showed a slight reduction in *Utility Acc* for the COMPAS dataset without improvement in *Attack Acc*. However, in this scenario, a Dummy Attack already achieves an accuracy of 0.701, indicating that the distribution of private features alone could lead to promising attacks. Our proposed approach demonstrated good performance for the Adult and Student datasets, reducing *Attack Acc* close to the Dummy Attack.

Tables 2, 3, and 4 show the *Utility Acc* and *Attack Acc* results for each classifier and dataset. This analysis reveals that no single classifier consistently outperforms others for either the target classification task or for performing an attack. Therefore, classifiers need to be calibrated based on the specific dataset.

Table 2. Comparison of *Utility Acc* and *Attack Acc* on the COMPAS dataset.

	Scenario 1		*Scenario 2*		*Scenario 3 (ours)*	
Classifier	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$
Decision Tree	0.549	0.801	0.535	0.776	0.527	0.799
KNN	**0.644**	0.899	**0.630**	0.764	0.607	0.768
MLP	0.639	**0.907**	0.618	0.752	**0.608**	**0.807**
Random Forest	0.555	0.807	0.536	**0.807**	0.534	**0.807**
SVM	0.570	0.810	0.557	**0.807**	0.557	**0.807**

Table 3. Comparison of *Utility Acc* and *Attack Acc* on the Adult dataset.

	Scenario 1		*Scenario 2*		*Scenario 3 (ours)*	
Classifier	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$
Decision Tree	0.748	0.745	0.759	0.665	0.750	0.665
KNN	0.796	**0.960**	0.793	**0.763**	0.776	0.633
MLP	**0.807**	0.929	**0.814**	0.760	**0.781**	0.633
Random Forest	0.750	0.726	0.749	0.686	0.749	**0.673**
SVM	0.749	0.802	0.749	0.704	0.750	**0.673**

Table 4. Comparison of *Utility Acc* and *Attack Acc* on the Student dataset.

	Scenario 1		*Scenario 2*		*Scenario 3 (ours)*	
Classifier	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$	Utility Acc $\uparrow$	Attack Acc $\downarrow$
Decision Tree	0.605	0.750	0.504	0.531	**0.630**	0.587
KNN	**0.663**	0.837	0.597	0.559	0.613	0.587
MLP	0.597	**0.885**	0.563	**0.634**	0.597	0.556
Random Forest	0.647	0.750	0.597	0.545	0.605	0.567
SVM	0.580	0.874	**0.638**	0.607	**0.630**	**0.601**

In Table 5, we analyze the effect of varying γ (0, 0.5, and 1) on *Utility Acc* and *Attack Acc*, with $\beta = 1$. This analysis examines how changing the penalty for attacks on private features influences performance, while keeping utility feature performance constant. As expected, *Utility Acc* generally increases when γ is set to 0, since the model learns a representation optimized solely for classification, making it more susceptible to attacks. On the other hand, tuning γ presents a more complex challenge. Overall, a γ value of 0.5 shows a good trade-off between *Utility Acc* and *Attack Acc*, providing a balanced performance across different classifiers and datasets.

Table 5. Comparison of *Utility Acc* and *Attack Acc* varying the privacy loss weight.

	Utility Acc$\uparrow$			*Attack Acc*$\downarrow$		
Dataset	$\gamma = 0$	$\gamma = 0.5$	$\gamma = 1$	$\gamma = 0$	$\gamma = 0.5$	$\gamma = 1$
COMPAS	**0.609**	0.581	0.568	0.789	**0.786**	0.791
Adult	**0.781**	0.768	0.770	0.645	**0.636**	0.639
Student	**0.664**	0.639	0.630	0.533	**0.524**	0.529

We highlight the results obtained on the Adult dataset, where the attack accuracy was reduced from 0.960 to 0.763 with the second scenario and to 0.673 with our approach, which is very close to the minimum (dummy classifier) accuracy of 0.644 (Table 1). This scenario represents a situation in which Scenario 2, involving the simple removal of the private feature, fails to limit the attack because there is a correlated feature that is used to conduct the attack. In this case, Table 6 shows the correlation values between the private feature ”sex” and the other attributes. Note that there are significant correlations, particularly between ”relationship” and ”sex”. In contrast, our method, detailed in Scenario 3, limits this type of attack because, in addition to removing the private features, it also learns a representation that obfuscates public features correlated with the private features through an internal attack strategy during training.

Table 6. Correlation values between private feature “sex” and the other attributes from Adult Dataset.

Feature	Relationship	Marital-status	Occupation	Income
Correlation	0.64	0.46	0.42	0.21

5. Concluding Remarks

In this paper, we explored the effectiveness of autoencoder-based approaches for privacy preservation in k-NN graph data. We examined three distinct experimental scenarios: (1) building k-NN graphs without any treatment of private features, (2) directly removing private features, and (3) utilizing a privacy-preserving representation space.

In our study, Scenario 2 and Scenario 3 yielded similar overall results. However, when utility features are correlated with private features, Scenario 2 tends to exhibit lower performance. This is due to the fact that correlated features can produce similar distances to the removed attributes, enabling attack models to exploit this information and retrieve sensitive data. In contrast, Scenario 3, which employs our proposed privacy-preserving representation approach, has the potential to learn a representation that indirectly penalizes such correlated features. This approach enhances resilience against privacy breaches while maintaining data utility, as the attack is mitigated during the representation learning.

Future work should address some limitations of this study. First, we plan to evaluate the proposed approach on a broader range of datasets, encompassing diverse characteristics and sizes. Second, we aim to integrate this privacy-preserving representation method with federated learning frameworks for graph-based methods to enhance the model’s robustness and applicability in distributed environments.

References

Al-Rubaie, M. and Chang, J. M. (2019). Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, 17(2):49–58.

Backstrom, L., Dwork, C., and Kleinberg, J. (2007). Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 181–190, New York, NY, USA. Association for Computing Machinery.

Chen, H., Zhu, T., Zhang, T., Zhou, W., and Yu, P. S. (2023). Privacy and fairness in federated learning: perspective of tradeoff. *ACM Computing Surveys*, 56(2):1–37.

Fu, D., Bao, W., Maciejewski, R., Tong, H., and He, J. (2023). Privacy-preserving graph machine learning from data to computation: A survey. *SIGKDD Explor. Newsl.*, 25(1):54–72.

Han, X., Yang, Y., Wang, L., and Wu, J. (2023). Privacy-preserving network embedding against private link inference attacks. *IEEE Transactions on Dependable and Secure Computing*.

Hay, M., Miklau, G., Jensen, D., Towsley, D., and Weis, P. (2008). Resisting structural re-identification in anonymized social networks. *Proc. VLDB Endow.*, 1(1):102–114.

Hoang, V. T., Jeon, H.-J., You, E.-S., Yoon, Y., Jung, S., and Lee, O.-J. (2023). Graph representation learning and its applications: a survey. *Sensors*, 23(8):4168.

Kong, C., Chen, B., Li, S., Chen, Y., Chen, J., Zhou, Q., Wang, D., and Zhang, L. (2020). Privacy attack and defense in network embedding. In *International Conference on Computational Data and Social Networks*, pages 231–242. Springer.

Li, K., Luo, G., Ye, Y., Li, W., Ji, S., and Cai, Z. (2020a). Adversarial privacy-preserving graph embedding against inference attack. *IEEE Internet of Things Journal*, 8(8):6904–6915.

Li, L., Fan, Y., Tse, M., and Lin, K.-Y. (2020b). A review of applications in federated learning. *Computers Industrial Engineering*, 149:106854.

Liu, K. and Terzi, E. (2008). Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 93–106, New York, NY, USA. Association for Computing Machinery.

Ma, L., Li, C., Sun, S., Guo, S., Wang, L., and Li, J. (2024). Privacy-preserving graph publishing with disentangled variational information bottleneck. *Concurrency and Computation: Practice and Experience*, 36(10):e7963.

Nasr, M., Mahloujifar, S., Tang, X., Mittal, P., and Houmansadr, A. (2023). Effectively using public data in privacy preserving machine learning. In *International Conference on Machine Learning*, pages 25718–25732. PMLR.

Oliveira, G. L., Marcacini, R. M., and Pimentel, M. d. G. C. (2021). Privacy-preserving on heterogeneous network embedding for clinical events. *Proceedings of the First MLSys Workshop on Graph Neural Net-works and Systems (GNNSys'21)*.

Tran, A.-T., Luong, T.-D., and Huynh, V.-N. (2024). A comprehensive survey and taxonomy on privacy-preserving deep learning. *Neurocomputing*, page 127345.