PriveTAB : Secure and Privacy-Preserving sharing of Tabular Data

Anantaa Kotal
anantak1@umbc.edu
University of Maryland, Baltimore County
USA

Aritran Piplai
apiplai1@umbc.edu
University of Maryland, Baltimore County
USA

Sai Sree Laya Chukkapalli
saisree1@umbc.edu
University of Maryland, Baltimore County
USA

Anupam Joshi
joshi@umbc.edu
University of Maryland, Baltimore County
USA

ABSTRACT

Machine Learning has increased our ability to model large quantities of data efficiently in a short time. Machine learning approaches in many application domains require collecting large volumes of data from distributed sources and combining them. However, sharing of data from multiple sources leads to concerns about privacy. Privacy regulations like European Union's General Data Protection Regulation (GDPR) have specific requirements on when and how such data can be shared. Even when there are no specific regulations, organizations may have concerns about revealing their data. For example in cybersecurity, organizations are reluctant to share their network-related data to permit machine learning-based intrusion detectors to be built. This has, in particular, hampered academic research. We need an approach to make confidential data widely available for accurate data analysis without violating the privacy of the data subjects. Privacy in shared data has been discussed in prior work focusing on anonymization and encryption of data. An alternate approach to make data available for analysis without sharing sensitive information is by replacing sensitive information with synthetic data that behave as original data for all analytical purposes. Generative Adversarial Networks (GANs) are one of the well-known models to generate synthetic samples that can have the same distributional characteristics as the original data. However, modeling tabular data using GAN is a non-trivial task. Tabular data contain a mix of categorical and continuous variables and require specialized constraints as described in the CTGAN model.

In this paper, we propose a framework to generate privacy-preserving synthetic data suitable for release for analytical purposes. The data is generated using the CTGAN approach, and so is analytically similar to the original dataset. To ensure that the generated data meet the privacy requirements, we use the principle of t-closeness. We ensure that the distribution of attributes in the released dataset is within a certain threshold distance from the real dataset. We also encrypt sensitive values in the final released version of the dataset to minimize information leakage. We show that in a variety of cases, models trained on this synthetic data instead of the real data perform nearly as well when tested on the real data. Specifically, we show that the machine learning models used for network event/attack recognition tasks do not have a significant loss in accuracy when trained on data generated from our framework in place of the real dataset.

IWSPA '22, April 24–27, 2022, Baltimore, MD, USA

ACM ISBN 978-1-4503-9230-3/22/04…$15.00
https://doi.org/10.1145/3510548.3519377

CCS CONCEPTS

• **Security and privacy → Data anonymization and sanitization**; **Privacy-preserving protocols**; *Domain-specific security and privacy architectures*; *Privacy protections*.

KEYWORDS

tabular datasets, network datasets, data privacy, secure data sharing, data generation, generative adversarial networks, machine learning

ACM Reference Format:
Anantaa Kotal, Aritran Piplai, Sai Sree Laya Chukkapalli, and Anupam Joshi. 2022. PriveTAB : Secure and Privacy-Preserving sharing of Tabular Data. In *Proceedings of the 2022 ACM International Workshop on Security and Privacy Analytics (IWSPA '22), April 24–27, 2022, Baltimore, MD, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3510548.3519377

1 INTRODUCTION

The importance of machine learning has increased significantly in the last decade. A growing number of domains now rely on machine learning to discover novel insights from large data sets. While some machine learning approaches can work with small data sets, the modern push in that area is for deep learning approaches where large volumes of data are required. The development of these novel ML techniques has overlapped with an increased reliance on technology for various aspects of our lives. This embedding of computing in the daily fabric of our lives means that substantial amounts of data are being collected at an individual and aggregated level.

The resulting potential for inappropriate dissemination and usage of a given consumer's private data and derived information has raised concern among the public [3], prompting the creation of a plethora of data protection regulations like the Payment Card Industry Data Security Standard (PCI DSS) [9], the European Union's General Data Protection Regulation (GDPR) [1], and the Children's Online Privacy Protection Act (COPPA) [8]. A Pew study in 2019[4]

showed that 81% of Americans don't feel they have control over data that is collected, and that the risk of collecting data outweighs the benefits. The past decade of research in machine learning has led to sophisticated algorithms that can obtain very significant insights into users based on data. It has often been said that users are not customers of social media companies, but products they sell to advertisers. To address the public's fear of large-scale data collection and its potential for misuse, we need to prioritize privacy guarantees for all potentially sensitive data, not just the Personally Identifiable Information (PII).

Previous studies have proposed different privacy approaches for machine learning. One approach is to distribute the learning process to each site, learn from the sensitive data, and share back the model which can be combined. An alternate approach is to share the data with the central site while respecting privacy. Among these various privacy approaches, differential privacy [15] is most widely used due to its algorithmic simplicity, and relatively small systems overhead. However, differential privacy does not guarantee complete anonymity for sensitive data. Besides differential privacy, there are many other privacy definitions, such as k-anonymity [16] and l-diversity [30]. The principle of t-closeness[28] requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall table. This provides a privacy guarantee in shared data that ties back to the notion of differential privacy [12].

Most of the anonymization techniques discussed before still require a subset of the real data to be shared. This is risky for some domains, where we do not want any part of the original data to be shared with a remote entity. For example, network event data is continuously being collected in hosts, firewalls, and routers. It includes information specific to the user's behavior, including the IP addresses most often visited or period of most online activity and even the contents of the packet they are sending. Such network data has been used in a variety of machine learning models to detect attacks, going back to the pioneering work of Lee and Stolfo[27]. The ability to have large network-level data from a variety of sources under attack can help build good machine learning-based intrusion detection systems.

However, security research has been significantly hampered by the inability of organizations to share such data with others, especially academic researchers. Even within an organization, there are restrictions on access to this data. At UMBC, for instance, researchers can get access to only the headers of UMBC network data, and that too after they have been anonymized. This reluctance to share is because such data can give away information about the user's behavior that is protected, or can give an outside organization significant insight into the sharing organization. In the worst case, such data can be exploited to the benefit of an adversary. So organizations are unwilling to share any form of this data with an outside entity that needs this data for computational tasks such as intrusion detection, network event modeling, etc. To guarantee no part of the user's data is being shared outside the user's private network even for research purposes, we need an alternative to sharing the real data that can be exploited. This is true for many other domains where data is sensitive due to either regulatory (e.g. medical) or competitive reasons.

In this work, we propose a framework that replaces sensitive data with synthesized data that closely resemble real data for analytical purposes, minimizing the need for accessing real sensitive data for distributed machine learning. Previous studies[10] have proposed synthetic data replacement for a sensitive attribute using models such as Decision Trees, Random Forest, Support Vector Machine, etc. Dandekar et al.'s work propose that the synthetic datasets are close to the original data based on statistical properties like mean or median. However, that is not a sufficient condition to replace real data with synthetic data for most modern machine learning systems. We need to test whether a classifier trained on the synthetic data can still predict accurately on data from the original dataset.

One of the approaches to generate synthetic data that bears a close resemblance to original data for any analytical task is using generative adversarial networks (GANs). GANs are a type of neural network that can feed on random noise as input, and as the training progresses produce realistic copies of the real data. GANs are often used for synthetic data generation and translation in image and text data [6, 20, 43, 46]. Synthetic data generated using GAN has been shown to replace real data for statistical and analytical purposes. Tabular data, like network event data, contains a mix of categorical variables (e.g. Event ID, Reporting Device, Overall Severity, etc.) as well as continuous variables (time to live, count, etc.). To generate synthetic tabular data that looks and behaves the same as real data, we use the CTGAN model as proposed by Liu et al. [45]. The CTGAN model accounts for multi-modality in continuous variables and class imbalance in discrete variables. It can synthesize tabular data that are close to real data for all analytical functions.

We use CTGAN [45] to generate synthetic tabular data. We ensure that the distribution of attributes in the sampled dataset is within a certain threshold distance from the real dataset, thus securing the principle of t-closeness[28]. Our data generator can create data that is very similar to the original without repeating data points from the original dataset. However, the data generator can not fabricate completely new values for some attributes like IP Addresses. To guarantee extra privacy, we encrypt selected sensitive attributes using a hashing algorithm. We show that the machine learning models for tabular data have a negligible loss in accuracy when trained on data generated through our framework. We show this specifically for event recognition tasks using network event data. We also use this use framework for standard tabular datasets collected from the UCI machine learning repository that is often used for ML classification tasks.

In the following sections, we go into the details of our proposed method and experimental results. The rest of the paper is organized is as follows: In Section 2, we describe the background and motivation for our work. In Section 3, we describe our proposed methodology. In Section 4, we give details of our experiment and results. We discuss related works in Section 5 We conclude the paper in Section 6.

2 BACKGROUND

2.1 Privacy in Shared Data

As researchers are developing more sophisticated machine learning models, there is a gap between the computational capabilities available through ML and the data available for research. Data scientists

are invested in gathering large volumes of data with secure and privacy-preserving approaches. There has been significant research in designing privacy-preserving data sharing methods. However, most approaches have a caveat associated with them.

The notion of privacy has been heavily discussed for data sharing in machine learning. Among these various privacy approaches, differential privacy [15] is most widely used due to its algorithmic simplicity, and relatively small systems overhead. However, differential privacy does not guarantee complete anonymity for sensitive data. For gradient-based learning methods, differential privacy applies random perturbation to the intermediate output at each iteration. This means that one cannot draw specific conclusions about any sample in the learning process. However, it still allows most of the data to be shared intact from one host to another. This is still a violation of privacy requirements if the hosts belong to different geographic and administrative boundaries. It also makes it hard to ascertain the extent of conclusions drawn from the shared data, which is a fundamental guarantee of data protection regulations. For example, in the case of network event data, it is not sufficient to hide or anonymize the destination IP list from an individual source. The host to which the data is being shared should not be able to draw any conclusion regarding the systemic behavior of the individual source from the data.

The other notions of privacy in shared data propose sharing data such that anonymity is guaranteed for data subjects. K-anonymity [16] proposes a way for data holders to release a version of their private data with scientific guarantees that the individuals who are the subjects of the data cannot be re-identified while the data remain practically useful. This is guaranteed from the principle that each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appears in the release.

In the 2007 paper on L-diversity [30], the authors point out that k-anonymized datasets are susceptible to adversarial attacks. An attacker can discover the values of sensitive attributes when there is little diversity in those sensitive attributes. Additionally, k-anonymity does not guarantee privacy against attackers using background knowledge. As a solution to these problems, the authors propose the notion of l-diversity which suggests that each equivalence class has at least l well-represented values for each sensitive attribute. However, l-diversity also has its limitations. In particular, it is neither necessary nor sufficient to prevent attribute disclosure.

As a solution, Li et al. [28] propose t-closeness, which requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall table. This means that the distance between the distributions of a sensitive in shared vs original table should be no more than a threshold, t. The distance measure used in this case is the Earth Mover's Distance (EMD) [39]. t-closeness with EMD satisfies generalization and subset property. It provides a practical approach to privacy in data sharing. Furthermore, it can be proved that t-closeness and ϵ-differential privacy are strongly related to one another when it comes to anonymizing data sets [12].

Previous studies have proposed synthetic data replacement for sensitive attributes as an approach to security. In the 2018 paper by Dandekar et al. [10], the authors use models such as Decision Trees, Random Forest, Support Vector Machine, etc. to generate new data. They show that the generated data have a statistical resemblance to the original data by showing that the datasets have similar means and close KL divergence values. However, this is not sufficient to determine that the synthetic data can be shared in place of original data for any learning methods. Unlike Dandekar et al.'s work, the test of our synthetic dataset is not its statistical properties like mean or median. We test whether a classifier trained on the synthetic data can still predict accurately on data from the original dataset.

2.2 Generative adversarial networks

Generative Adversarial Networks (GAN) were proposed as a framework for estimating generative models via an adversarial process. In the GAN framework, a generative model G captures the data distribution, and a discriminative model D estimates the probability that a sample came from the original distribution rather than G. The training procedure for G is to maximize the probability of D making a mistake. As the training progresses, the generator gets better at generating new examples that plausibly come close to the samples from the original distribution. The idea behind GAN can be formulated as a two-player min-max game with value function $V(G, D)$:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[logD(x)] + E_{z\sim p_z(z)}[log(1-D(G(z)))] \tag{1}$$

The original paper on GAN by Goodfellow et al. [19], describes GAN's utility in generating new plausible samples for image datasets, such as the MNIST handwritten digit dataset, the CIFAR-10 small object photograph dataset, and the Toronto Face Database. In general, there is a lot of evidence of GAN being used for synthetic data generation and translation in image and text data [6, 20, 43, 46].

Tabular data, like network event data, contains a mix of categorical variables (e.g. Event ID, Reporting Device, Overall Severity, etc.) and continuous variables (e.g. time to live, count, etc.) This is different from standard image data that are often generated through GAN. To generate synthetic tabular data that looks and behaves the same as real data, we need further constraints on the generation process to get the desired outcome. We will further discuss the difference between tabular data and image data in Section 3.

In an unconditioned generative model, there is no control over the modes of the data being generated. Conditional Generative Adversarial Nets (CGAN) [31] introduces the concept that by conditioning the model on additional information, it is possible to direct the data generation process. The objective function of the two-player minimax game is rewritten as:

$$\min_G \max_D V(G,D) = E_{x\sim p_{data}(x)}[logD(x|y)] + E_{z\sim p_z(z)}[log(1-D(G(z|y)))] \tag{2}$$

The Wasserstein GAN (WGAN), introduced by Arjovsky et al. [2] in 2017, is an extension of the original Generative Adversarial network. Instead of using a "Discriminator" to classify or predict the probability of a certain generated event as being real or fake, WGAN introduces the concept of a "Critic" that scores the realness or fakeness of a given event. This change is introduced because while training a generator, we should seek to minimize the distance

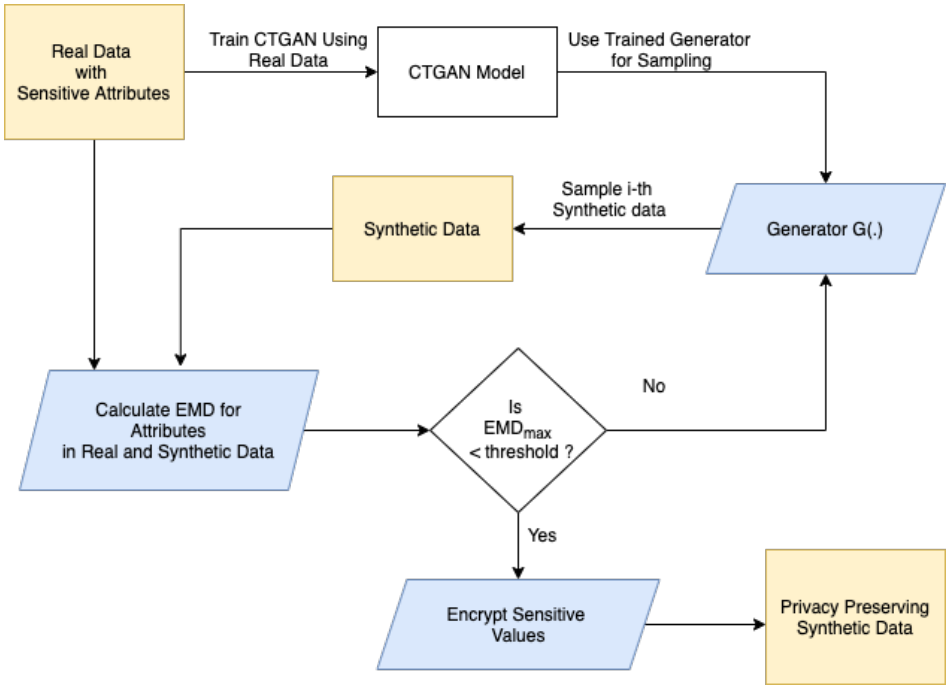


Figure 1: An architecture diagram for our proposed method

between the distribution of the data observed in the training dataset and the distribution observed in the generated samples.

The 2019 paper, "Modeling Tabular Data using Conditional GAN" [45] describes a GAN model that utilizes a conditional generator to address the challenges of a multi-modal continuous variable and imbalanced discrete column in tabular data. It uses a "critic" model that seeks to minimize the distance between generated and real data. CTGAN adds to the idea of TGAN in addressing data imbalance by employing a conditional generator and training-by-sampling method. The proposed model outperforms Bayesian methods on most of the real datasets for several metrics such as likelihood fitness and machine learning efficacy of the synthetically generated data. Due to its ability to produce realistic tabular data that bear statistical resemblance to the original data, we will be using the CTGAN model to generate data in our proposed method.

3 PROPOSED METHOD

In this paper, we propose a novel method of securely sharing sensitive data that utilizes the principles discussed above. In our proposed approach, we first train a CTGAN model using the real data. After training, the generator in the CTGAN model can generate a sample set that can be used instead of the real dataset. To ensure that privacy constraints are met, the distribution of sensitive attributes in the generated set should be within an acceptable distance from its distribution in the original dataset. Finally, to ensure that sensitive values, like IP addresses, from the original dataset are never revealed in the new dataset, selected attributes are encoded using a hashing algorithm. The dataset thus created is available for secure sharing with a remote host. An overview of our proposed approach is described in Figure 1. We go into the detail of each step in our approach further in this section.

3.1 Generation using CTGAN

The vanilla GAN architectures have proven to be successful in generating fake image data. Image data, when normalized, consists of real numbers within a fixed range. The real-valued or continuous variables that represent the image data can be modeled using a Gaussian distribution. This is particularly helpful for high-quality image generation using traditional GANs. Generating images of specific categories is still challenging as there is a high degree of variance associated with specific types of images. However, the central limit theorem suggests most of the real-valued datasets that are formed by random distributions merged together, are some form of Gaussian. In general, pixels' values, as stated in the paper [45], do follow a 'Gaussian-like' distribution when normalized using a min-max normalizer. However, continuous variables in domains other than computer vision, may not always come from a Gaussian-like distribution. A min-max normalization can lead to a vanishing gradient problem. This is because the values can have a much higher range than image pixel data. Some values can be very high, and when a min-max normalization is done, most values of that particular variable will be very close to 0. This highlights the need for a special type of GAN that can handle non-image continuous data as well as other types of discrete data.

The problem CTGAN addresses is that of generating fake tabular data with the help of GANs. In domains such as cybersecurity or finance, we are often faced with modeling datasets that have a mixture of continuous and discrete variables. The method, proposed in the paper [45] , claims to have dealt with the problem of modeling

tabular data that has a mix of continuous and discrete variables. The following steps were used in the paper by CTGAN to work with tabular data.

- *Mode-specific normalization*
- *Conditional Generator*
- *Training by sampling*

CTGAN handles continuous and discrete variables differently. *Mode-specific normalization* is used for the continuous variables. Since this model targets non-Gaussian multimodal continuous variables, the first step is to estimate the number of models for each continuous column. Variational Gaussian Mixture Model (VGMM) [29] is used to estimate the number of modes for each continuous column. The probability of the data coming from the 'k'th mode would be ρ_k described as follows.

$\rho_k = \mu_k \mathcal{N}(c_{i,j}; \eta_k, \phi_k)$

where $c_{i,j}$ is the 'j'th continuous variable of the 'i'th row, η_k is the kth mode and ϕ_k is the standard deviation of the kth mode. The modes are sampled from this probability distribution, and after that they are represented by two vectors α and β. $\beta_{i,j}$ is simply a one-hot vector representing which mode is sampled. For example, if VGMM estimates there are 3 modes, and the second mode is chosen, the vector $\beta_{i,j}$ becomes [0,1,0]. $\alpha_{i,j}$, described below, represents how much the value is separated from the mode.

$\alpha_{i,j} = \frac{c_{i,j} - \eta_2}{4\phi_2}$

α and β for all continuous columns are concatenated together.

The *Mode-specific normalization* described above deals with the multimodal nature of the continuous variables in tabular data. Discrete variables are represented by a one-hot vector encoding of the discrete values $d_{k,j}$. Each row vector r_j is represented by the concatenated vectors of $\alpha_{i,j}$, $\beta_{i,j}$, and $d_{k,j}$, where i ranges for all the continuous variables N_c and k ranges for all the discrete variables N_d. This can be seen in Figure 2. To deal with the discrete variables, a conditional vector is provided to the Generator of the GAN model. The Generator, in this case, becomes a *Conditional Generator* as it takes a condition or a vector as input. In Figure 2, we can see the *condition* as the concatenated vector of D1 and D2. This is also called the 'mask vector' in the CTGAN paper [45]. The *condition* tells us that we are setting a condition for one of the discrete columns to have a predefined value.

Out of all the discrete columns (N_d), one of them is uniformly chosen. Once a discrete column, say D_i, is chosen, a particular discrete value, say k, is sampled with the probability of $log(frequency(D_i = k))$. This helps in constructing the mask vector which forms the *condition* that is provided as an input to the *Conditional Generator*. Each $d_{k,j}$ is converted to empty sequence, except for the one-hot vector of D_i (if D_i is chosen). For the the vector representing D_i, only the kth bit is 1 and the rest are all 0s (if kth discrete value is chosen). In Figure 2, we can see the mask vector of D1 and D2. In this case, D2=1 is the *condition*. D1 has 3 values and D2 has 2 values. So the *condition* becomes a concatenation of [0,0,0] and [1,0].

The loss for training the *Conditional Generator* is cross-entropy loss acting only on the *condition*. This is to ensure that after being trained considerably well, the *Conditional Generator* will be able to produce an output sample that has the value of the discrete variable D2=1, if that is the condition provided. In the next step, the Critic or the Discriminator evaluates the quality of the generated sample. This is done by the method defined as *Training by Sampling*. It is a fairly simple mechanism. From the training set, a sample is chosen that satisfies the condition D2=1, if that was the condition provided to the generator at that iteration. The loss for the Critic of the Discriminator is the Euclidean distance between the sampled data from the training set and the generated sample.

In Figure 2, we can see an example of the algorithm in action.

- After processing the continuous variables with *Mode specific normalization*, a discrete variable is chosen at random with uniform probability. If there are two discrete variables D1 and D2, one of them is sampled.
- From the sampled discrete variable, in this case, D2, a discrete value is sampled with probability proportional to the log of the frequency of the value. This sampled value, in this case, C1, for the discrete variable D2 forms the condition vector.
- The condition vector, as we can see in Figure 2, denotes the value of the discrete variable that is sampled. It is provided as an input to the generator to generate a sample. Cross entropy loss is calculated between the condition and the one-hot encoding of the discrete variables. In the Figure 2, the portion of the vector is denoted by $d_{i,j}$.
- A data point is sampled from the training set, that satisfies the condition that was provided as an input to the generator. In this case, the condition was D2=C1. The discriminator, or the critic, calculates the Euclidean distance between the sampled data from the training set and the generated sample.

3.2 Sampling by EMD

After the Generator in the CTGAN model is trained, it can generate fabricated samples that are close to the real dataset. We want to generate a synthetic dataset that can be shared instead of the real data. For this, we want to sample the Generator model n-times to generate a dataset with n-datapoints. In the generator network model, the probability mass function (PMF) of the discrete variables is close to the PMF of the variable in the original dataset. Hence, when sampling the data this information can be exploited for the benefit of an adversary.

Li et al. [28] discuss the problems of inadvertently disclosing information from such attributes that are *quasi-identifiers*, i.e. the attributes whose values when taken together can potentially identify an individual. There are two types of potential information disclosure in shared data: identity disclosure and attribute disclosure. Identity disclosure occurs when an individual is linked to a particular record in the released data. Attribute disclosure occurs when new information about some individuals is revealed, i.e., the released data makes it possible to infer certain characteristics of an individual more accurately than it would be possible before the data release. Once the identity of an individual is disclosed, the corresponding attributes can be revealed, thus leading to attribute disclosure. It has even been recognized that disclosure of false attribute information can also cause harm [26]. So, even in shared data using completely fabricated samples we want to limit the disclosure risk. To ensure that the quasi-identifiers in the released dataset can be used for identity disclosure, the *t-closeness* principle requires that in an equivalent dataset, the distance between the distribution of a sensitive attribute in this set and the distribution

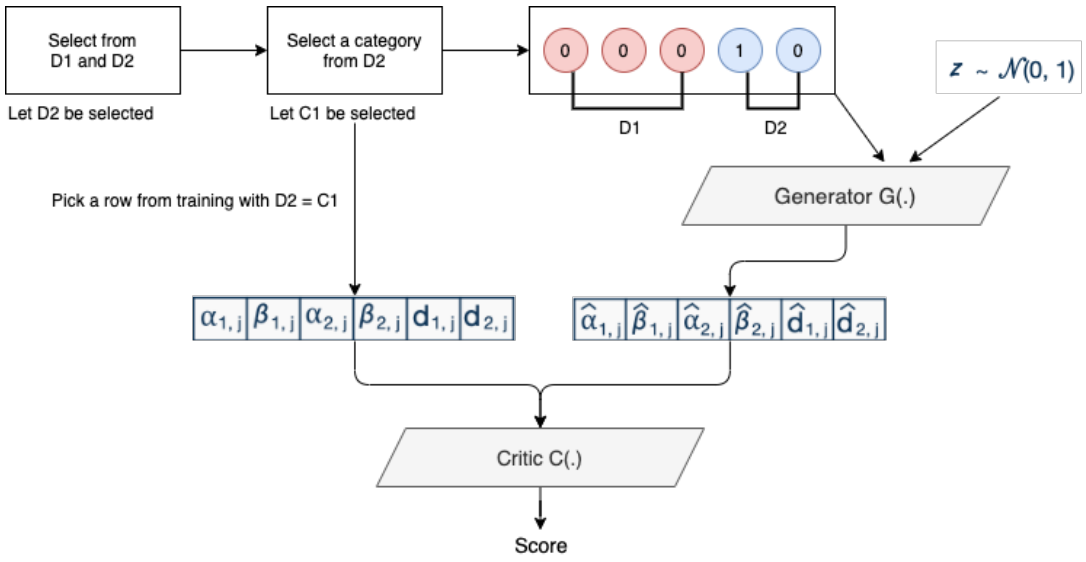


Figure 2: An architecture diagram for CTGAN model

of the attribute in the whole data set is no more than a threshold t. In the original paper by Li et al. [28], Earth Mover's Distance (EMD) is proposed as the distance measure between the distributions. For two distributions **P** and **Q**, the EMD, **D[P,Q]**, can be calculated as follows:

If **P** and **Q** are numerical attributes, let $r_i = p_i - q_i$,(i=1,2,...,m):

$$D[P,Q] = \frac{1}{m-1}(|r_1| + |r_1 + r_2| + ... + |r_1 + r_2 + ...r_{m-1}|) \quad (3)$$

$$= \frac{1}{m-1}\sum_{i=1}^{m}|\sum_{j=1}^{i} r_j| \quad (4)$$

If **P** and **Q** are categorical attributes,

$$D[P,Q] = \frac{1}{2}\sum_{i}|p_i - q_i| \quad (5)$$

We use the privacy principle of *t-closeness* to ensure that our released data has limited disclosure risk. Additionally, the distance measure helps us ensure that the distribution of discrete values in the released dataset bears statistical similarity to the original data. Hence, after the *i*-th sample is drawn from the generator we calculate the EMD for each attribute in the dataset. We keep drawing a sample from the generator, while the maximum EMD for any attribute is greater than the threshold t. Once, the EMD for all attributes is less than or equal to t, t-closeness is achieved and we stop sampling. The sample dataset that has been retrieved, passes on to the next step before being released.

3.3 Encrypting sensitive strings

Our data generator can create data that is very similar to the original without repeating data points from the original dataset. However, the data generator can not fabricate completely new values for some attributes like IP Addresses, which are drawn from a limited space. While the generator hides the correlations that an adversary can discover using traffic analysis, it can't completely hide the range of IP addresses in the data. This is information that a user or individual data source might be unwilling to share with a remote entity or an aggregating learner. To provide extra privacy, we use hashing/encryption here.

Specifically, we use a hashing algorithm to encrypt all sensitive values. This makes it harder to retrieve the original values of attributes from the released dataset. We use the SHA-256 algorithm [41] to encrypt all sensitive values. The benefit of using a standard hashing algorithm is that it scrambles the data deterministically. Hence, data from different sources can be encrypted using the same hashing function and identical values will have the same hashed value in the released data. We note that probing attacks for IP addresses hashed using SHA256 are not trivial, and more complex encryptions can be used if needed. Additionally, the hashing function accepts an input of arbitrary length and outputs a fixed-length result that makes sharing data over communication channels easier. After the sensitive values have been encrypted, the dataset is secure and can be released for learning with minimal risk to privacy.

4 EXPERIMENTS AND RESULTS

4.1 Datasets

In this section, we discuss the experiments on our proposed framework. We aim to find a secure way to release sensitive data, such that analytical tasks have the same results for the released dataset as they would have for the original data. To prove, the validity of our proposed framework, we use well-known datasets that are frequently used for machine learning-based classification tasks. We will generate a securely release-able version of each of the datasets and show that classification algorithms have comparable results for real data and data released securely through our framework. We will specifically use this framework to provide a secure and privacy-preserving way of releasing network event data. For the original dataset, we use the Network Event Data released as a part

	alert_ids	notified	categoryname	ip	alerttype	dstip	srcport	dstport	severity	username	signature
Real	wyX	0	Exploit	YW.RN.220.183	ThreatWatch Outbound	YT.LB.34.21	21384	443	4	1	1
Generated	wyX	0	Exploit	YW.RN.220.183	ThreatWatch Outbound	UJ.TZ.210.230	56511	443	2	1	1
Real	WYW	1	Exploit	YT.PK.194.174	IDPS Alert	YT.PK.194.174	49962	80	4	1	1
Generated	WYW	1	Exploit	YT.PK.194.174	Suspicious SMB Activity	YT.PK.194.174	61687	445	4	0	1

Table 1: A comparison of real datapoint from Network Event Data and generated datapoint from CTGAN generator

Dataset	# of Continuous attributes	# of Discrete attributes	# of Instances
Census Income	6	9	48842
Contraceptive Method Choice	2	8	1473
Credit Approval	10	6	690
Pima Indian Diabetes	8	1	768
Iris Flower	3	1	150

Table 2: Number of Attributes and Instances in the Datasets used for our experiments

of BigData 2019 Cup [21] that was used for false alarm identification. We go into the details of the dataset used for our experiments, below.

Network Event Data:"IEEE BigData 2019 Cup: Suspicious Network Event Recognition" [21] was a data mining competition organized jointly by companies Security On-Demand (SOD) and QED Software at the KnowledgePit online platform, in association with the IEEE BigData 2019 conference. In this challenge, participants were allowed to explore the network traffic data provided by SOD. This data contains threat watch alerts identified by SOD's systems and investigated by the SOC team members during six months. Network event alerts were represented by the data retrieved at three different processing stages, raw information extracted in form of network event logs, a series of localized alerts that usually correspond to single network connections assessed as suspicious according to rules designed by the SOD's Threat Reconnaissance Unit, and threat watch alert corresponding to time-interval-specific aggregations of localized alerts, which are finally investigated by security operators and analysts. The task in this competition was to come up with an efficient scoring model that could discern among truly meaningful threat watch alerts and false alarms.

In the data acquisition stage, specific attributes (e.g., device_type, reporting_device_code, device_vendor_code), were identified as sensitive and the organizers had to anonymize the dataset before releasing it. The preparation of such dictionaries required processing a very large data set of raw event logs – its size was over five times greater than the size of the final event logs provided to participants of the challenge ($\tilde{2}$.2T B). While this was sufficient for this competition, this is not a practical solution for data sharing. Cybersecurity investigators and machine learning researchers require large volumes of data, like the one released in the BigData 2019 Cup. We need a more practical and secure solution for persistent data sharing. As seen from the results of the BigData 2019 Cup, machine learning models provide an efficient and accurate solution to false alert identification that can considerably reduce the need for human intervention. Hence, it is worth investigating a secure solution to fast data release, such that more meaningful investigations can be carried out.

We use our proposed methodology to generate a dataset from the original data set of this challenge. The generated data is safe for release and contains the privacy guarantees we have described earlier. We will show that machine learning models used for false alarm identification on the original dataset have similar performance on the new dataset. Thus showing that the secure dataset generated from our framework can be used instead of the original dataset for analytical purposes.

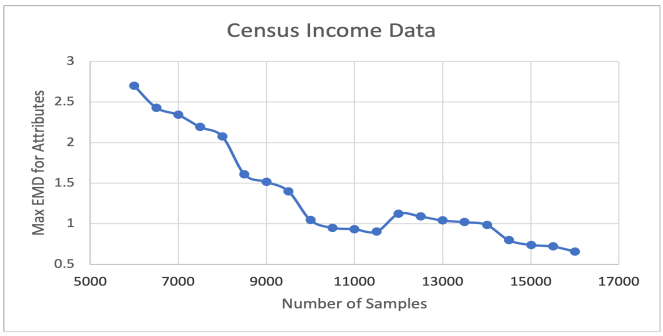
Other Datasets: To prove that our framework can be generalized to other datasets, we apply our proposed method to other well-known tabular datasets that are often used for machine learning classification tasks. The datasets are collected from the UCI Machine Learning Repository [14]. The datasets used for our experiments are:

- **Adult Census Income Data** [23], the classification task is to predict whether income exceeds 50K/yr based on census data.
- **Contraceptive Method Choice Data** [11], the classification task is to predict contraceptive method used in a family.
- **Credit Approval Data** [37], the classification task is to predict if the credit card application will be approved.
- **Pima Indians Diabetes** [40], the classification task is to predict the likelihood of diabetes in a Pima Indian female.
- **Iris Flower Data** [17], the classification task is to predict the variety of Iris flower.

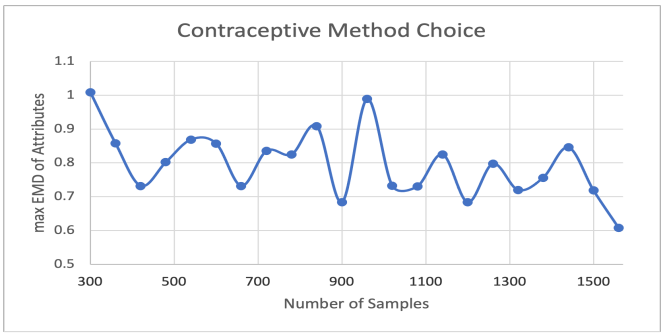
All datasets contain a mix of continuous and categorical variables. The details are provided in Table 2.

4.2 Generating Privacy-Preserving Synthetic Data

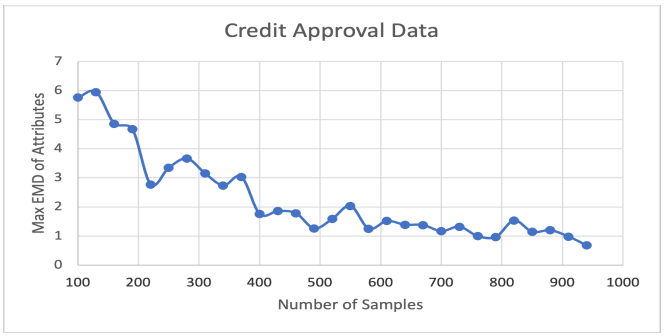
We use our framework to generate privacy-preserving synthetic data that can replace tabular datasets for machine learning models. An overview of the data generation pipeline is given in Figure 1. The pipeline broadly involves the following steps:



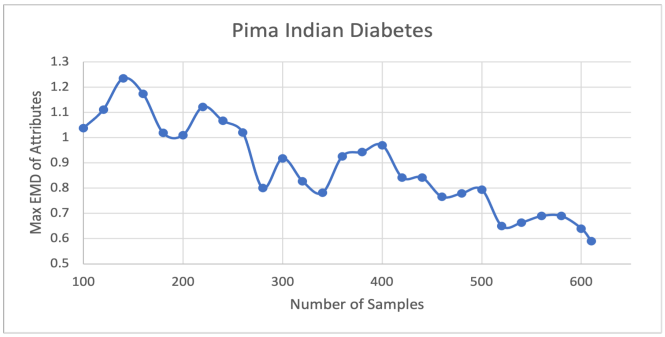
(a) Census Income Data



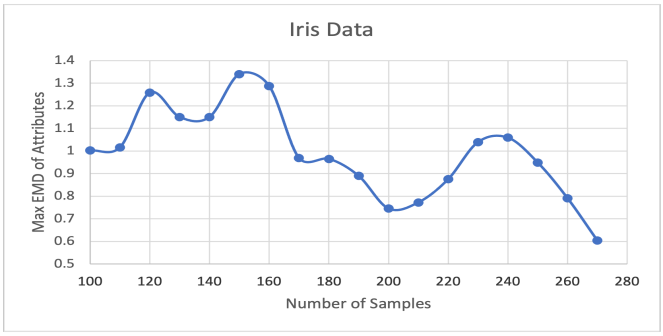
(b) Contraceptive Method Choice Data



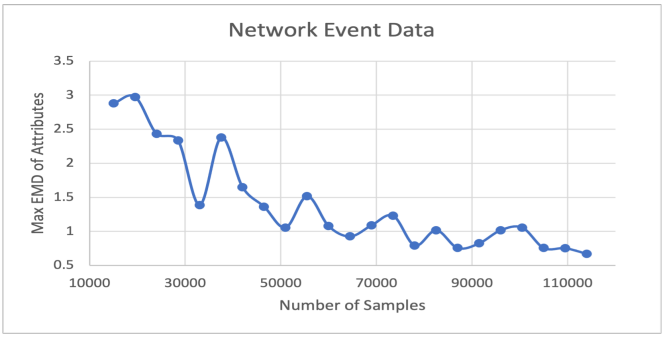
(c) Credit Approval Data



(d) Pima Indians Diabetes Data



(e) Iris Data



(f) Network Event Data

Figure 3: Plot of Max EMD of attributes vs Number of Generated Samples in different datasets

(1) Train a CTGAN model with the real dataset. Once trained, the generator of the CTGAN should generate data points close to the original data. The CTGAN constitutes of two components, a Generator, and a Critic or a Discriminator. The Generator has an input layer that is a concatenation of two vectors. The first vector is a 128 dimensional sampled vector from a Normal Distribution. The second vector is a one-hot vector of the same dimension of the conditional vector for the discrete data. This vector is dependent on the dataset. For the network event dataset, the dimension is 2995, which is the total number of discrete values in the dataset. The generator has two more fully connected layers of dimension 256. Each of these layers has ReLU activation, and is followed by a BatchNormalization layer. The final layer is of the dimension of the input data.

The input data dimension is dependent on the number of continuous variables, the maximum number of modes of the continuous variables, the number of discrete variables, and the possible number of discrete values for each of the variables. In the network event dataset, there are 10 continuous variables and 18 discrete variables. We also consider the maximum number of modes to be 10. The input dimension is the sum of 2* number of continuous columns * maximum number of modes and the total number of discrete values for all variables. For the network event data, the input dimension will be $20 * 10 + 2995 = 3195$. The input to the Discriminator is a vector of the same dimension. It is followed by 2 fully connected 256 dimensional layers. The layers have LeakyReLU activation. The dimension of the output is 1.

	Competition Winner	Baseline Classifier trained on Original Data	Classifier trained on Generated Data
AUC Score	0.93	0.92	0.91
Macro-Avg. of Precision	0.95	0.88	0.87
Macro-Avg. of Recall	0.92	0.98	0.98
Macro-Avg. of F-1 Score	0.93	0.92	0.92
Accuracy	0.96	0.95	0.95

Table 3: Comparison of Network Event Recognition task for Classifiers trained on Original Data and Generated Data

For reference, a subset of a row from the original network event data and the generated dataset is provided in Table 1.

(2) We iteratively sample data points from the CTGAN generator, till the EMD for all attributes in the generated dataset is within a threshold, T from the original dataset. Though there is no hard rule for choosing T, the distance should ideally be less than 1. For our experiment, we chose the value 0.6 for T. Since it is less than 1, it is sufficient to meet privacy requirements. Additionally in experiments over multiple datasets of varying sizes, we observe that this threshold value can be reached within approx. 50 iterations of sampling, with each iteration sampling 1/10-th the size of the dataset.
The change of maximum EMD for any attribute with the number of samples is plotted in Figure 3 for reference. We also ensure that no row from the original dataset is fully repeated in the generated dataset.

(3) The sensitive values in the generated dataset are encrypted using the SHA-256 algorithm. This is specific to the dataset being used. For the network event data, the organizers of the challenge specify that the IP addresses require special anonymization. We encrypted these fields in our dataset. For the other datasets, we encrypted all field values containing strings or special characters.

We compare the impact of these synthetic datasets on machine learning tasks.

4.3 Testing with ML models

To show that data generated through our framework can replace real data for machine learning tasks, we use it to train ML classifiers. First, we train a standard classifier using the original data. A subset of the original data is reserved for testing and not used in training. Then, we train another classifier (same hyperparameters) with just the generated dataset. We use the first classifier to predict the label for the test set reserved from the original data. We use the second classifier to also predict the label for the same test set. We calculate all validation metrics for both classifiers and compare the results.

Network Event Data: "IEEE BigData 2019 Cup: Suspicious Network Event Recognition" [21] challenge released Network Event dataset in association with SOD. The objective of the challenge was to efficiently discern among truly meaningful threat watch alerts and false alarms. Though the dataset was anonymized before being released, the anonymization task was costly and can not be generalized. We show here that the network event data anonymized through our framework has a similar performance as the original data for the network event recognition task.

We use an XGBoost classifier with Grid Search hyperparameter tuning for network event recognition on the original dataset. We first train the classifier on a subset of the original dataset, *training data* and test it on a subset of the original dataset, *test data*. We then train another classifier with the same hyperparameters on the dataset generated from our framework. We compare the scores in Table 3. For reference, we also provide the performance of the classifier model that won the competition. We tried to replicate the winning Machine Learning model from the description given. The baseline classifier we have used is slightly different but comparable in results.

Dataset	Baseline Accuracy on Original Data	Accuracy on Released Data	# of Samples in Released Data
Census Income	0.80	0.76	16700
Contraceptive Method Choice	0.53	0.52	1560
Credit Approval	0.86	0.79	930
Pima Indian Diabetes	0.75	0.75	610
Iris Flower	0.98	0.96	270

Table 4: Comparison of Accuracy in Original Data vs Data generated by our framework

Other datasets: To show that our model can be generalized to any standard tabular dataset, we apply it to other standard tabular datasets from the UCI Machine Learning repository. We use the XGBoost classifier for classification tasks described for each dataset. We first train the classifier on a subset of the original dataset, *training data*. We measure its performance by predicting the class labels of the remaining sample points in the dataset, the *test data*. We then train another classifier with the same hyperparameters on the data generated by our framework. We measure its performance by predicting the class labels of the same *test data* used for testing the original training set. We then compare the performance of the two classifiers. The baseline accuracy on the original data vs the accuracy on the data released through our framework has been described in Table 4. We also report the number of generated samples that were used.

4.4 Summary of Results

For all standard tabular datasets, there is minimal loss in accuracy for the classifier when trained on data generated through our framework. The average loss in accuracy for the six datasets used in our experiment is 3%. For the network event data, the AUC score for the classifier trained on the generated dataset is 0.91, which is a small difference from our baseline classifier score (0.92) and also comparable to the winner of the challenge (0.93). There is a small drop in Precision and F1-score, the recall is the same for both.

Overall, the classifiers trained on the generated datasets have comparable performance as the baseline classifiers. Thus, our framework provides a secure solution for releasing data with sensitive information that has minimal impact on learning tasks.

5 RELATED WORK

Privacy in data sharing has been heavily discussed in the past decade. Organizations are invested in finding secure, automated solutions to collecting and sharing data [25], [24]. However, organizations are still sceptical about sharing their data for use in research. Network data is useful for multiple purposes, the most common being creating Intrusion Detection Systems (IDS). Previous methods for IDS like Snort [18], uses rule-based methods for matching possible attacks with known attacks. The main drawback of rule-based methods is that it does not perform as well for novel attacks that bypass the existing rules used for detection. For this reason, ML-based methods are being used currently because the rules created by them are often more complex than human-designed rules. Lee et al. described methods to collect features for network events for ML analysis [27]. There are multiple supervised methods that work on IDS [32, 47]. Unsupervised methods, using deep learning models have also been useful in detecting cyber-threats involving network data [36, 42]. Reinforcement Learning-based methods are also becoming popular in detecting cyber-threats and malware [33, 35, 44]. Some researchers have shown that Random Forests outperform other state-of-the-art algorithms for Network IDS tasks [5]. However, all machine learning-based algorithms for network data analysis rely heavily on large volumes of network data being available for analysis. This is not practical. Network Data contain sensitive information that requires added privacy measures.

There has been some work done on the dataset that we have used [21] for network data sharing. Specifically, in this work, GANs were used for adversarial training and the discriminator was used as a classifier [34]. This was done to deal with the class imbalance problem that is common across ML algorithms working on network event data. GANs have been used to generate samples that are useful for other ML models [13]. The first paper talked about fully connected GANs [19]. Soon, Deep Convolution GANs [38] became state-of-the-art for GANs in image data. A problem that needed to be addressed for GANs was that of being unable to generate class-specific data. Conditional GANs [31] helped in the generation of class-specific data with the help of an additional class label for both the discriminator and the generator. InfoGAN [7], also helps in this regard by providing class-specific codes along with the noise vectors as an input to the generator. An additional neural network is required to estimate the posterior probability of the generated sample for the class

6 CONCLUSION

As machine learning models have grown in efficiency over the years, there is a gap between the computational capabilities of machine learning models and data available for use. For many sensitive domains, corporate and government entities are unwilling to share their data for use in ML research, despite the obvious benefits. Alleviating privacy concerns will go a long way in making more data available, thus benefiting researchers at large. This is especially true for cybersecurity. Organizations that investigate cyber attack events are reluctant to share their network-related data, even though they can greatly help not just cybersecurity research, but prevention of similar attacks in other places. We need techniques that help make such data widely available for accurate data analysis without violating the privacy of the data subject. Though privacy in data sharing has been frequently being discussed, there is still no perfect solution.

In this paper, we explored the approach of making data available for analysis without sharing sensitive information is by replacing sensitive information with synthetic data that behaves the same for machine learning tasks. We described a framework to generate such synthetic data using a combination of CTGANs, t-closeness measures, and hashing. We then tested a model trained on our synthetic data against the actual test data from the original dataset. We showed that for a variety of datasets selected from the UCI repository, our approach led to only marginal decreases in the accuracy measures of the classifier. We also tested our approach on an actual dataset of attack-related network events released as a part of the BigData 2019 Cup. Our experiments show that synthetically generated is analytically similar to the original dataset, dropping the AUC score from 0.92 to 0.91. This provides a privacy-preserving and secure alternative to sharing real tabular data. We hope that this could help organizations and individuals to release their sensitive data for academic research.

Kaggle has very recently released a medical dataset [22] as part of their Tabular Playground Series. They did not release the original dataset, rather a synthetic version was generated using a CTGAN. Though they do not completely describe their anonymization process, it is interesting that data organizations are invested in the secure release of data for public use using approaches similar to the one we have proposed.

REFERENCES

[1] 2013. General Data Protection Regulation (GDPR) – Official Legal Text. https://gdpr-info.eu/. (Accessed on 02/24/2021).

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*. PMLR, 214–223.

[3] Clark D Asay. 2012. Consumer Information Privacy and the Problems (s) of Third-Party Disclosures. *Nw. J. Tech. & Intell. Prop.* 11 (2012), xxxi.

[4] Brooke Auxier, Lee Rainie, Monica Anderson, Andrew Perrin, Madhu Kumar, and Erica Turner. 2019. Americans and Privacy: Concerned, Confused and Feeling Lack of Control Over Their Personal Information | Pew Research Center. https://pewrsr.ch/3ywnkLl. (Accessed on 02/24/2021).

[5] Mustapha Belouch, Salah El hadaj, and Mohamed Idhammad. 2018. Performance evaluation of intrusion detection based on machine learning using Apache Spark. *Procedia Computer Science* 127 (01 2018), 1–6. https://doi.org/10.1016/j.procs.2018.01.091

[6] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2018. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096* (2018).

[7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information

maximizing generative adversarial nets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2180–2188.

[8] Federal Trade Commission et al. 2016. Children's online privacy protection rule ("coppa"). *Retrieved on September* 16 (2016).

[9] PCI Security Standards Council. 2016. Official PCI Security Standards Council Site - Verify PCI Compliance, Download Data Security and Credit Card Security Standards. https://bit.ly/3ywnthR. (Accessed on 02/24/2021).

[10] Ashish Dandekar, Remmy AM Zen, and Stéphane Bressan. 2018. A comparative study of synthetic dataset generation techniques. In *International Conference on Database and Expert Systems Applications*. Springer, 387–395.

[11] DHS. 1987. National Indonesia Contraceptive Prevalence Survey 1987 - Summary Report. https://dhsprogram.com/pubs/pdf/SR9/SR9.pdf. (Accessed on 10/15/2021).

[12] Josep Domingo-Ferrer and Jordi Soria-Comas. 2015. From t-closeness to differential privacy and vice versa in data anonymization. *Knowledge-Based Systems* 74 (2015), 151–158.

[13] Georgios Douzas and Fernando Bação. 2017. Effective data generation for imbalanced learning using Conditional Generative Adversarial Networks. *Expert Systems with Applications* 91 (09 2017). https://doi.org/10.1016/j.eswa.2017.09.030

[14] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[16] Khaled El Emam and Fida Kamal Dankar. 2008. Protecting privacy using k-anonymity. *Journal of the American Medical Informatics Association* 15, 5 (2008), 627–637.

[17] Ronald A Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics* 7, 2 (1936), 179–188.

[18] Akash Garg and Prachi Maheshwari. 2016. Performance analysis of Snort-based Intrusion Detection System. (01 2016), 1–5. https://doi.org/10.1109/ICACCS.2016.7586351

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[21] Andrzej Janusz, Daniel Kałuza, Agnieszka Chądzyńska-Krasowska, Bartek Konarski, Joel Holland, and Dominik Ślęzak. 2019. IEEE BigData 2019 cup: suspicious network event recognition. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 5881–5887.

[22] kaggle.com. 2021. Tabular Playground Series - Oct 2021 | Kaggle. https://www.kaggle.com/c/tabular-playground-series-oct-2021. (Accessed on 10/15/2021).

[23] Ron Kohavi et al. 1996. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid.. In *Kdd*, Vol. 96. 202–207.

[24] Anantaa Kotal, Anupam Joshi, and Karuna Pande Joshi. [n. d.]. The Effect of Text Ambiguity on creating Policy Knowledge Graphs. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 1491–1500.

[25] Anantaa Kotal, Karuna Pande Joshi, and Anupam Joshi. 2020. ViCLOUD: Measuring Vagueness in Cloud Service Privacy Policies and Terms of Services. *UMBC Information Systems Department* (2020).

[26] Diane Lambert. 1993. Measures of disclosure risk and harm. *JOURNAL OF OFFICIAL STATISTICS-STOCKHOLM-* 9 (1993), 313–313.

[27] Wenke Lee and Salvatore Stolfo. 2000. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security, Vol. 3, No. 4, Pages 227–261* (2000).

[28] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. 2007. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*. IEEE, 106–115.

[29] Christopher M Bishop. 2006. Pattern recognition and machine learning. *Springer* (2006).

[30] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. 2007. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 3–es.

[31] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[32] Saurabh Mukherjee and Neelam Sharma. 2012. Intrusion Detection using Naive Bayes Classifier with Feature Reduction. *Procedia Technology* 4 (12 2012), 119–128. https://doi.org/10.1016/j.protcy.2012.05.017

[33] Thanh Thi Nguyen and Vijay Janapa Reddi. 2019. Deep reinforcement learning for cyber security. *arXiv preprint arXiv:1906.05799* (2019).

[34] Aritran Piplai, SSL Chukkapalli, and Anupam Joshi. 2020. NAttack! Adversarial Attacks to bypass a GAN based classifier trained to detect Network intrusion. In *6th IEEE International Conference on Big Data Security on Cloud*. IEEE.

[35] Aritran Piplai, Priyanka Ranade, Anantaa Kotal, Sudip Mittal, Sandeep Nair Narayanan, and Anupam Joshi. 2020. Using Knowledge Graphs and Reinforcement Learning for Malware Analysis. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2626–2633.

[36] Guo Pu, Lijuan Wang, Jun Shen, and Fang Dong. 2020. A hybrid unsupervised clustering-based anomaly detection method. *Tsinghua Science and Technology* 26, 2 (2020), 146–153.

[37] J Ross Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.

[38] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434* (2015).

[39] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[40] Jack W Smith, James E Everhart, WC Dickson, William C Knowler, and Robert Scott Johannes. 1988. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care*. American Medical Informatics Association, 261.

[41] Secure Hash Standard. 2009. The cryptographic Hash algorithm family: Revision of the secure Hash standard and ongoing competition for new hash algorithms. (2009).

[42] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. 2017. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

[43] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. 2018. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8798–8807.

[44] Liang Xiao, Xiaoyue Wan, Canhuang Dai, Xiaojiang Du, Xiang Chen, and Mohsen Guizani. 2018. Security in mobile edge caching with reinforcement learning. *IEEE Wireless Communications* 25, 3 (2018), 116–122.

[45] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *arXiv preprint arXiv:1907.00503* (2019).

[46] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. 2017. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 5907–5915.

[47] Jiong Zhang and Mohammad Zulkernine. 2005. Network Intrusion Detection using Random Forests.. In *Pst*. Citeseer.