USING MACHINE LEARNING TECHNIQUES FOR PREDICTION AND DATA GENERATION WITH APPLICATIONS TO DATA PRIVACY

by

Nazmiye Ceren Abay

APPROVED BY SUPERVISORY COMMITTEE:

Dr. Bhavani Thuraisingham, Co-Chair

Dr. Murat Kantarcioglu, Co-Chair

Dr. Latifur Khan

Dr. Yulia R. Gel

Dedicated to my family.

USING MACHINE LEARNING TECHNIQUES FOR PREDICTION AND DATA GENERATION WITH APPLICATIONS TO DATA PRIVACY

by

NAZMIYE CEREN ABAY, BS, MS

DISSERTATION

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

December 2019

ACKNOWLEDGMENTS

During my PhD years at UT Dallas, I have been supported by many wonderful people. I would like to express my sincere appreciation to all friends and colleagues.

Firstly, I would like to express my deepest appreciation and gratitude for my co-advisor, Dr. Bhavani Thuraisingham, for her continuous encouragement, support, and generosity. She has been supportive of my career goals and she has provided me extensive guidance both in research and my personal life in general. As my co-advisor and mentor, she has taught me more than I could ever give her credit for here.

I also thank my co-advisor, Dr. Murat Kantarcioglu, for his practical suggestions to my research problems. I would like to also extend my deepest appreciation to Dr. Cuneyt Akcora and Dr. Yan Zhou. I am truly grateful for all the times you gave me help, knowledge, and unlimited support. I also thank my committee members, Dr. Latifur Khan, and Dr. Lawrence Overzet, for contributing their time generously in serving on my proposal and dissertation defense committees. l am also very grateful to Mrs. Rhonda Walls for her patience, support and help.

I would like to extend my sincere thanks to Dr. Ismail Hakki Toroslu, Dr. Pinar Karagoz and Dr. Alev Mutlu for the continuous support of my MS study, for their patience, motivation, enthusiasm, and immense knowledge. I am very fortunate to have learned from such inspirational professors.

I am most thankful for my family. I am happy to accomplish this dream for you. It would be very difficult for me to overcome many challenges without my husband, Bulut. I am thankful to Bulut for believing in me and being there for me anytime I need him.

Last but not least, I would like to thank my friends, Aref Asvadishirehjini, Yasmeen Alufaisan, Vibha Chandramouli Belavadi, Maryam Imani, Imrul Anindya, and Mustafa Ozdayi, for simply making me a happier person. I am very fortunate to have such great friends.

The research reported herein was supported in part by NIH award 1R01HG006844, NSF awards CICI-1547324 and IIS-1633331.

October 2019

USING MACHINE LEARNING TECHNIQUES FOR PREDICTION AND DATA GENERATION WITH APPLICATIONS TO DATA PRIVACY

Nazmiye Ceren Abay, PhD
The University of Texas at Dallas, 2019

Supervising Professors: Dr. Bhavani Thuraisingham, Co-Chair
Dr. Murat Kantarcioglu, Co-Chair

Increasingly, machine learning (ML) applications are developed and become an integral part of many real-world applications. Especially, ML techniques are heavily used in research and industry to help make effective decisions. Despite the apparent recent success of ML techniques, there exist some domain-specific challenges that require in-depth investigations with respect to predictive accuracy, privacy protection and cybersecurity.

In this dissertation, we start with understanding the usability of ML techniques in the cryptocurrency transaction domain (e.g., Bitcoin) where there is no privacy concern (i.e., all Bitcoin transaction information is public) and show how to use ML techniques to make better predictions in real-time.

For application domains that involve sensitive data, collecting, sharing and refining of these sensitive data may raise serious privacy concerns. To address these concerns, we propose a privacy preserving synthetic data generation technique that leverages deep learning. The proposed technique allows participants to share the synthetic datasets freely without worrying about the individual privacy. Furthermore, we compare our proposed technique with the existing synthetic data generation algorithms, and investigate the utility of these algorithms under different use cases.

Finally, we explore the usage of the generated synthetic data to improve the cybersecurity posture of the organizations. Basically, we show that the generated synthetic data not only protect individual privacy but can be used to deceive (i.e., the synthetic data is indistinguishable from the real data) the potential cyberattackers. This in return could be used to reduce sensitive data leakage under successful cyberattacks where an attacker could be deceived to target synthetic data instead of the real, and sensitive data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Machine Learning (ML) has a profound influence on a wide range of applications that involve analyzing and generating data in many emerging domains. Companies increasingly rely on machine learning based software applications to make better decisions and predictions. The essence of machine learning techniques is analyzing, detecting, and interpreting patterns and structures in data to enable prediction, reasoning, and decision making without human intervention. As machine learning techniques continue to advance, they are increasingly deployed in critical real-world applications such as financial market, security, and health care domains. However, careless usage of ML techniques may cause inaccurate prediction or inherent risk to individual privacy. Depending on the application domain, there are different concerns that need to be addressed, such as predictive accuracy, privacy protection and cybersecurity.

This dissertation provides solutions to the aforementioned concerns regarding the use of machine learning models. We first propose to apply machine learning models in the cryptocurrency domain where all the data is public and there is no privacy concern. The goal is to make accurate predictions of the cryptocurrency price that is highly volatile. Next, for domains where ML techniques need to access sensitive data, we show how to develop ML based techniques, more specifically deep learning techniques, to generate synthetic data that is suitable for public sharing for ML while preserving individual privacy. Finally, we present data deception techniques for cybersecurity systems that can be used to generate "honeyfiles" to fool and track potential attackers.

Our first challenge is to apply machine learning models in real-time cryptocurrency analysis for predicting Bitcoin prices. Bitcoin is a cryptocurrency, a type of digital cash. It is a decentralized currency without any central authorities that can be sent from user-to-user on top of the peer-to-peer Bitcoin network. For Bitcoin, each transaction is recorded on a

distributed public ledger called blockchain. The transactions recorded on the blockchain can be accessed and analyzed publicly. Furthermore, all of the transactions could be represented by a graph which we refer to as the "blockchain graph". Unlike other financial networks, such as stock and currency trading, blockchain based cryptocurrencies have the entire transaction graph accessible to the public (i.e., all transactions can be downloaded and analyzed).

In this real-time cryptocurrency predictive model, we investigate the impact of the blockchain graph structure on cryptocurrency price by proposing different approaches to representing blockchain graph patterns; and we use these patterns to build machine learning models for price prediction. Bitcoin networks have been studied heavily to quantify the dynamics of the Bitcoin transaction network. Existing approaches leverage global blockchain graph structure to extract traditional graph features such as degree distribution, motif counts and clustering coefficients, and use these graph features in machine learning models such as random forest for assessment of their utility in price forecasting. As already observed by previous studies (e.g., (Swanson, 2014; Greaves and Au, 2015)), and also confirmed by our experimental results, these standard global graph based features fail to capture important properties such as transaction volumes, transaction amounts, and their relationships with the underlying local graph structure. Since these basic approaches do not provide conclusive insights into the blockchain graph dynamics and its impact on cryptocurrency price, we propose novel techniques inspired by topological data analysis and, particularly, persistent homology that can capture these higher order interactions. Here, we introduce the power of topological data analysis, particularly, persistent homology and associated network filtrations, to the analysis of blockchain graphs. Our experiment results show that the proposed persistent homology based machine learning models can significantly outperform (i.e., up to 38% improvement in root mean squared error) models that use only past price and standard features such as total transaction count. Finally, using these insights, we propose a cryptocurrency predictive model that can identify the financial market changes earlier with persistent homology.

For domains where ML techniques need to access sensitive data, collection and sharing of sensitive data with the third parties might cause privacy leakage. To address this privacy challenge, solutions have been proposed in two broad categories. In the first category, the data anonymization based approaches (e.g., (Sweeney, 2002)) try to use various definitions to sanitize data so that it cannot be easily re-identified. Although these approaches have some important use cases, they are not usually based on rigorous privacy definitions that can withstand various types of re-identification attacks. In the second category, synthetic data generation approaches have been proposed to generate realistic synthetic data using rigorous differential privacy definition (Dwork et al., 2014). Although these approaches have been shown to work in some limited cases, they have not been extensively tested on different types of use cases with different requirements (e.g., high dimensionality, correlation among features). Therefore, it was not clear which technique works well under what conditions for what type of data sets. We answer these questions by conducting extensive experimentation. Furthermore, we provide a new differentially private deep learning based synthetic data generation technique to address the limitations of the existing techniques. Our aim is to provide data generation tool for ML applications that protects both the privacy of individual and high data utility in terms of different ML metrics.

We also investigate the utility of data generation under the context of the cyber deception, to show "deceptivity" of the generated data when an adversary has some insight about the real data. To achieve this, honeypots are used to simulate the test cases. Creating deceptive data (i.e., HoneyFiles) has many challenges. For different settings, we may need different types of HoneyFiles. For example, to deceive an attacker and feed false information, deceptive technical plans (e.g., technical drawings of an airplane) could be generated. On the other hand, to make HoneyFiles more believable, fake text data could be added to such files. Since addressing all these different types of data require different techniques, we focus on generating deceptive HoneyFiles using our privacy preserving data generation tool,

and evaluate "deceptivity" rate of the generated data. Still, generating realistic relational HoneyFiles while not disclosing sensitive information is a significant challenge. In this part of the dissertation, we evaluate the effectiveness of relational HoneyFiles on real datasets, and show under what conditions differentially private deep learning techniques could be used to generate relational HoneyFiles.

The remainder of this dissertation is organized as follows. We start by discussing the related work to the problems we investigate in this dissertation in Chapter 2. After that, we review the tools and the techniques that we use in this dissertation in Chapter 3. In Chapter 4, we investigate the predictive accuracy of the machine learning techniques in cryptocurrency system, Bitcoin. Chapter 5 proposes the privacy preserving data generation algorithm for limiting the leakage of individual privacy. In Chapter 6 we present the decoy data generation algorithm for cybersecurity systems to enchance the computer security. Finally, in Chapter 7 we conclude the dissertation.

CHAPTER 2

RELATED WORK

In this chapter, we summarize the literature work related to the topics we studied in this dissertation. In Section 2.1 we review the works on Blockchain Networks on machine learning models. Section 2.2 contains an overview of different privacy preserving data generation techniques. In Section 2.3 we describe the previous works on decoy file generation for honeypots.

2.1 Blockchain Networks

The success of Bitcoin (Nakamoto, 2008) has encouraged hundreds of similar digital coins (Tschorsch and Scheuermann, 2016). The underlying Blockchain technology has been adopted in many use cases and applications. With this rapidly increasing activity, there have been numerous studies analyzing the blockchain technology from different perspectives.

The earliest works aimed to track the transaction network to locate coins used in illegal activities, such as money laundering and blackmailing (Androulaki et al., 2013; Ober et al., 2013). These findings are known as the taint analysis (Di Battista et al., 2015).

The Bitcoin network itself has also been studied from multiple aspects. Dyhrberg (Dyhrberg, 2016) studied Bitcoin's similarities to gold and the dollar, finding hedging capabilities and advantages as a medium of exchange. From a graph perspective, Baumann et al. (Baumann et al., 2014) analyzed centralities, and (Lischke and Fabian, 2016) found that since 2010 the Bitcoin network can be considered a scale-free network. Furthermore, (Kondor et al., 2014) tracked the evolution of the Bitcoin transaction network, and modeled degree distributions with power laws. Although these studies analyzed the Bitcoin graphs, the primary focus was on global graph characteristics.

Kristoufek (Kristoufek, 2015) analyzed potential drivers of Bitcoin prices, such as the impact of speculative and technical sources. A number of recent studies show the utility of

global graph features to predict the price (Kondor et al., 2014; Greaves and Au, 2015; Madan and Zhao, 2015). For instance, (Sorgente and Cibils, 2014) studied the impact of average balance, clustering coefficient, and number of new edges on the Bitcoin price. These findings suggest that certain network features are correlated with price; for example, the number of transactions put into a block indicates a price increase.

Community detection on weighted networks (Jog and Loh, 2015) has not been applied to blockchains yet, but two network flow measures were recently proposed by (Yang and Kim, 2015) to quantify the dynamics of the Bitcoin transaction network and to assess the relationship between flow complexity and Bitcoin market variables. Furthermore, (Madan and Zhao, 2015) identified 16 features (e.g., number of Tx) for 30, 60 or 120 minute intervals and used random forest models to predict the price. The core idea behind all these approaches is to extract certain global network features and to employ them for predictions. However interactions of features (Henelius et al., 2016) are not widely studied. Most recently, (Akcora et al., 2018) introduced the notion of *chainlet* motifs to understand the impact of local topological structures on Bitcoin price dynamics, and showed that employing aggregated chainlet information leads to more competitive price prediction mechanisms. In contrast to global network features, chainlets provide a finer grained insight at the network transactions. In practice, chainlets can be used to refine the above-mentioned models. However, the chainlet approach is limited to analysis of transaction types and does not account for critical information such as the transferred amounts.

In Chapter 4, we remedy some of these short comings using persistent homology based features, which bring significant performance gains over the results reported in (Akcora et al., 2018).

2.2 Differentially Private Data Generation Models

Extensive research has been conducted on publishing private data for preserving privacy. In this section, we discuss the related techniques with their strengths and limitations.

In statistical analysis, publishing a marginal table while preserving the privacy has been a fundamental research goal. One of the initial efforts in addressing this problem is proposed by Barak et al. (Barak et al., 2007). In this method, a full contingency table constructed on the original data is represented by the Fourier coefficients. The noise is then added to these coefficients in order to construct the desired k-way marginal tables, instead of perturbing the original data. Despite its feasibility and widespread use in low dimensional data, the number of Fourier coefficients, 2^d, grows exponentially with increased dimensionality. This results in intractable computational cost when working with high dimensional data. Another method is designed by Ding et al. (Ding et al., 2011) to work with high dimensional data such as online analytical processing (OLAP). In this framework, strategic cuboids that are useful to generate other cuboids are chosen first, and a private version of these cuboids is constructed by using differential privacy. The main limitation of this study arises while constructing the strategic cuboids. As all possible cuboids are iteratively traversed and selected, the number of the cuboids grows with the dimensions of the data, resulting in an increased complexity. A more practical and efficient approach, known as PriView, addresses the high dimensionality problem (Qardaji et al., 2014). PriView also constructs the private k-way marginal tables for $k \geq 3$. While constructing private marginal tables, PriView first extracts low-dimensional marginal views from the flat data and adds noise to the views. Next, PriView applies a reprocessing technique to ensure the consistency of the noisy views. Afterwards, PriView applies maximum entropy optimization on the views to obtain the k-way marginal tables. PriView is reported as a more efficient technique in terms of time and space complexity; however, it can be employed on binary data only.

There are other frameworks, designed particularly for differential optimization problems. First, Dwork et al. (Dwork et al., 2006) propose an output perturbation technique that directly add noise to the regularized objective function after optimization. This technique is outperformed by the objective perturbation technique proposed by Chaudhuri et al. (Chaudhuri et al., 2011) which adds noise to the objective function before optimization. We denote this work as PrivateSVM and compare its results to those of DP-SYN in the experiments section.

Differential privacy has been implemented in a number of data analysis tasks, including regression models (Chaudhuri and Monteleoni, 2009; Zhang et al., 2012), classification models (Jagannathan et al., 2009; Rubinstein et al., 2009; Vaidya et al., 2013) and association rule mining (Li et al., 2012; Zeng et al., 2012).

Generating artificial data from the original one is another privacy preserving technique for data publication. Here, instead of using the sanitization models discussed previously, Rubin (Rubin, 1993) introduces repetitive perturbation of the original data as a substitute to the original data. To execute this technique, Zhang et al. (Zhang et al., 2014) present a synthetic data generation technique, PrivBayes. PrivBayes is defined as a differential generative model that decomposes high dimensional data into low dimensional marginals by constructing a Bayesian network. Afterwards, noise is injected into these learned low dimensional marginals to ensure differential privacy and the synthetic data is inferred from these noised marginals. Although PrivBayes is credited as an effective technique, as we will show in our experiments, our proposed technique has a significant improvement over PrivBayes.

Acs et al. (Ács et al., 2017) model generative neural networks to produce synthetic samples. The authors first cluster the original datasets into k clusters with private kernel k-means. Afterwards, they use generative neural networks for each cluster to create synthetic data. In our experiments, we denote this work with DP-VAE and compare its results to our method.

Bindschaedler et al. (Bindschaedler et al., 2017) present another differential generative framework. The authors introduce an idea of *plausible deniability*, rather than adding noise to the generative model directly. Plausible deniability is ensured by a *privacy threshold* in releasing synthetic data. Here, an adversary cannot tell whether a particular input belongs to the original data by observing synthetic records.

Park et al. (Park et al., 2016a) propose a private version of the iterative expectation maximization algorithm. They effectively combine differential privacy and expectation maximization algorithm to cluster datasets. Here, we use this approach to discover patterns in latent space. We observed an improvement in the performance of this technique when used with partitioning the original dataset into unique data label groups. We use this modified version in our experiments (Park et al., 2016a) as DP-EM(SYN) and compare its results in the experiments section.

In some cases, combining differentially private algorithms has been proven to be useful in formulating more complex privacy solutions. However, such combinations may result in degradation of the privacy protection as more information is leaked by multiple usage of the private techniques. To track the total privacy loss while executing such mechanisms, Dwork et al. (Dwork et al., 2014) propose basic and advanced composition theorems. Abadi et al. (Abadi et al., 2016) propose another advanced composition theorem known as the *moments accountant* and verify that it has the best overall privacy bound in the literature. Abadi et al. also utilize moment accountant while constructing a deep learning technique to classify images.

Despite their success in data utility measures, most of the proposed methods in the literature are impractical to be implemented for high dimensional data. In Chapter 5, we compare existing techniques on different datasets using different utility metrics moreover, we present a novel approach that utilizes deep learning techniques coupled with an efficient analysis of privacy costs to generate differentially private synthetic datasets with higher data utility.

2.3 Decoy File Generation

Cyber deception mechanisms have been heavily studied to enhance the computer security. However, most of the existing techniques are not focused on generating deceptive data. Here, we review the existing cyber deception techniques with their limitations and strengths.

Honeypots are a prominent cyber deception mechanism to investigate and analyze the unauthorized intrusions (Spitzner, 2003). Honeypots are designed as trap based isolated systems that appear vulnerable to attackers. Legitimate users are not supposed to interact with them and any interaction with honeypots is considered an illicit attempt. While interacting with intruders, honeypots gather information of cyberattackers to disclose intruders' behavior for forensic analysis. Although honeypots are a notable cyber deception technique, they have limitations. Since honeypots are *fake environments*, they might fail to simulate the real services. As attackers become more sophisticated, they ensure their safety by using more advanced systems to distinguish "fake" and real system to avoid honeypots (Holz and Raynal, 2005). Moreover, honeypots might create irredeemable risks for the real user environment when the attacker can use honeypots as a bridge to the real user environment (Almeshekah and Spafford, 2016).

In addition to Honeypots, decoy injection mechanisms evolved to integrate real systems in aiding defensive computer deception. These mechanisms serve as a *decoy* to intruders to mitigate unauthorized threats by distracting attackers from a target that have sensitive information.

Yuill et al. (Yuill et al., 2004) presents an intrusion detection system that installs *decoy files* on file servers with enticing names to capture the attention of attackers. These decoy files are constantly monitored and when accessed by any intruder, the system will trigger an alarm to notify system administrator. However, in some cases, decoy files fail to influence the perception of attackers since published data (e.g., password file stolen from LinkedIn [1])

[1] `https://www.cnet.com/news/linkedin-confirms-passwords-were-compromised/`

provide attackers insight to distinguish between real and fabricated data. Attackers can enhance their technique and re-attack again. To circumvent attacker insight, Juels et al. (Juels and Rivest, 2013) proposes *Honeywords* to defend hashed password databases by generating "fake passwords" that seem real to attackers. In their work, they preserve N-1 "fake passwords" referred as honeywords for each legitimate user password in the database. If any of the honeywords is submitted for logging into databases, attack has been detected and system administrator is notified that database has been hacked. Although honeywords are useful to detect the unauthorized intruders, in some cases it may deteriorate system performance because each submitted password is compared with all previously generated honeywords which slows down the authentication process for legitimate users. Also, generating and preserving the honeywords increases the storage requirement N times. Still this approach is only applicable for password setting.

Our approach proposes decoy data generation to fool attackers without degrading system performance. Although, decoy files are used in a cyber defensive system to entice attackers, they may reveal sensitive information if care is not taken during data generation. To preserve individual privacy, the decoy files require sanitization of sensitive information. Dwork (Dwork, 2006) proposes a data privacy model as ε-*differential privacy* to ensure the protection of private data from leakage by perturbing the data with random noise based on ε. Differential privacy has been implemented in a number of data analysis tasks, including regression models (Chaudhuri and Monteleoni, 2009; Zhang et al., 2012), classification models (Rubinstein et al., 2009; Vaidya et al., 2013) and privacy preserving data publishing (Bindschaedler et al., 2017; Ács et al., 2017; Zhang et al., 2014). In some cases, it is required to combine differentially private algorithms to formulate complex privacy solutions. To track the total privacy loss while executing these repetitive mechanisms, Abadi et al. (Abadi et al., 2016) proposes the advanced composition theorem known as the *moment accountant* and verify that it has the best overall privacy bound in the literature. In this work, we also employ the moment accountant to bound privacy of the proposed technique to generate decoy files.

To balance both utility and user privacy, Rubin (Rubin, 1993) introduces repetitive perturbation of the original data as a substitute to the original data. However, data generation may suffer from curse of dimensionality when the data has more than dozen attributes. To overcome the curse of dimensionality, Zhang et al. (Zhang et al., 2014) presents PrivBayes as a private generative model that decomposes high dimensional data into low dimensional marginals by constructing a Bayesian network. Afterwards, noise is injected into previously constructed low dimensional marginals to ensure differential privacy and the synthetic data is inferred from these sanitized marginals. Acs et al. (Ács et al., 2017) models another generative approach to produce synthetic samples. First, the original data is partitioned into k clusters with private kernel k-means. Then, each previously clustered data is inputted to private generative neural networks to create synthetic data.

Park et al. (Park et al., 2016b) proposes DpEm as a private version of the iterative expectation maximization algorithm. They combine differential privacy and expectation maximization algorithm to cluster datasets. Here, we use this approach to discover patterns in latent space. We observed an improvement in the performance of this technique when used with partitioning the original dataset into unique data label groups. Here, we use this modified version in our experiments (Park et al., 2016b) as DpEm^{+} and compare its results in the experiments section.

Similar to the clustering approach, in Chapter 5, new generative deep learning method is proposed that produces synthetic data from a dataset while preserving the utility of the original dataset. The original data is partitioned into groups, and then the private auto-encoder (a type of deep learning model) is employed for each group. Auto-encoder learns the latent structure of each group, and uses expectation maximization algorithm to simulate them. In Chapter 6, the applicability of these techniques in the context of generating relational honey data is investigated.

CHAPTER 3

BACKGROUND FOR RELATED CONCEPTS

In this chapter, the concepts and techniques utilized in this dissertation are described.

3.1 Topological Data Analysis

In this subsection we provide a general overview of the associated mathematical apparatus to bring the persistent homology tools to the analysis of blockchain networks.

Let $\mathbb{X} = \{X_1, \ldots, X_n\}$ be a set of data points in a metric space (e.g., the Euclidean space or a manifold). Select a threshold ϵ_k and form a graph G_k with the associated adjacency matrix $A = \mathbb{1}_{d_{ij} \leq \epsilon_k}$, where d_{ij} is the distance between points X_i and X_j. Changing the threshold values $\epsilon_1 < \epsilon_2 < \ldots < \epsilon_N$ results in a hierarchical nested sequence of graphs $G_1 \subseteq G_2 \subseteq \ldots \subseteq G_N$ that is called a *graph filtration*. That is, we glean the intrinsic geometry of $\{X_i\}_{i=1}^n$ from a multi-lens perspective, associated with a graph filtration. The main idea is to assess which geometric features remain persistent over the set of thresholds and, hence, are likely to play a significant role in functionality of the blockchain network.

Since it is generally hard to extract meaningful topological and geometric information from a discrete set of points, we associate an abstract simplicial complex with each G_k, $k = 1, \ldots, N$, which, in turn, allows to approximate the geometry underlying $\{X_i\}_{i=1}^n$ with a combinatorial structure. Furthermore, by quantifying all topological invariants associated with a simplicial complex, we bypass subjective selection of features, or feature engineering. For instance, the Vietoris-Rips (VR) combinatorial complex is one of the the most popular choices in TDA due to its simplicity and computational advantages (Carlsson, 2009; Zomorodian, 2010).

Definition 3.1.1 (Vietoris-Rips complex)**.** *A Vietoris-Rips complex at threshold ϵ, denoted by VR_ϵ, is the abstract simplicial complex consisting of all k-element subsets of*

$\mathbb{X} = \{X_1, \ldots, X_n\}$, *called* $(k-1)$*-simplices,* $k = 1, \ldots, K$*, whose points are pairwise within distance of* ϵ*. If* $\mathbb{X} \subseteq \mathbb{R}^d$*, a 0-simplex can be identified with a point, a 1-simplex with a segment, a 2-simplex is a triangle and a 3-simplex is a tetrahedron.*

Armed with the associated *VR filtration*, $VR_1 \subseteq VR_2 \subseteq \ldots \subseteq VR_N$, we can track qualitative topological features such as connected components, 1-dimensional holes, 2-dimensional holes and their higher-order analogs, that appear and disappear with an increasing threshold ϵ.

Persistent Homology. Systematic evaluation of patterns and dynamics of multiscale network geometry can be approached via *persistent homology*. Homology studies shapes in data, whereas the word persistent implies a focus on shapes that persist (do not disappear) in some notion of continuum. We simulate this continuum by increasing a threshold ϵ_k by very small steps in the graph filtration. As the threshold increases, more edges are added to the graph and certain shapes (e.g., loops) start to appear. We use Betti numbers to count these shapes for the graph at a given threshold.

Betti numbers represent counts of p-dimensional holes where $p = 0, 1 \ldots, \infty$. Lower Betti numbers have a simple interpretation on networks. For instance, β_0 is the number of connected components; β_1 is the number of 1-dimensional holes (loops), etc. We formally define a Betti number as follows:

Definition 3.1.2 (Betti numbers)**.** *The p-th Betti number* β_p*,* $p \in Z^+$*, of a simplicial complex is the rank of the associated p-th homology group defined as the quotient group of the cycle and boundary groups. The p-th Betti number of the VR complex at threshold* ϵ *is denoted by* $\beta_p(\epsilon)$*.*

A convenient way to visualize persistent-homology-based summaries is by a *barcode* plot which is closely related to Betti numbers. A barcode is a set of stacked horizontal intervals

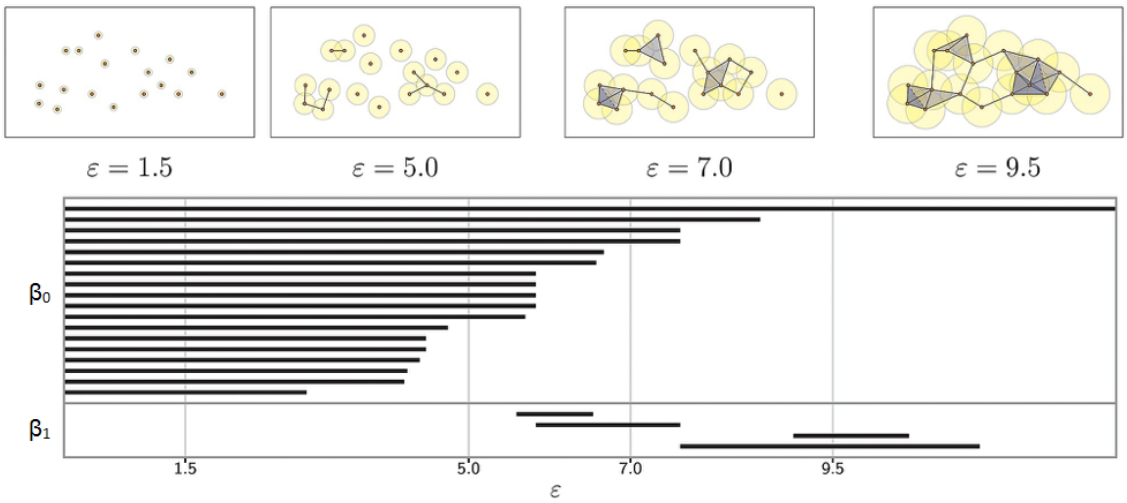


Figure 3.1: Barcode of a Vietoris-Rips complex built over 18 points in the plane. The top four figures are the snapshots of the evolving complex as threshold ϵ increases. The ϵ values corresponding to the two ends of horizontal bars mark the birth and death of topological features. To find Betti numbers, we count the number of times respective horizontal bars intersect the vertical line through ϵ. For example, for $\epsilon = 7$, $\beta_0 = 4$ and $\beta_1 = 1$ (Topaz et al., 2015).

(or bars), called *persistent intervals*, representing the birth and death of topological features of various dimensions (see Figure 3.1).

Betti Numbers for a Blockchain Network

The complexity of learning on large networks is a known issue (Shervashidze et al., 2009). Although Betti numbers provide a non-parametric solution to combine information on edge distance with node connectedness, the computational complexity of Betti calculations prohibits their usage in large networks. For example, for 2-simplicial complexes, "currently no upper bound better than a constant times n^3 is known" (Edelsbrunner and Parsa, 2014). For Betti numbers $\beta_{p>3}$, the complexity becomes too restrictive. This problem is compounded in the Bitcoin network because address reuse is discouraged. As such, every day brings more than 500K new nodes to the network. Betti number computations on such large networks is unfeasible.

To solve the complexity issues, we propose a novel approach that computes Betti numbers on a network of $N \times N$ nodes where N is the size of the amount matrix $\mathcal{A}$. Each of the N^2 unique chainlets (e.g., $\mathbb{C}_{2\to3}$) creates a node in the new network, where edge distance between two nodes is computed with a suitable 'distance' d. We describe the main steps as follows:

Given a heterogeneous Blockchain network with transferred bitcoins on edges,

1. All the transferred amounts are converted from Satoshis to bitcoins (dividing by 10^8), then added one (so that the values after taking logarithm are non-negative) and log-transformed: $a' = \log(1 + a/10^8)$, where a is an amount in Satoshis.

2. For each chainlet of a given time period, we compute the sample q-quantiles for the associated log-transformed amounts (Hyndman and Fan, 1996): a k-th q-quantile, $k = 0, 1, \ldots, q$, is the amount $Q(k)$ such that

$$\sum_{i=1}^{\tau} \mathbb{1}_{y_i<Q(k)} \approx \frac{\tau k}{q} \text{ and } \sum_{i=1}^{\tau} \mathbb{1}_{y_i>Q(k)} \approx \frac{\tau(q-k)}{q},$$

   where τ is the total number of transactions. The (dis)similarity metric d_{ij} between chainlet nodes i and j is defined as the quantile-based distance

$$d_{ij} = \sqrt{\sum_{k=0}^{q} [Q_i(k) - Q_j(k)]^2}.$$

3. We construct a sequence of thresholds $\epsilon_1 < \epsilon_2 < \ldots < \epsilon_S$ covering a range of distances during the entire 365-day period. For each threshold ϵ_k, we build the corresponding Vietoris-Rips complex whose 0-simplices are single chainlets and 1-simplices are pairs of chainlets with distance less than or equal to ϵ_k. Thus, we obtain the filtration of VR complexes $VR_1 \subseteq VR_2 \subseteq \ldots \subseteq VR_S$.

4. We compute $x_t = \{\beta_0(\epsilon_1), \ldots, \beta_0(\epsilon_S); \beta_1(\epsilon_1), \ldots, \beta_1(\epsilon_S)\}$ on VR filtrations.

In constructing the new network, we use and hence retain the amount information from the Blockchain network. Furthermore, each node type (chainlet) encodes the number of inputs and outputs in a transaction. This way, we combine distance (computed from transferred coins) with edge connectedness while restricting the network size. Our new TDA approach can work with networks of any size, and our experimental results show predictive power of its topological features.

Betti derivatives

When we consider the Betti numbers of dimension p against increasing filtration thresholds, we obtain a discrete curve which we refer to as the *p-th Betti curve*. Functional analysis of Betti curves allows us to assess dynamics of essential mesoscopic features as a function of thresholds. Furthermore, we introduce a novel concept of *Betti derivatives* up to order $\ell > 0$ on VR filtrations:

$$\partial\beta_p(\epsilon_k) = \beta_p(\epsilon_{k+1}) - \beta_p(\epsilon_k)$$

$$\partial^2\beta_p(\epsilon_k) = \partial\beta_p(\epsilon_{k+1}) - \partial\beta_p(\epsilon_k),$$

$$\dots$$

$$\partial^\ell\beta_p(\epsilon_k) = \partial^{\ell-1}\beta_p(\epsilon_{k+1}) - \partial^{\ell-1}\beta_p(\epsilon_k),$$

where $k = 1, 2, \dots, S-1$, $p = \{0, 1, \dots\}$ values are determined by how many Betti numbers we choose to use, and S is the number of filtration steps. These finite difference are analogues of derivatives for smooth functions. The inclusion of the rates of change of the Betti curves is intended to systematically capture dynamics of essential topological features and to enhance the predictive power. In (Hofer et al., 2017) the topological features of dimension zero are split into the essential (persisting till the end of filtration) and non-essential. However, there could be features that persist over a significant range of threshold values but disappear right before the filtration ends and thus fall under the category of non-essentials. In contrast, our

approach considers the Betti curves along with their shape rate derivatives as a whole and thereby allows to view such features under a more general umbrella of the essential features.

3.2 Differential Privacy

In this subsection we provide a general overview of Differential privacy. Differential privacy is the formal mathematical model that ensures privacy protection, and it is primarily used to analyze and release sensitive data statistics (Dwork et al., 2006). Differential privacy utilizes randomized algorithms to sanitize sensitive information while bounding the privacy risk of revealing sensitive information.

Theorem 1 (**(ε, δ)**-Differential Privacy (Dwork et al., 2006))**.** *For two non-negative numbers ε, δ, a randomized algorithm, $\mathcal{F}$, satisfies (ε, δ) -differential privacy iff for any neighboring pair d, d' and $S \subseteq$ Range($\mathcal{F}$), the following formula holds:*

$$Pr[\mathcal{F}(d) \in S] \leq \exp \varepsilon Pr[\mathcal{F}(d') \in S] + \delta. \tag{3.1}$$

Here, the neighboring pair differ from each other with only one entry while the remaining entries are identical. In Theorem 3.1 (Dwork et al., 2006), δ is a relaxation to *ε-differential privacy* that formulates the probability of privacy leakage. However, to avoid such leakage, Dwork et al. (Dwork et al., 2014) shows that δ must be chosen smaller than $1/n$ for a data of n samples.

Our proposed technique sanitizes sensitive data based on a widely used differentially private technique, the Gaussian mechanism (Blum et al., 2005). The deterministic function f takes d as input. $f(d)$ perturbs the input with noise sampled from the normal distribution $\mathcal{N}$, based on ϵ, δ, and s_f which is the sensitivity of f defined as follows:

Definition 3.2.1 (Sensitivity (Dwork et al., 2006)). *For a given function f, the sensitivity of f is defined as a maximum absolute distance between two neighboring pairs (d,d')*

$$s_f = \max_{(d,d')} \|f(d) - f(d')\|, \tag{3.2}$$

where $\|.\|$ is L_1 norm.

The (ε, δ) -*differential privacy* of function f over data d is guaranteed by $\mathcal{F}(d)$ with the Gaussian mechanism:

$$\mathcal{F}(d) = f(d) + z, \tag{3.3}$$

where z is a random variable from distribution $\mathcal{N}(0, \sigma^2 {s_f}^2)$. Here, when $\varepsilon \in [0, 1]$, the relation among the parameters of Gaussian mechanism (Dwork et al., 2014) is such that

$$\sigma^2 \varepsilon^2 \geqslant 2 \ln 1.25/\delta s_f^2.$$

3.3 Deep Learning

Deep learning is a subfield of machine learning that can be either supervised or unsupervised (Hinton et al., 2012). The power of deep learning comes from discovering essential concepts of data as nested hierarchy concepts where simpler concepts are refined to obtain complex concepts. We focus on the auto-encoder, an unsupervised deep learning technique that outputs a reconstruction of its input.

An auto-encoder is trained by optimizing an objective function. Stochastic gradient descent (SGD) (Song et al., 2013) is used to solve this optimization problem. Rather than iterating over every training instance, SGD iterates over a *mini-batch* of the instances. For a given training set of m samples, $D=\{x_i\}_{i=1}^m$ and $x_i \in \mathbb{R}^d$, the objective function is given as:

$$\min_w \mathcal{L}(w) = \frac{1}{|B|} \sum_{x_i \in B} \ell(w; x_i), \tag{3.4}$$

where B is the *mini-batch*, w is auto-encoder model parameter and ℓ is the discrepancy cost of example x_i and its reconstruction $\tilde{x_i}$.

At each step t, model gradient is computed for a given batch B_t and learning parameter η. Then, the model parameter is updated for the next step as follows:

$$w_{t+1} = w_t - \eta \left(\frac{1}{|B_t|} \sum_{x_i \in B_t} \nabla_w \, \ell(w; x_i) \right). \tag{3.5}$$

Figure 3.2 presents two main phases of an auto-encoder: the **encoder** and the **decoder**.

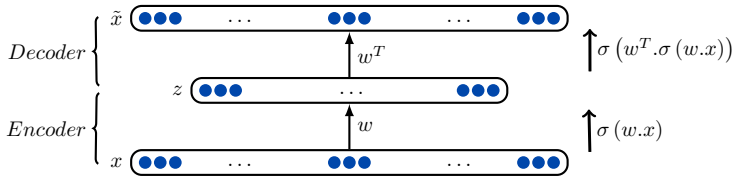


Figure 3.2: One hidden layer auto-encoder that encodes the input to the latent space, and decode the latent space to the reconstruction of input.

The **encoder** maps its input to a hidden intermediate layer that usually has less neurons than the input size to get a latent representation of the input. Here, the element-wise activation function σ maps $x \in \mathbb{R}^d$ into $z \in \mathbb{R}^{d'}$ where $d' < d$. On the other hand, the **decoder** takes the latent representation z, and reconstructs $\tilde{x} \in \mathbb{R}^d$.

CHAPTER 4

CHAINNET: LEARNING ON BLOCKCHAIN GRAPHS WITH TOPOLOGICAL FEATURES [1]

4.1 Introduction

In financial markets, machine learning applications has become more prominent to draw insights and make predictions in calibrating trading decisions. For those applications, accurate and robust prediction is paramount to decrease risk in companies' investments. To achieve it, many different predictive models are proposed to financial systems, however, existing approaches are failed to identify the stock market trends and patterns due to high volatility of the financial systems. To better predict the market changes, we propose a cryptocurrency predictive model with topological features. To evaluate the contribution of our approach, we use Bitcoin data as a use case in this dissertation. Here, we use Bitcoin as a use case since future of Bitcoin and cryptocurrencies and its potential impact in financial markets have created discussions (Mattila et al., 2016). One interesting aspect of popular cryptocurrencies such as Bitcoin is that each transaction is recorded on a distributed public ledger called blockchain. The transactions recorded on the blockchain can be accessed and analyzed by anyone. Furthermore, all of the transactions could be represented by a graph which we refer to as the "blockchain graph". Existence of the blockchain graph raises important questions such as "How does the blockchain graph structure impact the underlying cryptocurrency price?"

In this chapter, we focus on answering this question by proposing different approaches to represent blockchain graph patterns; and we use these patterns to build machine learning models for price prediction.

[1] ©2019 IEEE. Reprinted, with permission, from Nazmiye Ceren Abay, Cuneyt G. Akcora, Yulia R. Gel, Umar D. Islambekov, Murat Kantarcioglu, Yahui Tian, Bhavani Thuraisingham, "ChainNet: Learning on Blockchain Graphs with Topological Features", To Appear in the 2019 IEEE International Conference on Data Mining (ICDM), Nov 2019.

To draw conclusive insights into the blockchain graph dynamics and its impact on cryptocurrency price, we propose novel techniques inspired by topological data analysis and, particularly, persistent homology that can capture these higher order interactions.

Persistent homology, or analysis of properties of progressively finer simplicial complexes, unveils some critical characteristics behind functionality of a blockchain graph and interactions of its components at multi-scale levels, which are otherwise largely inaccessible with conventional analytical methods. Such an approach provides a number of important benefits. First, we systematically account for mesoscopic changes in the blockchain graph geometry, both in terms of transaction patterns and associated transaction volumes. Second, analysis of the combinatorial structure of the abstract simplicial complexes associated with a blockchain graph allows bypassing a stage of feature engineering – we no longer need to subjectively select topological features, such as degree distribution, but instead use an exhaustive knowledge on topological invariants of the blockchain graph and evaluate its predictive utility for cryptocurrency price dynamics. Furthermore, the limited studies on application of topological data analysis to other types of networks show that persistent homology features outperform conventional graph features such as betweenness centrality, clustering coefficient and nodal degree in network classification and segmentation (Garg et al., 2016). In this dissertation, we introduce the power of topological data analysis, particularly, persistent homology and associated network filtrations, to the analysis of blockchain graphs.

The contributions in this approach can be summarized as follows:

- To our knowledge, we are the first ones to introduce the persistent homology tools to cryptocurrency predictive analytics. Furthermore, we couple mesoscopic topological features of Blockchain with Machine Learning techniques to predict Bitcoin prices.

- Using extensive empirical analysis, we show that our proposed persistent homology based machine learning models can significantly outperform (i.e., up to 38% improvement in

root mean squared error) models that use only past price and standard features such as total transaction count.

4.2 Problem Definition

Problem Statement: Let $x_t \in \mathbb{R}^d$ be a set of features computed on the Bitcoin blockchain. Let $(x_1, y_1), \ldots, (x_t, y_t)$ be the observed data where $Y = \{y_1, \ldots, y_t\}$ are the corresponding Bitcoin prices in dollars. At a time point t, estimate the Bitcoin price $y_{t'}$ where $t' > t$.

Before solving this problem, we need to address these additional questions:

- How can real world Bitcoin prices be determined by blockchain network activity? Can the causality be proven?

  ♦ Our hypothesis is that input and output based structure of Bitcoin transactions encode various buyer and seller motivations that reflect market sentiment, which in turn determines price movements. For example, investments in the currency are encoded in transactions that contain more inputs than outputs. Similarly, selling behaviour creates transactions with more outputs than input addresses. Already, our previous results (Akcora et al., 2018) offer evidence for a causality between blockchain activity and Bitcoin price. In this chapter we offer further evidence for the causality.

- Most Bitcoin transactions on online exchanges are handled in-house by exchanging private/public keys pairs between users. How can we account for these missing transactions?

  ♦ We are aware that in-house transactions can be as many as 3 to 30 times (See Figure 4 in (Antulov-Fantulin et al., 2018)) the number of transactions published in the blockchain. However, we claim that in-house transactions are still periodically published to the blockchain in batches. Transaction histories of exchange addresses, such as the

Coinbase Bitcoin address, [2] contain evidence to support our claim. Otherwise a data loss would bring about huge losses, as happened to the Mt. Gox exchange in 2014. Although they contain a lagged version of data, exchange transactions still contain useful information, and their amounts can be utilized in a predictive model.

- From a methodological perspective, why is the price prediction problem important?

  ♦ Price prediction is important as it impacts a billion dollar industry in cryptocurrencies. However, we would like to point out another aspect. We argue that price, which is arbitrated off-chain in real world, is a unique external validator for testing the power of machine learning models on a complex system that is created worldwide by real actors. For example, we use the price to validate the predictive power of topological data analysis tools and constructs, e.g., Betti numbers. As price is inherently related to real life phenomena, such as network growth and influential user behaviour, we envision that many network growth, scaling and influence models (Gionis et al., 2012) can be validated by using settings similar to ours.

We provide two solutions to our research problem: *graph filtration (FL)* and *Betti signatures*. The first approach is based on graph filtration. That is, we filter the transaction network with increasing thresholds of Bitcoin amounts, and create multiple realizations of the network. Afterwards, we merge these realizations to train a model. The second approach uses topological signatures to capture persistent features in terms of Betti numbers and Betti derivatives.

The Betti approach is based on rigorous mathematical foundations of algebraic topology and provides a mesoscopic view of the system, whereas the graph filtration is a heuristic that allows manually selecting amount thresholds and associated filtering of the network. Next, we describe these two approaches in details.

[2] `https://www.blockchain.com/btc/address/1LQTXi1iWULMd4aKn5tKpcgT3xgJiTV5Dm`

4.3 Learning Graph Based and Topological Features

4.3.1 Learning Graph Representations

We first introduce existing blockchain network models and explain their shortcomings. Next we give our chainlet model, and extract *graph filtration* features using the chainlets.

In a typical blockchain graph such as the one used by Bitcoin, an owner of multiple addresses (i.e., each address represents an account, each person may have many addresses/accounts) can combine them in a transaction and send coins to multiple output addresses. Therefore, the Bitcoin blockchain consists of two types of nodes: transactions, and addresses that are input/output of transactions. Earlier results on Blockchain analysis are based on constructing graphs with a single type of node: *transactions* (Ron and Shamir, 2013) or *addresses* (Filtz et al., 2017) constituted nodes and currency transfers created edges between nodes. By choosing a single type of node, these approaches omit either address or transaction information in the graph. In our approach we construct a heterogeneous Blockchain graph with both address and transaction nodes. Figure 4.1 shows a blockchain graph with transactions as rectangle and addresses as circle shaped nodes, respectively. Each directed edge connects an address to a transaction, and the edge direction denotes a transfer of currency. Blockchain edges are naturally ordered in time with respect to the block they appear in.

Once the graph is constructed, shapes of transactions, and how they connect addresses conveys information on how the graph further extends in time. For all purposes, a Blockchain graph can be thought as a forever forward branching forest where transaction nodes appear only once, and address nodes may appear multiple times (but in practice address reuse is discouraged on Bitcoin and other blockchains).

With its input and output addresses, each transaction represents an immutable decision that is encoded as a subgraph on the blockchain graph. Recently graph **chainlets** were proposed to encode and aggregate this information (Akcora et al., 2018). On the directed,

heterogeneous blockchain graph $\mathcal{G} = (V, E, B)$, V is a set of vertices, and $E \subseteq V \times V$ is a set of directed edges. The set $B =\{$ **Address**, **Transaction**$\}$ represents node types. On the blockchain graph $\mathcal{G}$, k-chainlets are defined as follows:

Definition 4.3.1 (The k-Chainlet)**.** *A blockchain subgraph $\mathcal{G}' = (V', E', B)$ is a subgraph of $\mathcal{G}$ (i.e., $\mathcal{G}' \subseteq \mathcal{G}$), if $V' \subseteq V$ and $E' \subseteq E$. Let $\mathcal{G}_k = (V_k, G_k, B)$ be a subgraph of $\mathcal{G}$ with k nodes of type $\{$**Transaction**$\}$. The $\mathcal{G}_k$ is called a graph k-chainlet.*

As the k value increases, k-chainlets encode higher order structures on the graph and the number of distinct shaped chainlets also increases. As each transaction can have thousands of inputs and outputs, even for the most basic case of $k = 1$, k-chainlets can have millions of distinct shapes.

For simplicity, $k = 1$-chainlets are referred to as *chainlets*. We encode chainlets with two dimensions: for $|i|$ *input* addresses and $|o|$ *output* addresses, the chainlet is denoted as $\mathbb{C}_{i\to o}$. The example Figure 4.1 shows three distinct chainlets: 2 of shape $\mathbb{C}_{2\to 2}$ (around t_1 and t_4), 1 of shape $\mathbb{C}_{3\to 1}$ (around t_2) and 1 of shape $\mathbb{C}_{1\to 3}$ (around t_3).

In motif analysis of networks (Milo et al., 2002), each motif (such as a triangle among three nodes) is counted on the network, and network dynamics are linked to motif densities. Chainlet analysis provides a similar role on Blockchain graphs; by counting the occurrence of certain shapes, a graph can be summarized with chainlet densities.

Occurrence and Amount Matrices: For a graph chainlet if there exists a $G_k \in G$, we say that there exists an **occurrence**, or *embedding* of $\mathcal{G}_k$ in $\mathcal{G}$. Another aspect of a chainlet is the amount of currency that is transferred from its inputs to outputs.

With occurrence and amount information of chainlets, the blockchain graph is represented with $[i_{max} \times o_{max}]$ dimensional occurrence and amount matrices, where the cell of i-th row and o-th column represents information on the chainlet $\mathbb{C}_{i\to o}$.

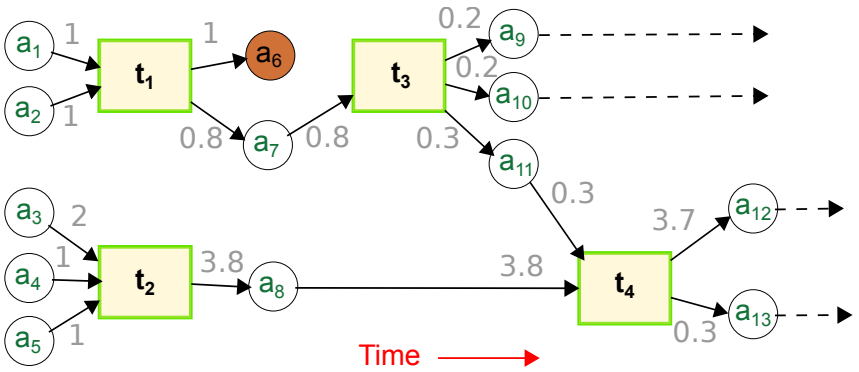


Figure 4.1: A Bitcoin graph with 4 transactions and 13 addresses. Amounts on edges show currency transfers. The difference between input and outputs amounts, if exists, shows the transaction fee collected by miners.

Example 1. *Consider the toy example in Figure 4.1, where both* $i_{max} = 3$ *and* $o_{max} = 3$. *Resulting* 3×3 *occurrence and amount matrices are given below as* $\mathcal{O}$ *and* $\mathcal{A}$, *respectively. In total, there are four chainlets but only three distinct shapes.* $\mathbb{C}_{1\to3}$ *and* $\mathbb{C}_{3\to1}$ *occurs once* ($\mathcal{O}_{13} = \mathcal{O}_{31} = 1$), *and* $\mathbb{C}_{2\to2}$ *occurs twice* ($\mathcal{O}_{22} = 2$). *The total amounts transferred by each chainlet are given as* $\mathcal{A}_{13} = 0.8$, $\mathcal{A}_{22} = 4.1 + 2$ *and* $\mathcal{A}_{31} = 3.8$.

$$\mathcal{O} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \textit{and } \mathcal{A} = \begin{bmatrix} 0 & 0 & 0.8 \\ 0 & 6.1 & 0 \\ 4 & 0 & 0 \end{bmatrix}$$

Graph Filtration (FL). Given the amount and occurrence information, a natural combination of them entails filtering the occurrence matrix with user defined thresholds on amounts, or filtering the amount matrix with user defined thresholds on occurrences. In both cases, the user defined threshold implies a heuristic aspect.

FL creates multiple occurrence matrices of a Bitcoin network at a given time period, and uses them as the feature set to train a prediction model. Algorithm 1 represents the main steps. At a given time period t, chainlets of the time period are iterated over with a set of thresholds. A chainlet $\mathbb{C}_{i\to j}$'s occurrence is recorded in the associated occurrence matrix $\mathcal{O}^{\epsilon}$

Algorithm 1 FL: Graph Filtration

Input: $\mathcal{G}$: Blockchain graph, time t, $\epsilon_{1,..S}$: set of S filtration thresholds.

for $\epsilon \in \epsilon_{1,..S}$ **do**
    $\mathcal{O}^{\epsilon} \leftarrow []$ //initialize occurrence matrix
end for
for chainlet $\mathbb{C}_{i \to j} \in \mathcal{G}_t$ **do**
    for each threshold $\epsilon \in \epsilon_{1,...,S}$ **do**
        if $\epsilon \leq amount(\mathbb{C}_{i \to j})$ **then**
            $\mathcal{O}^{\epsilon}_{ij} \leftarrow 1 + \mathcal{O}^{\epsilon}_{ij}$
        end if
    end for
end for
return $x_t = [\mathcal{O}^{\epsilon_1} \dots \mathcal{O}^{\epsilon_S}]$// concatenated occ. matrices

if the amount transferred by the chainlet $amount(\mathbb{C}_{i \to j}) \geq \epsilon$. The process is repeated for all inputted data. Resulting occurrence matrices are row-wise concatenated and output as the FL feature set for time period t (i.e., x_t).

The FL captures persistent graph structures by retaining edges among nodes according to a set of threshold values. For a threshold value $\epsilon \in \epsilon_{1,...,S}$, we only record the occurrence of chainlet, if the amount transferred by the chainlet is $\geq \epsilon$.

4.3.2 Learning Topological Representations

In this section, we introduce topological data analysis and define two types of information on network topological signatures. The first type, Betti numbers (e.g.,β_0, β_1), are computed on the network after using an increasing distance threshold and filtering out edges with distances above the threshold. The second information, Betti derivatives, encodes the rate of change in the computed Betti numbers for increasing thresholds.

Topological data analysis (TDA) and, persistent homology, in particular, is an emerging methodology at the intersection of algebraic topology, statistics and machine learning that allows to systematically infer qualitative and quantitative mesoscopic geometric structures directly from the data and to enhance our understanding on the hidden role of geometry in

functionality of a complex system (Carlsson, 2009; Chazal and Michel, 2017). The limited studies on application of TDA to random graphs indicate that analysis of topological invariants, (e.g., *Betti numbers*), outperform methods based on conventional graph features (Garg et al., 2016). Our goal is to bring the persistent homology tools to the analysis of blockchain networks; and the details of TDA, Betti Numbers and Betti Derivatives are provided at 3.1

4.4 Experiments

In this section, we show the performance of predictive models in our ChainNet framework. A detailed technical report on ChainNet is available at (Abay et al., 2019).

We begin by describing our dataset. Section 4.4.2 explains how we train the models on the time series of the learned features. Next, we introduce our machine learning models and define multiple baseline approaches for performance comparison purposes.

4.4.1 Data

We downloaded and parsed the entire Bitcoin transaction graph from 2009 January to 2018 December. Using a time interval of 24 hours, we extracted daily transactions on the network and created the Bitcoin graph. Our datasets are available online. [3] Our Bitcoin price (USD) data is downloaded from `blockchain.com` which aggregates prices from worldwide online exchanges. [4]

Filtration data. We analyzed Bitcoin transactions to find an appropriate dimension N for the occurrence matrix. On the Bitcoin graph % 90.50 of the chainlets have N of 5 (i.e., $\mathbb{C}_{i\to o}$ s.t., $i < 5$ and $o < 5$) in average for daily snapshots. This value reaches % 97.57 for N

[3]Please see: `https://github.com/Cakcora/coinworks`

[4]Due to the extreme divergence in prices from the rest of the world, Korean exchanges are excluded in Bitcoin price arbitration.

of 20. We chose $N = 20$, because it can distinguish a sufficiently large number (i.e., 400) of chainlets, and still offer a dense matrix. Our models achieved a satisfactory performance with $\epsilon \in \{0, 10, 20, 30, 40, 50\}$ thresholds in the graph filtration. However we note that ϵ partitions can be further improved.

Betti and Betti Derivative Data. We use the Betti numbers estimation routine of the Perseus (Nanda, 2017) software which provides an efficient algorithm to compute Betti numbers β_0 and β_1 and persistent intervals using discrete Morse theory.

We used $\epsilon \in \{50, 100, 200 \text{ and } 400\}$ as ϵ thresholds in calculating Betti features. Overall, we find no improvement in prediction accuracy for $\epsilon > 400$. Furthermore, there is no single optimal ϵ value to be used in all statistical and machine learning models. In the technical report, for each model's best performance we detail the used ϵ values.

The dynamics of Betti values is depicted in Figures 4.2 and 4.3 for 25 thresholds. Figure 4.2 reveals visible variation in β_0 and β_1 numbers across 365 days. In the present study, we focus on VR complexes of dimension one. This implies that 1-dimensional holes are formed by three or more nodes, which in turn leads to a general negative association between the β_0 and β_1 curves – as ϵ increases, more simplices are added to the complex, thereby reducing the number of connected components and increasing the number of 1-dimensional holes. For the same reason, we see in Figure 4.3 that the spikes in average β_0 curves match the plummets of the corresponding β_1 curves and vice versa. On July 20, 2017 the Bitcoin Improvement Proposal 91, to trigger Segregated Witness (SegWit) activation, is locked in. This has resulted in the start of the new bullish wave. Remarkably, we find that the spike in Bitcoin in mid July 2017 have been preceded by an increase in β_0, and decreases in β_1 and average daily transactions. Moreover, the extrema of β_0, β_1 curves and average daily transactions in July 2017 are well aligned.

In addition to FL and Betti related features, we also experimented with basic features: price, mean degree of addresses (MeanDegree), number of new addresses (NumNewAddress),

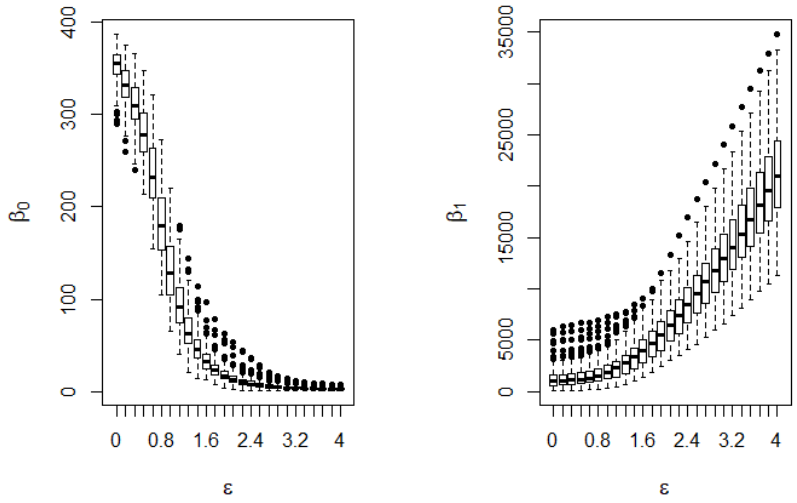


Figure 4.2: Boxplots of β_0 and β_1 numbers for various threshold ϵ values.

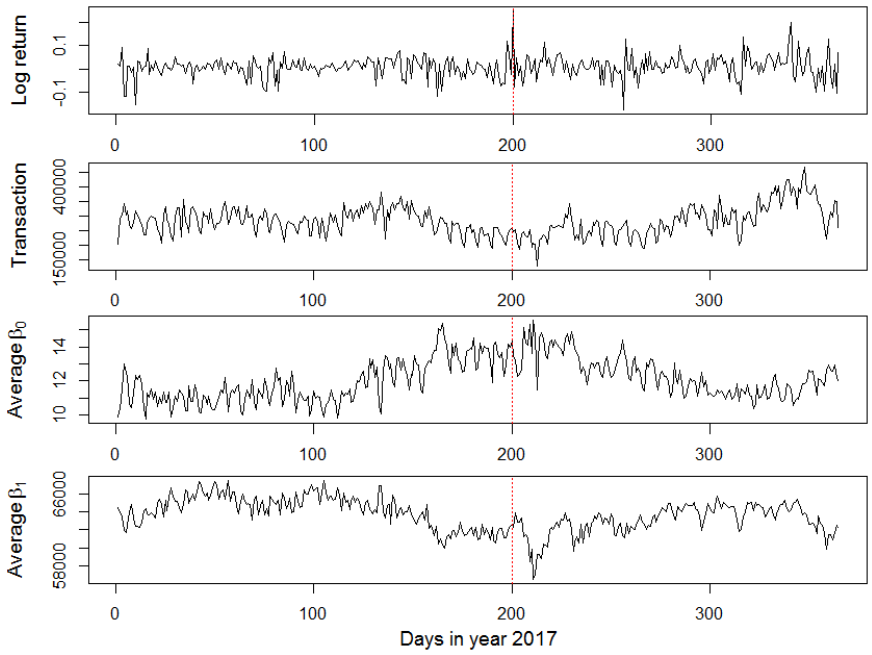


Figure 4.3: Time series of daily log returns, transactions, average β_0 and β_1 numbers in 2017.

mean and total coin amount transferred in transactions (meanTxAmount and TotalTxAmount, respectively) and address network average clustering coefficient (ClusCoeff). Among these, we only found Price and TotalTx to be useful predictors and included them in our models. Table 4.1 shows all the considered features.

Table 4.1: Features used in Machine Learning models for a given day.

Approach	**Feature Set**
Basic features	$Price, TotalTx, MeanDegree, NumNewAddress$ $MeanTxAmount, TotalTxAmount, ClusCoeff$
Filtration	$Price, TotalTx, \mathcal{O}^{\epsilon_1} \dots \mathcal{O}^{\epsilon_S}$
Betti (Sec. 3.1)	$Price, TotalTx, \beta_0(\epsilon_1), \dots, \beta_0(\epsilon_S), \beta_1(\epsilon_1), \dots, \beta_1(\epsilon_S)$
Betti derivative (Sec. 3.1)	$Price, TotalTx, \beta_0(\epsilon_1), \dots, \beta_0(\epsilon_S), \beta_1(\epsilon_1), \dots, \beta_1(\epsilon_S),$ $\beta_0'(\epsilon_1), \dots, \beta_0'(\epsilon_S), \beta_1'(\epsilon_1), \dots, \beta_1'(\epsilon_S)$

4.4.2 Setting for Feature Time Series

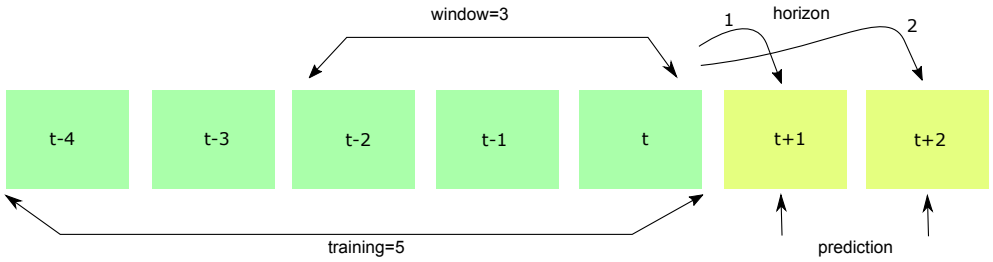


Figure 4.4: The sliding window based regressor model. The example model trains with data from the last $m = 5$ days, and uses the data from t, $t-1$ and $t-2$ (window=3) to make a prediction for either day $t+1$ (horizon=1) or day $t+2$ (horizon=2).

Given the features, we employ a time based approach to predict the Bitcoin price, as shown in Figure 4.4. Our goal is to catch trends in the price data, based on the observation that price movements in the preceding days are a good indicator of future prices.

ChainNet employs three time related concepts: training length, window (lag) and horizon. Training length is the number of past time periods whose data we use to train our model.

Window is the number of past time periods whose data we use to predict Bitcoin price. Horizon is the number of days whose price we predict ahead.

In the most basic case of prediction horizon $h = 1$ and prediction window $w = 1$, the model learns to predict the price of day $\hat{y}_{t+1}$ by using the data x_t of day t. Similarly, for any window w, the model uses data from $\{x_{t-w}, \ldots, x_t\}$ to predict the price $\hat{y}_{t+h}$.

Details of the sliding prediction approach is given in Algorithm 2. Input is time indexed data points and output is the model parameters trained on the given input. For given window w and horizon h values, time series data is processed to utilize the history of the current day, t (Line 2-5 in Alg. 2). Each x_t is replaced by the successive values of time series between $t - w - h$ and $t - h$ (Line 3 in Alg. 2). Newly generated $\hat{x_t}$ is and its corresponding price, y_t, is appended to the train list (Line 4-5 in Alg. 2). After all days are iterated on, dimension reduction is applied to the generated $\hat{x}_{train}$ to obtain compensated data (Line 6 in Alg. 2). At the end, model is optimized with the previously obtained train data and the algorithm returns the obtained model parameters for out-of-sample predictions (Line 7-8 in Alg. 2).

Algorithm 2 SPred: Sliding prediction

Input: Data:$\{(x_t, y_t)$: $t \in T\}$ where $x_t \in \mathbb{R}^d$; y_t: the daily bitcoin price in dollars; l: training length; w: sliding window length; h: prediction horizon; d_2: pca dimension
Output: θ: Model Parameters.

1: $x_{train}, \hat{x}_{train}, y_{train} \leftarrow \{\}$
2: **for** each $t \in [h + w : l]$ **do**
3: $\quad \hat{x}_t \leftarrow [x_{t-w-h+1}, \ldots, x_{t-h}; y_{t-w-h+1}, \ldots, y_{t-h}]$ // row-wise
4: $\quad \hat{x}_{train} \leftarrow \hat{x}_{train} \cup \hat{x}_t$
5: $\quad y_{train} \leftarrow y_{train} \cup \hat{y}_t$
6: **end for**
7: $x_{train} \leftarrow PCA(d_2, \hat{x}_{train})$
8: $\theta =$ model.fit(x_{train}, y_{train})
9: **return** θ

We consider the following two parameters in all predictive models: window $w \in \{3, 5, 7\}$, horizon $h \in \{1, 2, 5, 7, 10, 15, 20, 25, 30\}$, training length $l \in \{25, 50, 100, 200\}$. As the interaction of horizon, window and training length parameters may exhibit nonlinear effects on the

prediction, we conduct a grid search by varying all parameters, and report the predicted price values for the best model. The technical report contains the used values of all parameters in models.

An important point in our sliding prediction approach is that, we train a model per each prediction. As a result, we train a model 365 times to predict Bitcoin prices in 2017. We chose this setting because gain results improved over a batch prediction model. As we model data with low dimensional features, the cost of this approach was negligible.

4.4.3 Statistical and Machine Learning Models Used

We evaluate ChainNet performance by using one statistical (ARIMAX) and four machine learning models:

ARIMAX refers to the Auto-Regressive Integrated Moving Average model (with exogeneous variable) that is a conventional benchmark model in time series analysis and forecasting that accounts for data non-stationarity (Box et al., 2015).

XGBT is the eXtreme Gradient Boosting which applies gradient boosting algorithms to decision trees (Chen and Guestrin, 2016).

RF stands for Random Forest which is a supervised ensemble of multiple simple decision trees to estimate the dependent variables of the data (Ho, 1995).

GP presents Gaussian Process based Regression technique which is designed to estimate the regressor parameters with the maximum likelihood principle (Williams and Rasmussen, 1996).

ENET refers to the elastic net model which is designed as a regularized linear regression model with the L1 and L2 penalties of the *lasso* and *ridge* methods (Zou and Hastie, 2005).

Deep Learning Models. Given the recent popularity of Deep Learning, we initially considered Recurrent Neural Networks and Long Term Short Memory models in ChainNet. However, our experiments did not yield satisfactory results. We hypothesize that these models require more training data to achieve convergence than we can possibly supply at this point.

Parameter Setting for Models. For the hyper-parameter tuning of ARIMAX, the orders for auto-regression and moving average terms are chosen from $\{0, 1, 2\}$. For the tree based approaches such as XGBT, RF, generated number of trees are chosen from {10, 50, 100, 200, 300, 400, 500, 1000}. For the learning rate of XGBT, we tried values from $\{0.01, 0.1, 1.0\}$. ENET regularization parameters for L1 and L2 and penalty constants are selected from {0.0001, 0.001, 0.01, 0.1, 1.0, 10.0} and {0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0}. In hyper-parameter tuning of GP, regression types, correlation types, and regularization parameters are chosen from {constant, linear, quadratic}, {absolute exponential, squared exponential, generalized exponential, cubic, linear}, {0.001, 0.01, 0.1, 1.0, 10.0} respectively.

High Dimensionality. Since we use a windowed (lagged) history of the data, dimensionality of the training data increases rapidly.

For example, consider the Betti model with $S = 50$ filtrations. In addition to Price and TotalTx, each day has 50 β_0 and 50 β_1 Betti values. For $w = 3$, the model uses $(3 \cdot (100 + 2) = 306)$ features, whereas there can be at most $(2018 - 2009) * 365$ training instances if we use the entire Bitcoin history. Decreasing the number of thresholds (e.g., $S = 5$) can reduce dimensionality, but this approach reduces the power of Betti models as well, due to decreased threshold granularity.

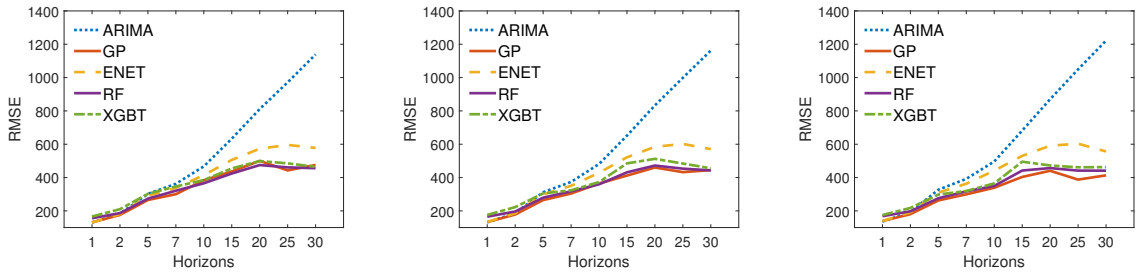
We apply Principal Component Analysis (PCA (Jolliffe, 2011)) to the lagged feature sets of FL, Betti and Betti derivative; in Algorithm 2 Line 7 PCA maps the high dimensional data into low dimensional data with the dimension of $d_2 \in \{5, 10, 15, 20\}$.

4.4.4 Baseline Performance

The simplest baseline for ChainNet can be constructed by training models on Price and TotalTx in a sliding window prediction scheme. We did not use other baseline features such as mean degree (see discussion in Section 4.4.1) since adding those features reduces the performance of the baseline models. We train baseline models without reducing the dimensionality (d_2=d in Alg. 2), because input features are very few; for $w = 3$, the models use 6 features in training. We assess model performance with root mean squared error (RMSE) as follows:

$$RMSE = \sqrt{\frac{1}{|T|}\sum_{t \in T}(y_t - \hat{y}_t)^2}, \tag{4.1}$$

where $|T|$ is the number of days, $\hat{y}_t$ is the predicted price and y_t is the true observed price on the t^{th} day.



(a) Window=3 (b) Window=5 (c) Window=7

Figure 4.5: RMSE of sliding window based predictions of 2017 Bitcoin prices in different window and horizon values.

In our rolling predictive framework, we achieve the best results with a training length of 100 days, that is, each considered model is adaptively re-estimated for each y_t using data from the previous 100 days. We only report the best results from each model with the hyper-parameter optimization.

Figure 4.5 shows the performance of the five models in prediction. ARIMAX has the worst performance for $h > 7$, whereas Gaussian Process (GP) has the best RMSE values

overall. We note that as the window value increases, performance does not improve. This implies that considering past information on price and total number of transactions does not deliver improvement in forecasting accuracy. In fact, from window 3 to 7, the RMSE values of the best model, GP, is approximately similar while $h < 10$. For $h > 10$, the RMSE values decrease 13% from window 3 to 7.

Other Baseline Research Studies. Our previous work on the utility of Chainlets constitutes a baseline for ChainNet; for horizon $h = 7$, ChainNet gains (38%) are higher than what we have achieved in (5%) (Akcora et al., 2018). The closest scholarly work to ChainNet is detailed in a report by Greaves et al. (Greaves and Au, 2015), where the authors extract both graph centric features (e.g., mean degree) and transaction features (e.g., mean amount) from the Bitcoin address graph, and use support vector machines to predict the Bitcoin price. As the authors also note at the end of their study, these features do not bring more information over a model that uses price data only. Indeed our experiments showed high error rates for predictions with the authors' experimental setting. More powerful models have been used in (Kondor et al., 2014; Shah and Zhang, 2014) with better results. We adopt similar machine learning models in this work, but in addition to the traditional features (see Table 4.1) ChainNet utilizes novel feature sets in FL, Betti and Betti derivative models.

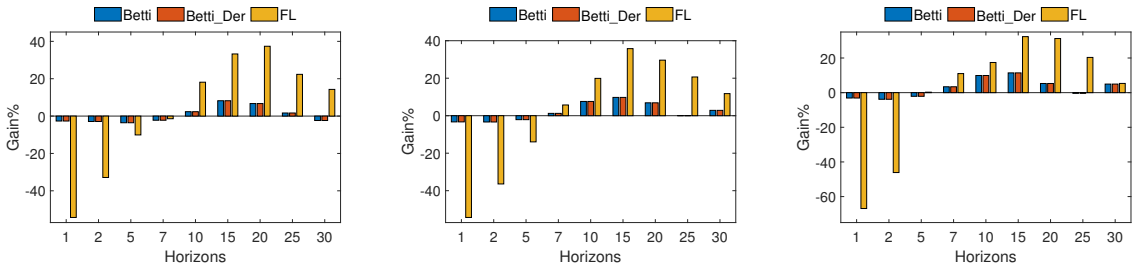
4.4.5 ChainNet Model Performance

In this section, we provide performance of the predictive models built with FL, Betti and Betti derivative features. *Our hypothesis is that adding these features will increase model performance*, i.e., RMSE in predictions will decrease over their associated baseline values.

Performance Gain. In our analysis, we report the percentage predictive gain, or decrease in $RMSE$ for a specific machine learning model m w.r.t. its baseline model m_0 as

$$\Delta_m(w,h) = 100 \times \left(1 - \frac{RMSE_m(w,h)}{RMSE_{m_0}(w,h)}\right)$$

where $RMSE_{m_0}(w,h)$ and $RMSE_m(w,h)$ are delivered by a baseline model m_0 and a competing model m, respectively.



(a) Window=3 (b) Window=5 (c) Window=7

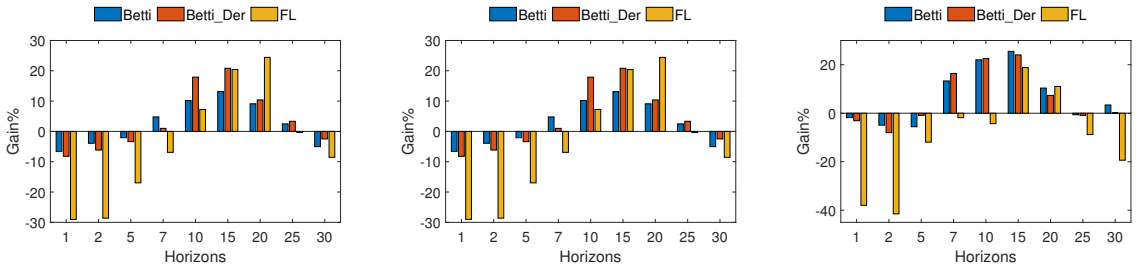
Figure 4.6: Elastic Net model performance.

Enet performance results are shown in Figure 4.6, which indicate that up to seven days, models do not improve when trained with ChainNet features. A similar trend is visible in the Random Forest(RF) results, as given in Figure 4.7. However, in RF results, for increasing horizons gain values dip below 0%, whereas Enet gains stay above 0%. In both models $h = 1, \ldots, 5$ predictions have negative gains. These results indicate that for immediate future, these machine learning models perform better *when trained on price and transaction counts (TotalTx) only.*

Intuitively, if Bitcoin price increases/decreases consistently in the last w days, we expect the trend to continue in the following days. RF and ENET models capture this trend better without the ChainNet features in short horizons.

Figures 4.8 and 4.9 show that XGBT and GP predictions improve for increasing horizons, but decrease for $h > 15$. Specifically $h = 1$ predictions reach a positive gain only in XGBT $w = 7$. XGBT also offers the best gains for $h = 2$, but its performance deteriorates for $h > 15$.

In constructing the XGBT model, the boosting approach focuses on examples that increase the error rate of objective function at each step. We hypothesize that this specific focus is the reason for XGBT's better performance.



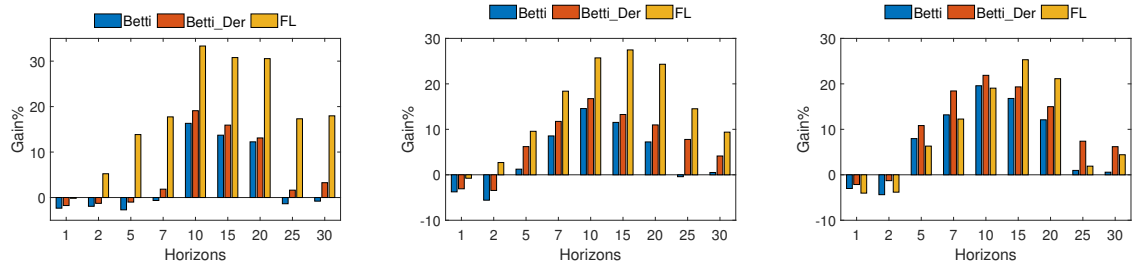
(a) Window=3 (b) Window=5 (c) Window=7

Figure 4.7: Random Forest Performance.

The highest gain values for $h \leq 7$ are achieved in XGBT Betti models for $w = 7$ (38% in Figure 4.9c). Our heuristic approach, FL, has an interesting trend; its usage in models lead to better gains for higher horizons. On the other hand Betti models achieve better gain values for short horizons. Considering these results, ChainNet can use Betti and Betti derivatives for short ($h < 10$) term prediction, and use FL for $h > 15$.

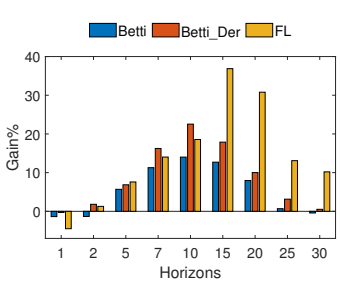
An important result is that next day predictions ($h = 1$) do not improve significantly (i.e., at most 2% in Figure 4.9c) with ChainNet features. In other words, topological and graph based signals in the blockchain have a negligible causal affect on the next immediate day.

Our results offer evidence for the hypothesis that considering topological features in predictive models bring a significant gain. ChainNet uses Betti models and FL for short and long term predictions, respectively.

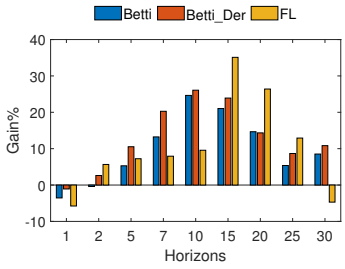


(a) Window=3 (b) Window=5 (c) Window=7

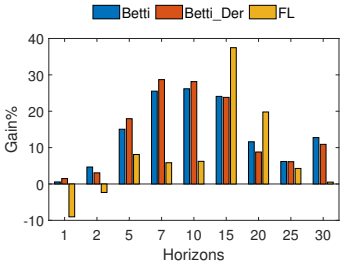
Figure 4.8: Gaussian Process (GP) based regression performance.



(a) Window=3



(b) Window=5



(c) Best gain results: Window=7

Figure 4.9: Extreme Gradient Boosting (XGBT) performance.

CHAPTER 5

PRIVACY PRESERVING SYNTHETIC DATA RELEASE USING DEEP LEARNING [1]

5.1 Introduction

In today's world, machine learning applications have been heavily used to almost every aspect of human life. These applications are continuously tracking our daily crucial activities thereby making our life easier and faster. To draw robust and accurate insight with these applications, machine learning applications collect and refine privacy sensitive micro-data, i.e., information at the level of individual respondents to build machine learning models. However, sharing micro data carries inherent risks to individual privacy. For example, a municipal dataset that contains information about bike sharing application has been used to identify individuals and their transit patterns (Vogel et al., 2011). Similarly, a taxi ride data set from New York have been used to identify certain individuals' addresses and their trips to certain night clubs (Wong, 2017). These examples show that there is an important societal need in sharing micro data with machine learning application while protecting individual privacy to offer important services and facilitate.

To address this privacy challenge, solutions have been proposed in two broad categories. In the first category, the data anonymization based approaches (e.g., (Sweeney, 2002)) try to use various definitions to sanitize data so that it cannot be easily re-identified. Although these approaches have some important use cases, they are not usually based on rigorous privacy definitions that can withstand various types of re-identification attacks. In the second category, synthetic data generation approaches have been proposed to generate

[1] ©2018 ECML. Reprinted, with permission, from Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, Latanya Sweneey, "Privacy Preserving Synthetic Data Release Using Deep Learning", 2018 The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD), Jun 2018.

realistic synthetic data using rigorous differential privacy definition (Dwork et al., 2014). Although these approaches have been shown to work in some limited cases, they have not been extensively tested on different types of use cases with different requirements (e.g., high dimensionality, correlation among features). Therefore, it was not clear which technique works well under what conditions for what type of data sets. We answer these questions by conducting extensive experimentation. Furthermore, we provide a new differentially private deep learning based synthetic data generation technique to address the limitations of the existing techniques.

In this chapter, we propose an auto-encoder technique (DP-SYN), a generative deep learning technique that generates privacy preserving synthetic data for machine learning applications. We test our approach on benchmark datasets and compare the results with other state-of-the-art techniques. We show that our proposed technique outperforms them in terms of three evaluation metrics.

Our contributions can be summarized as follows:

- We test existing techniques using different datasets with different properties using three utility metrics. We show that none of the existing techniques consistently outperforms others on all types of data sharing tasks and datasets.

- We propose a novel differentially private deep learning based synthetic data generation technique that is shown to be robust under different utility metrics with respect to different synthetic data generation tasks.

- We show that our approach does not deteriorate when faced with imbalanced or high dimensional datasets. Due to an inner partitioning of the latent structure, our approach gives more robust results in noise addition and works with both relational and image data.

5.2 Privacy Preserving Data Generation Model

This section describes the main components of our differentially private synthetic data generation approach. We first introduce our private auto-encoder and explain the private expectation maximization algorithm. Next, we present the privacy analysis of the proposed technique.

5.2.1 Differentially Private Synthetic Data Generation Algorithm

Our main framework aims to generate synthetic data without sacrificing the utility. A similar approach is proposed in (Abadi et al., 2016) which designs a private convolutional neural network on supervised learning. However, this method can only be used in classification tasks. We combine this method with DP-EM and to create a generative deep learning model.

Assume that we have the sensitive dataset $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where every instance $x \in \mathbb{R}^d$ has a label $y \in \{1, \ldots, k\}$. We partition the sensitive dataset D into k groups denoted as $\hat{D}_1 \ldots \hat{D}_k$ such that every instance x in a group $\hat{D}_i \in D$ has the same label y. The value of k is limited by the number of unique labels in dataset D.

Figure 5.1 shows the two main steps of our approach. For each data group we build a private generative auto-encoder which are denoted with DP-SYN. The lower pane of the figure shows the inner working of a DP-SYN.

The details of our technique are shown in Algorithm 3. After partitioning the dataset D into k groups (Line 1 in Alg. 3), the noise injected to each group is also partitioned (Line 4 in Alg. 3), as specified in the sequential composition theorem (Dwork et al., 2006). For each previously obtained group we build one private auto-encoder (Line 5 in Alg. 3), which is detailed in Algorithm 4. Next, we obtain the private latent representation of the group (Line 6 in Alg. 3), and inject it into a differentially private expectation maximization (DP-EM) function. The DP-EM function is detailed in (Park et al., 2016a). The main task of DP-EM is to detect different latent patterns in the encoded data and to generate output

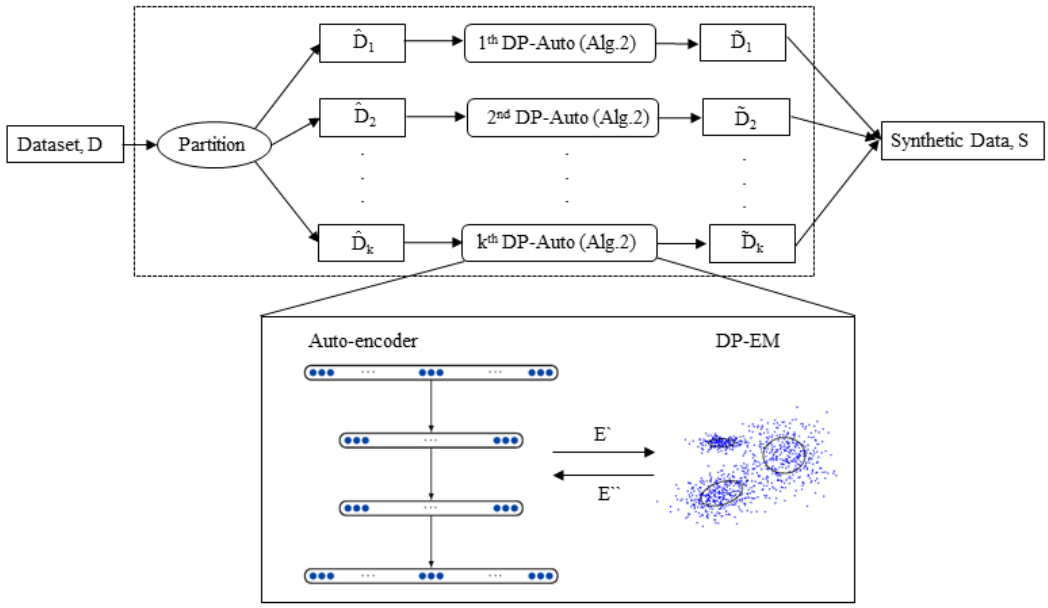


Figure 5.1: Differentially Private Synthetic Data Generation DP-SYN

Algorithm 3 DP-SYN: Differentially Private Synthetic Data Generation

Input: D: $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ where $x \in \mathbb{R}^d$, $y \in \{1,\ldots,\mathrm{k}\}$; η: learning rate; T: iteration number; ε: privacy budget; δ: gaussian delta; σ: standard deviation; C: clipping constant.

Output: S: Synthetic data.

1: $\{\hat{D}_1 \ldots \hat{D}_k\} \leftarrow$ Partition data records in D based on associated labels
2: $S \leftarrow \{\}$
3: **for** $i = 1$ to k **do**
4: Partition ε into $\varepsilon_A = \varepsilon/2$, $\varepsilon_H = \varepsilon/2$ and δ into $\delta_A = \delta/2$, $\delta_H = \delta/2$
5: $W \leftarrow \text{DP-Auto}\left(\hat{D}_i, \eta, \mathrm{T}, \varepsilon_A, \delta_A, \sigma, \mathrm{C}\right)$ // see Alg. 4
6: $E' \leftarrow \textit{encode}\left(W, \hat{D}_i\right)$
7: $E'' \leftarrow \text{DP-EM}\left(E', \varepsilon_H, \delta_H\right)$ // see (Park et al., 2016a)
8: $\tilde{D}_i \leftarrow \textit{decode}\left(W, E''\right)$
9: $S \leftarrow S \cup D'_i$
10: **end for**
11: **return** S

data with similar patterns. Here, DP-EM is used to sample encoded data (Line 7 in Alg. 3) and newly sampled encoded data is decoded with using the model parameter W (Line 8 in

Alg. 3) . $\tilde{D}_i$ is the synthetic data associated with an inputted group $\hat{D}_i$ and appended to the S to be output (Line 9 in Alg. 3).

Algorithm 4 DP-Auto: Differentially Private Auto-encoder

Input: η: Learning rate; T: iteration number; ε_A: privacy budget; δ: gaussian delta; σ: standard deviation; C: clipping constant.

Output: w: Model parameter.

1: ℓ is the objective function
2: $\nabla\ell$ is the gradient of objective function
3: initialize w_0 randomly
4: $\varepsilon'_A = 0$
5: **for** $t = 1$ to T **do**
6: **if** $\varepsilon'_A < \varepsilon_A$ **then**
7: $B_t \leftarrow$ random batch
8: $i_t \sim b$ where $x_{i_t} \in B_t$
9: $z_{i_t} \sim \mathcal{N}(0, \sigma^2 C^2)$
10: $w_{t+1} \leftarrow w_t - \eta \cdot \left(\frac{1}{|B_t|} \sum_{i_t} (\nabla\ell(w_t; x_{i_t}) + z_{i_t}) \right)$
11: $\varepsilon'_A \leftarrow$ calculate privacy loss with moments accountant
12: **end if**
13: **end for**
14: **return** w

5.2.2 Building a Private Auto-Encoder

In this section, we discuss the private auto-encoder given in Algorithm 4.

Our private auto-encoder employs steps to improve the optimization process with gradient computation and clipping. While a gradient is computed for a batch in the standard stochastic training techniques, we compute the gradient for each training instance instead. This approach improves the optimization process since it reduces the sensitivity of the gradient present at each instance (Goodfellow, 2015). Norms of the gradients define the direction that optimizes the network parameters. However, in some deep networks, the gradients can be unstable and fluctuate in a large range. Such fluctuations can inhibit the learning process due to the

increased vulnerability of the networks. To avoid this undesired situation, we bound norms of the previously computed gradients by a clipping constant C (Pascanu et al., 2013).

After clipping the gradients, noise is sampled from the Gaussian distribution with zero mean and standard deviation of σC and added to the previously clipped gradients (Line 9—10 in Alg. 4). At the end of each batch, model parameters of the network are updated with the negative direction of the learning rate η multiplied by the averaged noisy gradients. At the end of this step, the private auto-encoder outputs the model parameter w (Line 11 in Alg. 4).

5.2.3 Privacy Analysis

The privacy analysis of our proposed technique employs the *moments accountant* approach developed by Abadi et. al. (Abadi et al., 2016) to keep track of the privacy cost in multiple iterations. Moments accountant is a combination of both the strong composition theorem (Dwork et al., 2010) and the privacy amplification theorem (Beimel et al., 2010). Moments accountant has an improvement on estimating of the privacy loss for composing differentially private Gaussian mechanisms, and it is the best for overall estimation of privacy budget in literature (Abadi et al., 2016).

In our proposed work, while training the auto-encoder, we track the privacy loss at the end of each batch iteration. As given in Lines 5—11 of Alg. 4, we compute the value of current privacy loss ε' that has been spent on private auto-encoder in a given iteration $t \in T$. Training ends when ε' reaches the final privacy budget ε.

According to moments accountant, Algorithm 4 is (ε, δ)-differentially private if the privacy loss for any $\varepsilon' < k_1(|B|/n)^2 T$ is such that for some constants k_1, k_2:

$$\varepsilon' \geq k_2 \frac{|B|/n\sqrt{T\log[][]1/\delta}}{\sigma},$$

where T is the number of training steps and $|B|$ is the mini-batch for a data of n samples with a given privacy budget ε, gaussian delta δ and standard deviation σ of the Gaussian distribution.

5.3 Experiments

In this section, we present the experimental results to demonstrate the efficiency of our proposed approach. We compare our results with other state-of-the-art techniques. To ensure fairness, we also employ the *Gaussian mechanism* in these techniques.

We start the evaluation by explaining the experimental settings. We evaluate the performance with statistical measures, accuracy in machine learning models and agreement rate. For each task, 70% of the data is used as a training set, while the rest is used for testing.

5.3.1 Experimental Settings

Datasets. We test the proposed approach on nine real datasets. The following is a brief description of each dataset:

1. The **Adult** (Lichman, 2013) dataset contains the information of **45222 individuals**, extracted from the 1994 US census. The dataset shows whether the income of the individuals exceeds fifty thousand US dollars. The dataset contains **15 features**.

2. The **Lifesci** (Lichman, 2013) dataset contains **26733 records** and **10 principal components** from chemistry and biology experiments.

3. The **Optical Digit Recognition** (ODR) (Lichman, 2013) dataset contains **5620 handwritten digits of 10**. Each instance is represented by **64 numeric features**.

4. The **Spambase** (Lichman, 2013) dataset contains **4601 emails**, each of which is labeled as spam or non-spam. Each instance has **58 attributes**.

5. The **Contraceptive Method Choice** (CMC) (Lichman, 2013) dataset contains **9 features** of **1473 married women** to predict their current contraceptive method choice.

6. The **German Credit** (Lichman, 2013) dataset contains the anonymized information of **1000 customers** with **20 features**. Each customer is classified as good or bad credit risk.

7. The **Mammographic Mass** (Lichman, 2013) dataset contains the information of **961 patients'** mammographic masses of with **5 attributes**. The class value shows that patient has breast cancer or not based on mammographic mass.

8. The **Diabetes** (Lichman, 2013) dataset contains the information of female patients who are at least 21 years old. Each patient is classified as diabetic or not diabetic. The dataset has **768 records** with **8 features**.

9. The **BreastCancer** (Lichman, 2013) dataset contains the information about whether a patient has breast cancer or not. It has **699 patient records** with **10 features**.

In all experiments, we compare our results with four state-of-the-art techniques: PrivateSVM (Chaudhuri et al., 2011), PrivBayes (Zhang et al., 2014), DP-EM(SYN) (Park et al., 2016a) and DP-VAE (Ács et al., 2017).

We repeat each experiment 10 times for each task and report the average measurements in our experimental results. In total, our experiments consist of 7840 runs of the mentioned techniques. Here, we only report the best results from each algorithm.

5.3.2 Accuracy in Machine Learning Models

In this set of experiments, we evaluate the accuracy in a Support Vector Machine (SVM) (Hearst et al., 1998) task. More specifically, we report the percentage of incorrectly classified tuples

as the *misclassification rate*. For the PrivateSVM, out of two proposed approaches we only report results from the objective perturbation approach because it outperforms the output perturbation approach.

For each training set, we generate synthetic data by using each method and construct SVM models on the synthetic data. Performance of these models is evaluated on the test set. PrivateSVM has a regularization parameter λ for the objective function. We run PrivateSVM with $\lambda \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-8}\}$ and pick the model that reports the lowest misclassification rate.

Figure 5.2 presents the misclassification rate of the techniques for a given (ε, δ) pair. In the figure, the black straight line shows the misclassification rate on the original dataset, i.e., without privacy. Here, it presents the best case to aim for. We now compare the performance of DP-SYN with respect to each state-of-the-art method.

Figure 5.2 shows that DP-SYN has better performance than PrivBayes for eight out of the nine datasets. Only for the Adult dataset, PrivBayes performs slightly better than our DP-SYN approach. DP-SYN outperforms DP-VAE for seven datasets. For BreastCancer and Diabetes, DP-VAE has better performance; however, it fails to classify any instance in the GermanCredit dataset. For high dimensional datasets such as Spambase and ODR, the misclassification rate of DP-EM(SYN) is two times bigger than that of DP-SYN. A reason for this high misclassification rate shows that DP-EM(SYN) fails in generating synthetic data task when the dimension of data is more than two dozens. DP-SYN also outperforms PrivateSVM in five datasets. PrivateSVM is specifically designed for SVM, and it is expected to have lower misclassification rates in SVM tasks. However, PrivateSVM cannot be employed in other machine learning tasks easily.

Consequently, we point out that on datasets of different types, no single method gives the best misclassification rate consistently. As shown in Figure 5.2-c with the absence of DP-VAE, some algorithms cannot even classify a dataset if the dataset is highly imbalanced.

Considering these issues, Figure 5.2 shows that DP-SYN can be employed on all datasets and reports competitive results.

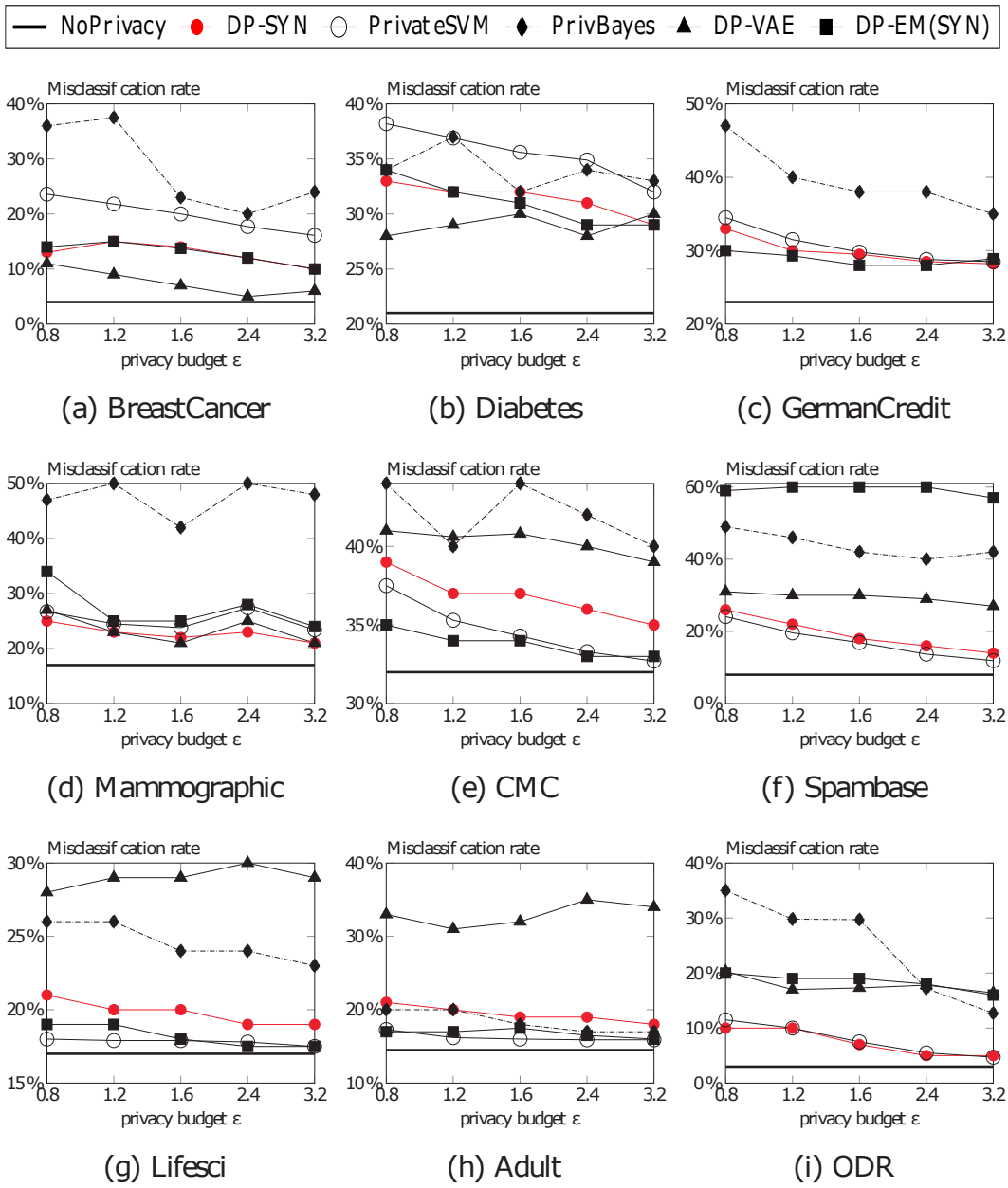


Figure 5.2: Misclassification rates for the nine datasets.

5.3.3 Statistical Measures

We evaluate the quality of synthetic data in terms of statistical utility. We generate k-way marginals of the dataset and compare the probability distribution of the noisy and original marginals. *Total variation distance* (Shah, 2009) is used to report the statistical difference between the noisy and original marginals. The datasets used in the experiments are large, leading to prohibitively large queries. Hence, considering this problem, we generate only 2-way and 3-way marginals as used in (Zhang et al., 2014).

Figure 5.3 shows that DP-SYN performs better than PrivBayes, DP-VAE and DP-EM(SYN) for the 3-way marginals of BreastCancer and Diabetes datasets. In 2-way marginals of BreastCancer and Diabetes datasets, our method performs better than state-of-the-art with the exception of PrivBayes. However, for 2-way marginals of Mammographic, our results are competitive with those of PrivBayes. Overall, DP-SYN preserves the statistical information better than comparable to the state-of-the-art techniques in all datasets.

5.3.4 Agreement Rate

In this section, we evaluate the quality of the synthetic data in terms of the *agreement rate* in an SVM label prediction task. Specifically the agreement rate is defined as the percentage of records for which the two classifiers make the same prediction (Bindschaedler et al., 2017).

Figure 5.4 shows the performance of four techniques in terms of SVM agreement rate and its standard deviation which indicates the certainty and consistency in model predictions.

For the BreastCancer dataset, our approach has the highest agreement rate for privacy loss $\varepsilon \in \{0.8, 1.2, 1.6\}$. For the remaining two cases where $\varepsilon \in \{2.4, 3.2\}$, our approach outperforms DP-VAE and PrivBayes and it has slightly lower agreement rate than DP-EM(SYN). PrivBayes has the lowest agreement rate and the highest standard deviation in most cases. This is expected since PrivBayes does not have much improvement on SVM classification of BreastCancer as previously shown in Section 5.3.2.

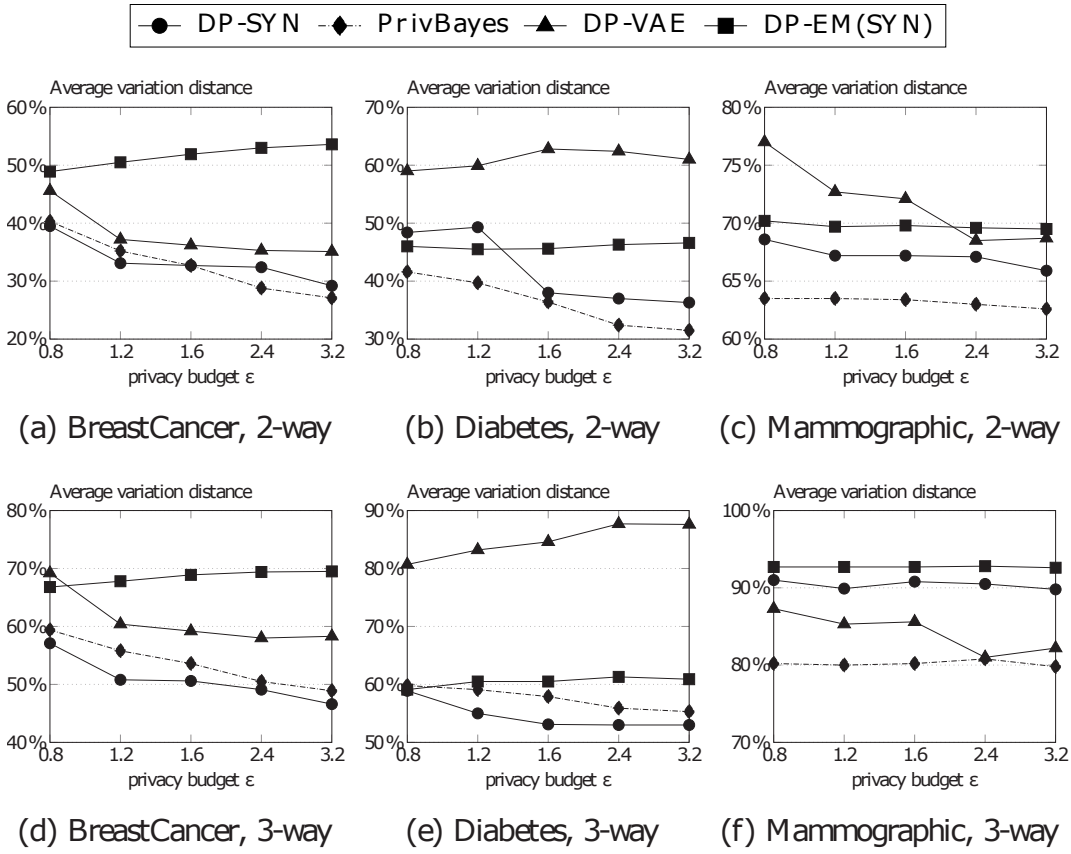


Figure 5.3: Statistical difference between the noisy and original k-way marginals.

For the Spambase and Mammographic datasets, our technique achieves significantly higher agreement rate than that of other state-of-art approaches. For Spambase DP-SYN has lowest standard deviation which indicates high consistency with the SVM classifier that runs on original Spambase training set. We expect such a highest agreement rate because the proposed approach outperforms other techniques in terms of SVM accuracy in Figure 5.2.

For the Adult dataset, the proposed method outperforms DP-VAE, PrivBayes when $\varepsilon \in \{0.8, 1.2\}$. For the remaining cases, the performance of DP-SYN is better than DP-VAE and comparable to DP-EM(SYN) and PrivBayes.

In conclusion, our approach exhibits a significant improvement in the majority of the test cases as evident from SVM agreement rate.

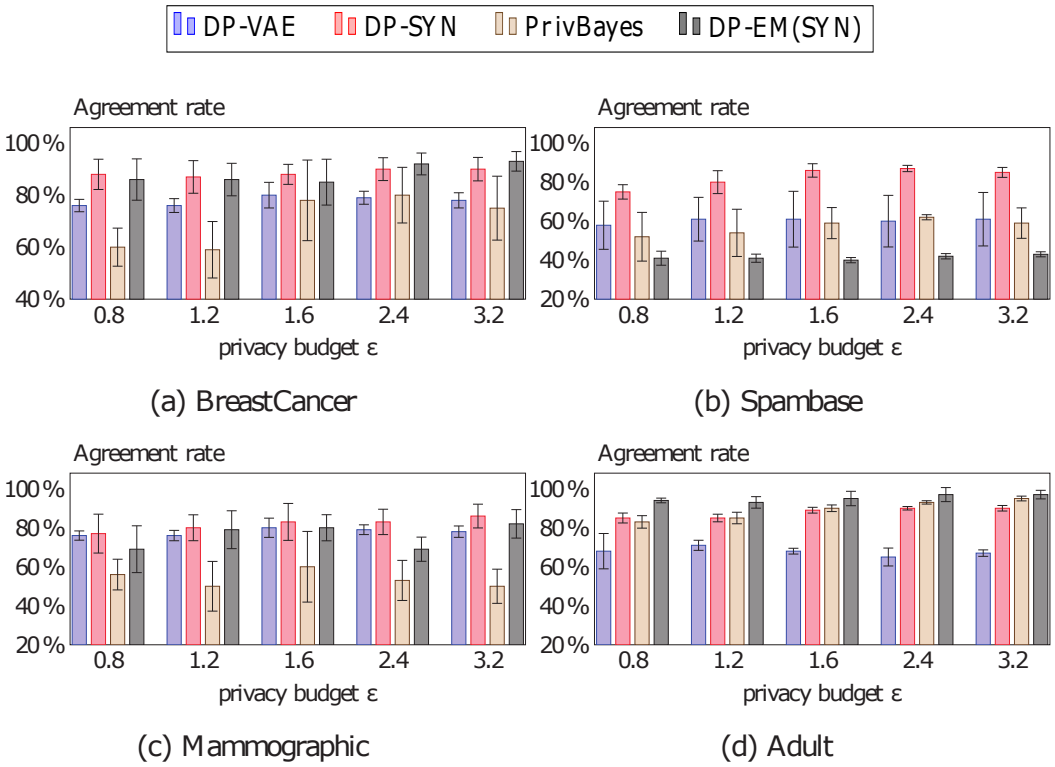


Figure 5.4: SVM agreement rate of the four methods reported on the four datasets.

5.3.5 Impacts on Minority Groups

In the synthetic data generation process, the majority population of the data can be over-presented. In such cases, the generated data may not preserve adequate information about the racial/ethnic minorities of the population.

We evaluate our method in terms of its performance in preserving the minority population information in the data. More specifically, we compute the percentage of misclassified minorities in each dataset. To this end, we create synthetic *Adult* and *CMC* datasets, and compare performance results on them. The *Adult* dataset has three minorities (i.e., Black, Eskimo, Asian) that comprise 13% of the data instances, whereas in the *CMC* dataset the only minority (i.e., Non-Islam) appears in 14% of the data.

Figure 5.5 shows the minority misclassification rates for all methods. For this task, SVM models are trained on synthetic data generated by DP-SYN, PrivateSVM, PrivBayes,

DP-EM(SYN) and DP-VAE. In the figure, the black straight line shows the minority misclassification rate on the original dataset, i.e., without privacy. In both datasets, synthetic data generation increases the minority misclassification rate as much as 3.8 times; in the Adult dataset, misclassification rate for the Eskimo minority increases from 9% to 34% when privacy loss $\varepsilon = 2.4$ for DP-VAE.

In Figure 5.5-b, the Black minority of the Adult dataset shows the lowest increase in misclassification; from the NoPrivacy case to a privacy loss of 0.8, the misclassification rate increases from 9% to less than 15%. We hypothesize that this is due to the high percentage of Blacks (9%) in the population. In Figures 5.5-c and 5.5-d, the rates increase to higher values and show bigger variance because Asian and Eskimo minorities are only 3% and 1% of the Adult dataset; their low percentage coupled with the added privacy result in worse performance.

Our DP-SYN has lower misclassification rates than DP-VAE in all the datasets. Similarly, DP-SYN has better performance than PrivBayes for privacy loss values lower $\varepsilon = 1.6$. The DP-EM(SYN) results are similar or slightly better than those of DP-SYN, but these better results are only available with our modifications to the *DP-EM* algorithm.

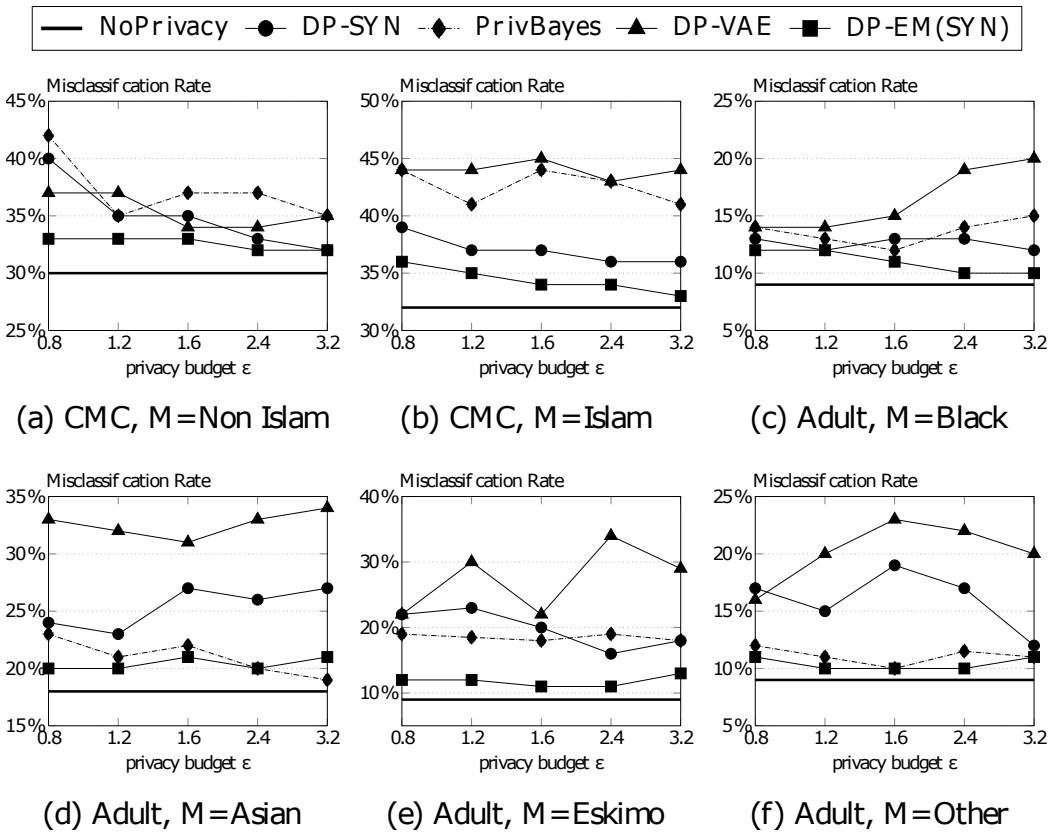


Figure 5.5: Minority misclassification rate for CMC and Adult datasets.

CHAPTER 6

USING DEEP LEARNING TO GENERATE RELATIONAL HONEYDATA [1]

6.1 Introduction

Despite the widespread use of Machine Learning applications, there are many privacy-related problems which have not been addressed so far. To increase the cybersecurity posture of the machine learning applications, in this chapter, we will investigate the generated data in terms of "deceptivity" aspect. Machine learning applications hold our crucial activities (i.e., passwords, credit cards, images) and they are exposed to many adversarial attacks. To enhance the security of the machine learning applications, privacy preserving synthetic data can be also used to deceive the potential cyberattackers. To simulate the test cases, we generate the decoy files with privacy preserving data generation tool and use those files with honeypot environment to track the adversarial behaviours under different scenarios. Honeypots (e.g., (Spitzner, 2003)) have been proposed to provide deceptive targets (i.e., servers) for attackers. Similarly, HoneyFiles (e.g., (Yuill et al., 2004)) have been proposed to lure attackers to spend time in searching files and potentially disclose their intent. Still, to our knowledge, none of the previous work tries to create deceptive 'data' (i.e., HoneyFile) to fool potential attackers. Unfortunately, lack of realistic deceptive data may make it easier for an attacker to detect deception. For example, without good HoneyFile, it may be easier to spot a fake database hosted on a honeypot.

The main purpose of this line of work is to preserve individual privacy while providing data utility. Therefore, it is not clear whether they could be applicable for generating

[1] ©2018 ARO. Reprinted, with permission, from Nazmiye Ceren Abay, Cuneyt G. Akcora, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, "Using Deep Learning to Generate Relational HoneyData", 2018 Autonomous Cyber Deception, Publisher: Springer (ARO), Sep 2018.

good HoneyFile. In the context of cyber deception, it is important that the HoneyFile is indistinguishable from real data so that it can easily fool the attacker.

Creating deceptive data (i.e., HoneyFile) has many challenges. For different settings, we may need different types of HoneyFile. For example, to deceive an attacker and feed false information, deceptive technical plans (e.g., technical drawings of an airplane) could be generated. On the other hand, to make HoneyFiles more believable, fake text data could be added to such files. Since addressing all these different types of data require different techniques, in this chapter, we focus on generating deceptive HoneyFile that is relational data. The main differentiating factor for relational data is that the number of columns and the types of the columns in a given dataset are known in advance. Still, generating realistic relational HoneyFile while not disclosing sensitive information is a significant challenge.

We need to answer questions, such as, 1) how to automatically generate relational HoneyFile? 2) how to measure whether the generated relational HoneyFile is deceptive enough? In this chapter, we try to answer these questions by leveraging existing work in differentially private synthetic data generation and explore its effectiveness for generating relational HoneyFile.

As a part of this chapter, we propose an important measure for understanding the effectiveness of HoneyFile. Basically, given the available information, a potential attacker may not build an effective machine learning model to distinguish between real vs HoneyFile. We evaluate the effectiveness of relational HoneyFile on real datasets, and show under what conditions differentially private deep learning techniques could be used to generate relational HoneyFile.

6.2 Decoy File Data Generation Model

This section describes the main components of our differentially private synthetic data generation approach. We first introduce our private auto-encoder and explain the private

expectation maximization algorithm. Next, we present the privacy analysis of the proposed technique.

6.2.1 Decoy File Data Generation Algorithm

6.2.2 Differentially Private Synthetic Data Generation Algorithm

DpSyn has the primary purpose of generating synthetic data that is indistinguishable from the real data from the attacker's perspective given background knowledge. DpSyn also preserves the privacy by bounding the privacy loss with differential privacy. Abadi et al. (Abadi et al., 2016) applies the moment accountant on differentially private deep learning. Here, we make several modifications to this work and extend it as a data generative model.

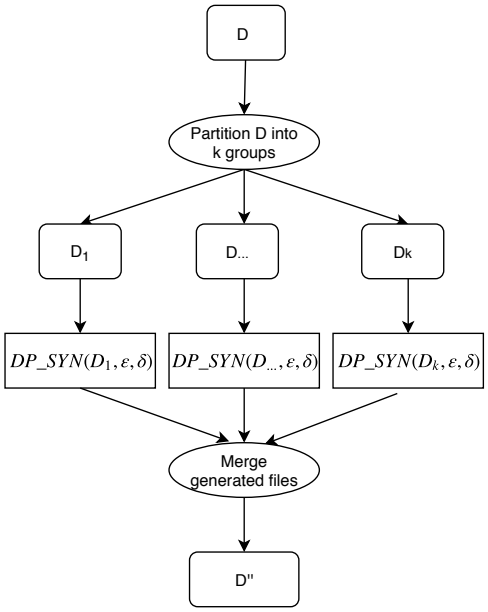


Figure 6.1: Differentially Private Synthetic Data Generation DpSyn

Fig. 6.1 shows the fundamental steps of DpSyn. The dataset D contains a sequence of n training examples $(x_1, y_1), \ldots, (x_m, y_m)$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Our learning approach partitions the dataset D into k groups denoted as $\{D_1, \ldots, D_k\}$. Partitioning of training examples is employed based on label $y \in \mathbb{R}$ associated with training example $x \in \mathbb{R}^d$. Group

number k is identified by the unique label number. After partitioning the dataset into k groups $\{D_1, \ldots, D_k\}$, for each group private generative autoencoder is constructed to generate synthetic data.

More details regarding the aforementioned algorithm has been presented in Chapter 5.

6.3 Experiments

In this section, we explain our experimental setting and discuss our results. First we briefly introduce our datasets and detail parameter settings for the used machine learning models. Afterwards, we give our results for two cyber deception tasks: i) *attacker with no synthetic knowledge* and ii) *attacker with synthetic knowledge*.

6.3.1 Datasets

Datasets. We evaluate the proposed differentially private deep learning based honeydata generation approach on four real datasets. The following is a brief description of each dataset:

1. The **Diabetes** (Lichman, 2013) dataset contains the information of **768 female patients** who are at least 21 years old. Each patient is classified as diabetic or non-diabetic. The dataset contains **8 features**.

2. The **Adult** (Lichman, 2013) dataset contains the information of **45222 individuals**. The dataset shows whether the income of the individuals exceeds 50K US dollars. The dataset contains **15 features**.

3. The **BreastCancer(Diagnostic)** (Lichman, 2013) dataset contains the information about whether a patient has breast cancer or not. It has **569 patient records** with **32 features**.

4. The **Spambase** (Lichman, 2013) dataset contains **4601 emails**, each of which is labeled as spam or non-spam. Each instance has **58 attributes**.

6.3.2 Parameter Settings

Parameter setting for Data Generation. Our DpSyn technique generates synthetic data by using Deep Auto-encoders (Baldi, 2012). An auto-encoder is trained on n data points. Once a model is learned, the auto-encoder can be used to generate *any number of data points* (e.g., honeydata). For n training samples, we report the results of the privacy loss (i.e., the measure of potential leakage to an attacker) using differential privacy with (ε, δ) parameters that is computed from the noise level σ. We fix the δ as $\frac{1}{n}$, and compute the value of ε for each iteration $t \in T$. In moment accountant, we use several noise levels to obtain consistent results. The large noise level ($\sigma = 6.0$) is implemented for small $\varepsilon = 1.0$ and the small noise level ($\sigma = 4.0$) is implemented for large $\varepsilon \in \{2.0, 4.0\}$. In these settings with the increasing ε values synthetic data generation techniques are perturbed less since small noise is added to these techniques.

In all synthetic datasets (i.e., the generated relational honeydata), biases are initialized to zero, while the initial values of the weights θ are randomly chosen from a zero-mean normal distribution with a standard deviation of 0.05. For each dataset, we form a new auto-encoder to generate its corresponding honeydata.

Parameter setting for Machine Learning Models. We employ four machine learning models in measuring the efficiency of our approach in synthetic data generation for cyber deception: One-class SVM (Schölkopf et al., 2001), two-class SVM (Hearst et al., 1998), Logistic Regression (LR) (Nerlove and Press, 1973) and Random Forest (RF) (Breiman, 2001). We chose to employ these methods because they are widely used for classification tasks (Kotsiantis et al., 2007). Furthermore, these machine learning models will be used to explore whether an attacker can distinguish between the real data vs the honeydata easily.

For the hyper parameter of one-class SVM, we experiment with $\{linear, poly, rbf\}$ and gamma values $\{1.0, 0.1, 0.01, 0.001\}$. We select the most consistent results of one-class SVM with different (ε, δ) pairs. For two class SVM, we employ the LinearSVM (Fan et al., 2008).

6.3.3 Benchmark Techniques

In the first task, we assume that the attacker has knowledge about real data where we model the background knowledge as the number (i.e., $\{50, 100, 200, 400\}$) of data points known to the attacker. This approach is similar to the setting reported in (Bindschaedler et al., 2017). In this scenario, the attacker does not have access to honeydata samples. We evaluate the quality of the generated honeydata by observing whether the attacker can distinguish the real vs honeydata by leveraging the obtained real data. In these experiments, the attacker employs a one-class SVM model that is built on the $\{50, 100, 200, 400\}$ real data points.

We measure the success of DpSyn by the attacker's failure to separate honeydata and real data by using the classifier. The test data is an equal mix of 50% real and 50% honeydata. *A honeydata generation technique achieves best results if the attacker's SVM model labels all synthetic data as real, which results in a 50% accuracy.*

Fig. 6.2 shows the performance of techniques in deceiving the attacker on the four real datasets. In Fig. 6.2a, we show performance for various training number sizes $50, 100, 200$ and 400.

Each training set is used by increasing noise additions, which is calibrated to $\varepsilon = 1.0, 2.0$ and 4.0, where $\varepsilon = 4.0$ shows the least amount of added noise in the model. As training size increases, performance of all three techniques improve and accuracy values approach 50%. In Spambase and BreastCancer datasets PrivBayes and DpEm$^+$ perform worse compared to Adult and Diabetes datasets. We hypothesize that PrivBayes and DpEm$^+$ are more vulnerable to the curse of high dimensionality; both Adult and Diabetes datasets have less than 15 attributes, whereas BreastCancer and Spambase have more than 32 attributes.

DpSyn results are comparable with those of DpEm$^+$ and PrivBayes in the Adult dataset. In Diabetes and Spambase datasets, DpSyn has better accuracy. PrivBayes has its worst results in the BreastCancer dataset. In all figures, increasing ε values result in values closer to the desired 50% accuracy value.

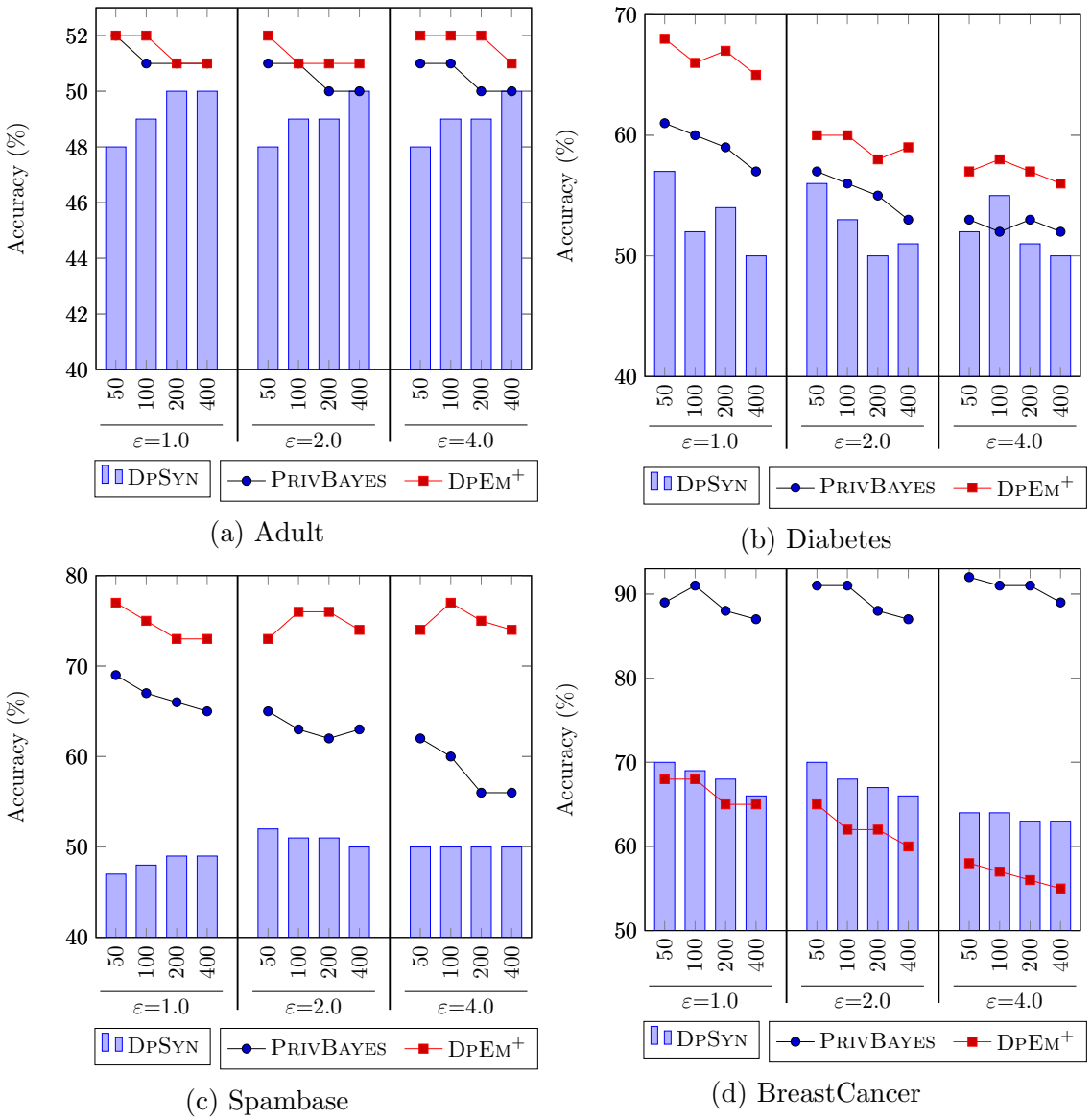


Figure 6.2: [Color online]. Accuracy results of one-class SVM classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%.

In the first task, the attacker had access to real data only. In this second task, the attacker has access to both real and honeydata, which may help the attacker in distinguishing honeydata. The knowledge of honeydata implies that attacker was fooled into accepting some honeydata previously; when the attacker fails while using the honeydata (e.g., could

not use the honeydata for identity theft), he/she may start analyzing other stored data files to authenticate them.

The attacker employs three binary classification models (i.e., two-class SVM, LR and RF) that are built on a mix of 50% real and 50% synthetic data. The classifier is trained to learn two labels: real and synthetic (i.e., honeydata). The classifier is tested on a mix of 50% real and 50% honeydata. The accuracy of a model is given as the percentage of correctly classified data points. For the best performance in honeydata generation, the attacker must wrongly classify all honeydata points as real, which results in a 50% accuracy.

In this set of experiments, we report our results for each of the three Machine Learning classification models separately. We begin by demonstrating the Random Forest results.

Fig. 6.3 demonstrates accuracy of the three techniques for the Random Forest classifier. The honeydata generated by DpSyn is more similar to real data when compared to the synthetic data generated from PrivBayes for Diabetes, Spambase and BreastCancer datasets. For the Adult dataset DpEm$^+$ results are closer to the desired 50% level. DpSyn performs better than DpEm$^+$ for Diabetes and Spambase. The failure of DpEm$^+$ and PrivBayes on Spambase and Diabetes datasets is expected since the attacker can already distinguish the difference between synthetic and real data with one-class SVM. DpSyn exhibits remarkable improvement for the majority of the RF test cases when compared to DpEm$^+$ and DpSyn.

The performance of the attacker with Logistic Regression (LR) is demonstrated in Fig. 6.4. The test results of LR is consistent with the other machine learning models previously shown in Figs. 6.3 and 6.2. In fact, on average, LR results are highly correlated (0.94) with RF results in all datasets. However, it is noticeable that attacker with RF is better able to distinguish between real and honeydata when compared to LR; accuracy levels are lower in the LR results. DpSyn outperforms PrivBayes in three out of four datasets. For Adult dataset PrivBayes has slightly better results than DpSyn. For this set of experiments with LR, DpSyn results are similar to those of DpEm$^+$ in the Diabetes, Spambase and BreastCancer datasets.

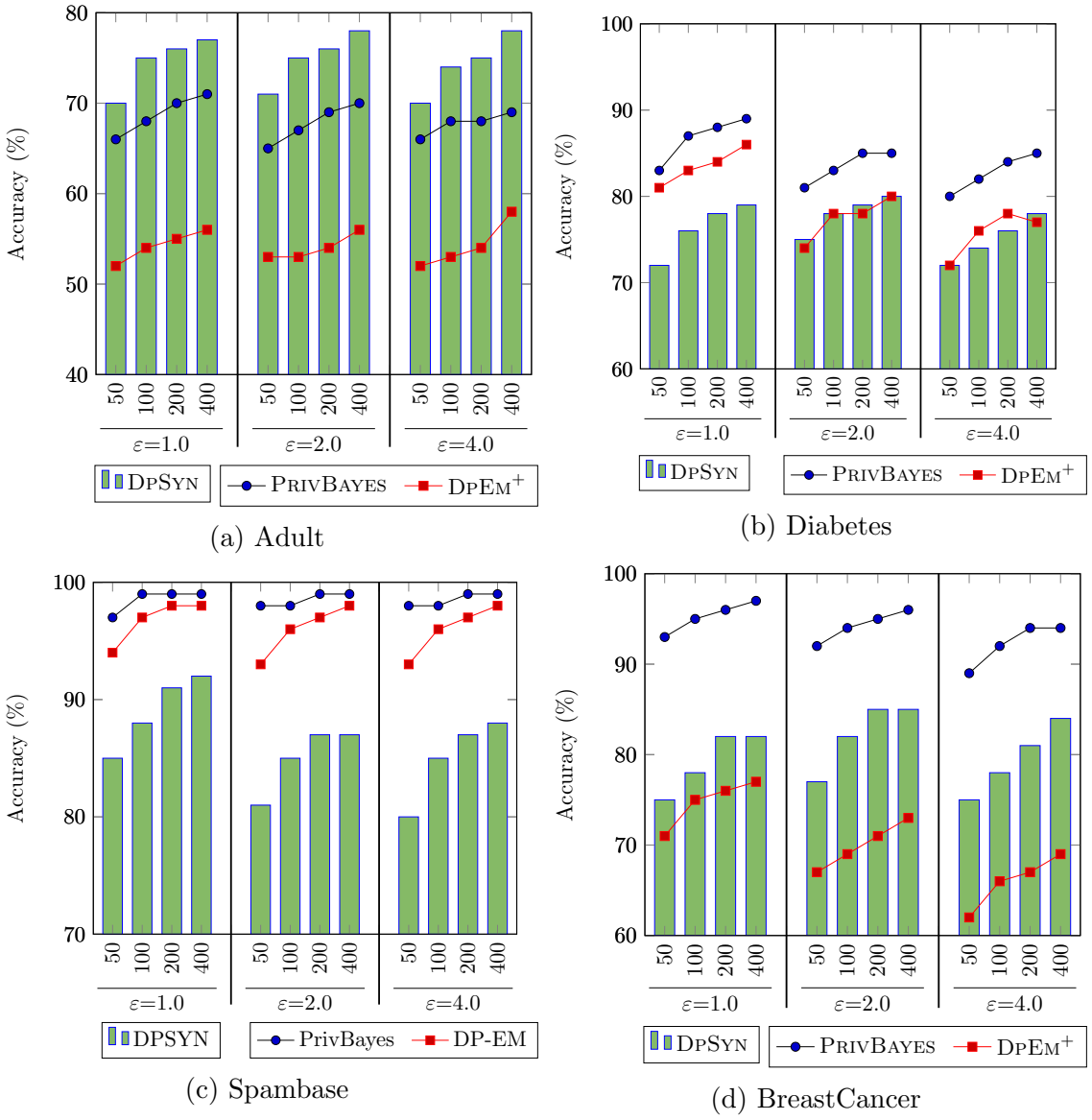


Figure 6.3: [Color online]. Accuracy results of RF classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%.

The performance of the attacker with two-class SVM is demonstrated in Fig. 6.5. Results are consistent with those of LR and RF in Figs. 6.4 and 6.3. Only in the Adult dataset, we see a slight increase of difference between PRIVBAYES and DPEM$^+$ performances with the increase of training numbers.

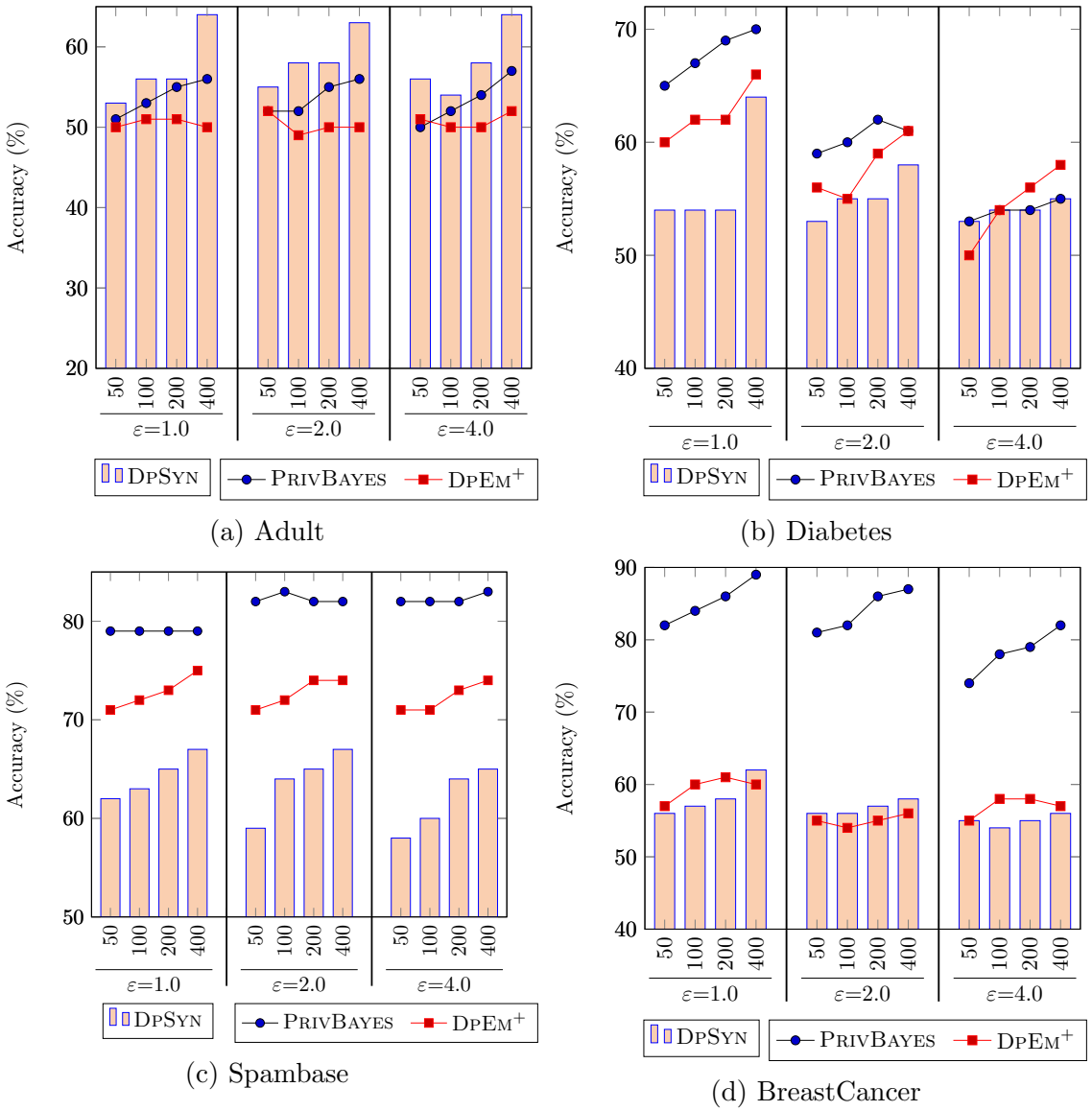


Figure 6.4: [Color online]. Accuracy results of LR classifiers of attackers that are modeled on different number of real data with varying privacy budgets. The desired accuracy is 50%.

In all Machine Learning models, our method DpSyn generates honeydata that has better indistinguishability (i.e., the attacker has less accuracy in distinguishing real vs honeydata) for cyber deception. With increasing training dataset size, the classifiers that could be used by the attacker perform better in all data generation techniques. Except for the Adult dataset, DpSyn outperforms PrivBayes significantly in all experimental

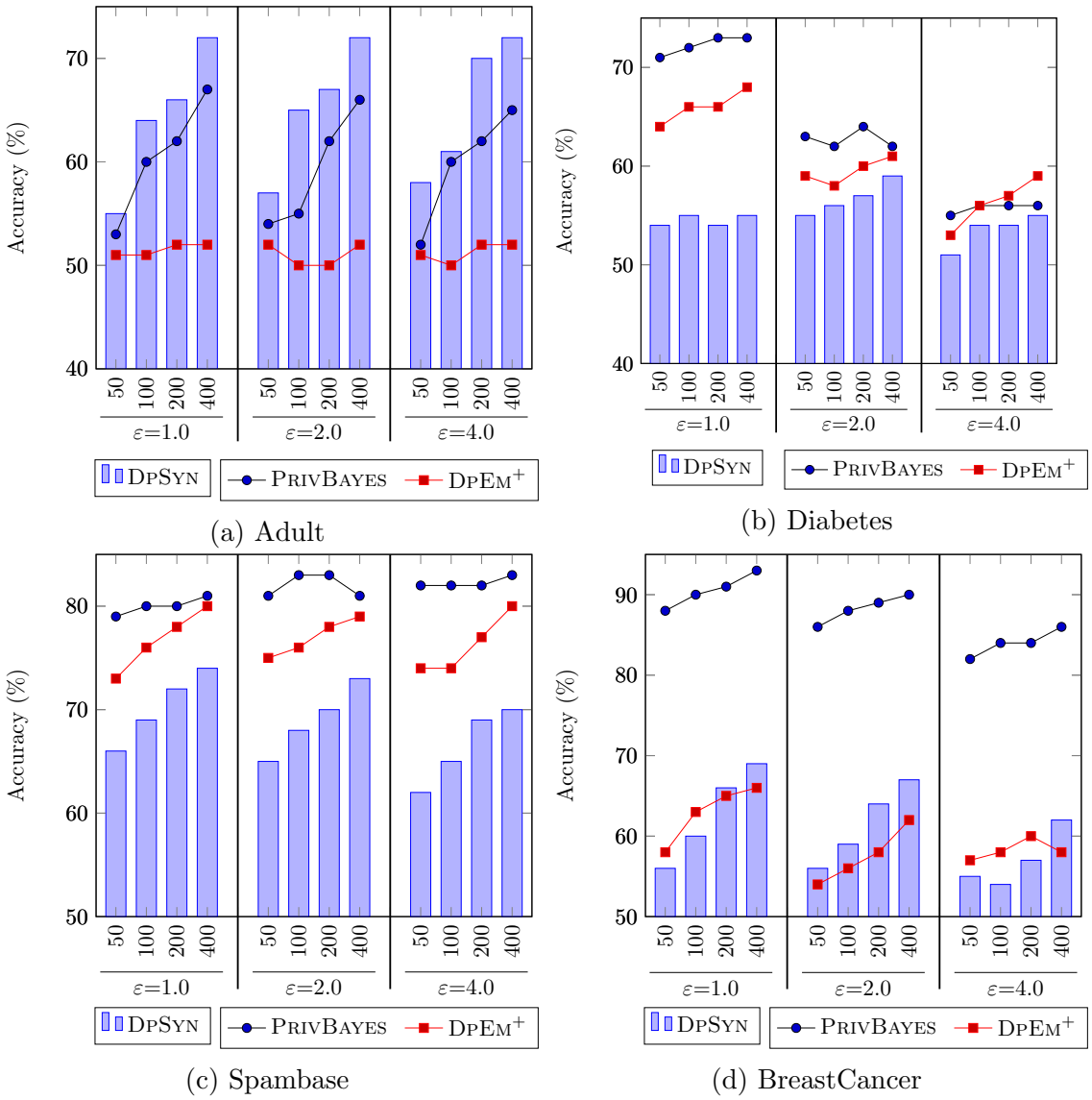


Figure 6.5: [Color online]. Accuracy results of binary SVM classifiers of attackers that are modeled on different percentages of real data with varying privacy budgets. The desired accuracy is 50%.

settings. Compared to DPEM$^+$, DPSYN generates better or comparable synthetic data for the Spambase, BreastCancer and Diabetes datasets. For the majority of the test cases, increasing ε values harm the attacker machine learning model. However, increasing ε values result in less added noise which may cause the leakage of sensitive data. Hence, there is a trade-off between the quality of honeydata and the prevention of sensitive data leakage.

CHAPTER 7

CONCLUSION

This dissertation addresses the predictive accuracy, privacy protection and cybersecurity issues in machine learning applications.

First, we started by developing the crypto-currency price prediction machine learning (ML) models to predict the Bitcoin price. To have accurate Bitcoin price prediction, our proposed ChainNet utilizes topological characteristics of a blockchain graph. ChainNet extracts topological constructs over a graph and computes quantitative summaries in the form of Betti numbers and Betti derivatives which are then used in model building for the Bitcoin price prediction.

Our results on the full Bitcoin network show that in less than 7 day ahead predictions, Betti models bring a prediction gain of almost 40% over baseline approaches.

In the second part of our research, for ML applications that need to access the sensitive information, we propose a new generative deep learning method, DP-SYN, that produces privacy-preserving synthetic data from a dataset while preserving the utility of the original dataset. Our generative auto-encoder method partitions the original data into groups, and then employs the private auto-encoder for each group. Auto-encoder learns the latent structure of each group, and uses expectation maximization algorithm to generate synthetic data. This approach eliminates impurity of groups and results in more accurate representations for each latent group. We test DP-SYN on nine datasets and compare to four state-of-the art methods reported for privacy-preserving synthetic data generation. Our evaluation process uses statistical, ML based and agreement rate based metrics. Although not a single method outperforms others consistently in all tasks, our experiments show that DP-SYN gives robust results across all datasets, and performs better than state-of-the-art in multiple settings for both relational and image based datasets. Furthermore, DP-SYN performance does not deteriorate when the original dataset is imbalanced or high dimensional.

Finally, we explore the applicability of privacy-preserving deep learning based synthetic data generation techniques for creating HoneyData that can fool potential cyber-attackers. We define a ML based metric (i.e., the accuracy of any ML model in distinguishing real vs HoneyData) to measure the goodness of generated deceptive HoneyData. Although, our results indicate that existing techniques could be leveraged for HoneyData generation, care must be taken in setting the privacy parameters used in HoneyData generation.

REFERENCES

Abadi, M., A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318. ACM.

Abay, N. C., C. G. Akcora, Y. R. Gel, M. Kantarcioglu, and B. Thuraisingham (2019). A technical report on chainnet. `https://github.com/cakcora/CoinWorks/blob/master/chainnetreport.pdf`.

Ács, G., L. Melis, C. Castelluccia, and E. D. Cristofaro (2017). Differentially private mixture of generative neural networks. *CoRR abs/1709.04514*.

Akcora, C. G., A. K. Dey, Y. R. Gel, and M. Kantarcioglu (2018). Forecasting bitcoin price with graph chainlets. In *The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Melbourne, Australia*, pp. 1–12.

Almeshekah, M. H. and E. H. Spafford (2016). Cyber security deception. In *Cyber Deception*, pp. 23–50. Springer.

Androulaki, E., G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun (2013). Evaluating user privacy in bitcoin. In *IFCA*, pp. 34–51. Springer.

Antulov-Fantulin, N., D. Tolic, M. Piskorec, Z. Ce, and I. Vodenska (2018). Inferring short-term volatility indicators from the bitcoin blockchain. In *International Workshop on Complex Networks and their Applications*, pp. 508–520. Springer.

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pp. 37–49.

Barak, B., K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar (2007). Privacy, accuracy, and consistency too: A holistic solution to contingency table release. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, New York, NY, USA, pp. 273–282. ACM.

Baumann, A., B. Fabian, and M. Lischke (2014). Exploring the bitcoin network. In *WEBIST (1)*, pp. 369–374.

Beimel, A., S. P. Kasiviswanathan, and K. Nissim (2010). Bounds on the sample complexity for private learning and private data release. In *TCC*, Volume 5978, pp. 437–454. Springer.

Bindschaedler, V., R. Shokri, and C. A. Gunter (2017). Plausible deniability for privacy-preserving data synthesis. *Proceedings of the VLDB Endowment 10*(5), 481–492.

Blum, A., C. Dwork, F. McSherry, and K. Nissim (2005). Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 128–138. ACM.

Box, G. E., G. M. Jenkins, G. C. Reinsel, and G. M. Ljung (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Breiman, L. (2001). Random forests. *Machine learning 45*(1), 5–32.

Carlsson, G. (2009). Topology and data. *Bulletin of American Mathematical Society (N.S.) 46*(2), 255–308.

Chaudhuri, K. and C. Monteleoni (2009). Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, pp. 289–296.

Chaudhuri, K., C. Monteleoni, and A. D. Sarwate (2011). Differentially private empirical risk minimization. *Journal of Machine Learning Research 12*(Mar), 1069–1109.

Chazal, F. and B. Michel (2017). An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists. *ArXiv e-prints*, 1–38.

Chen, T. and C. Guestrin (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794. ACM.

Di Battista, G., V. Di Donato, M. Patrignani, M. Pizzonia, V. Roselli, and R. Tamassia (2015). Bitconeview: visualization of flows in the bitcoin transaction graph. In *IEEE VizSec*, pp. 1–8.

Ding, B., M. Winslett, J. Han, and Z. Li (2011). Differentially private data cubes: optimizing noise sources and consistency. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 217–228. ACM.

Dwork, C. (2006). Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, Berlin, Heidelberg, pp. 1–12. Springer-Verlag.

Dwork, C., K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor (2006). Our data, ourselves: Privacy via distributed noise generation. In *Eurocrypt*, Volume 4004, pp. 486–503. Springer.

Dwork, C., F. McSherry, K. Nissim, and A. Smith (2006). Calibrating noise to sensitivity in private data analysis. In *TCC*, Volume 3876, pp. 265–284. Springer.

Dwork, C., A. Roth, et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science 9*(3–4), 211–407.

Dwork, C., G. N. Rothblum, and S. Vadhan (2010). Boosting and differential privacy. In *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pp. 51–60. IEEE.

Dyhrberg, A. H. (2016). Bitcoin, gold and the dollar–a garch volatility analysis. *Finance Research Letters 16*, 85–92.

Edelsbrunner, H. and S. Parsa (2014). On the computational complexity of betti numbers: reductions from matrix rank. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on discrete algorithms*, pp. 152–160. SIAM.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin (2008). Liblinear: A library for large linear classification. *Journal of machine learning research 9*(Aug), 1871–1874.

Filtz, E., A. Polleres, R. Karl, and B. Haslhofer (2017). Evolution of the bitcoin address graph.

Garg, A., D. Lu, K. Popuri, and M. F. Beg (2016). Cortical geometry network and topology markers for parkinsons disease. *arXiv preprint:1611.04393*, 1–10.

Gionis, A., T. Lappas, and E. Terzi (2012). Estimating entity importance via counting set covers. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 687–695. ACM.

Goodfellow, I. (2015). Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*.

Greaves, A. and B. Au (2015). Using the bitcoin transaction graph to predict the price of bitcoin. *No Data*.

Hearst, M. A., S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf (1998). Support vector machines. *IEEE Intelligent Systems and their applications 13*(4), 18–28.

Henelius, A., A. Ukkonen, and K. Puolamäki (2016). Finding statistically significant attribute interactions. *arXiv preprint arXiv:1612.07597*.

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine 29*(6), 82–97.

Ho, T. K. (1995, Aug). Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Volume 1, pp. 278–282 vol.1.

Hofer, C., R. Kwitt, M. Niethammer, and A. Uhl (2017). Deep learning with topological signatures. *Advances in Neural Inf. Processing Systems 30*, 1634–1644.

Holz, T. and F. Raynal (2005). Detecting honeypots and other suspicious environments. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pp. 29–36. IEEE.

Hyndman, R. J. and Y. Fan (1996). Sample quantiles in statistical packages. *The American Statistician 50*(4), 361–365.

Jagannathan, G., K. Pillaipakkamnatt, and R. N. Wright (2009). A practical differentially private random decision tree classifier. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, pp. 114–121. IEEE.

Jog, V. and P. Loh (2015). Recovering communities in weighted stochastic block models. In *53rd Annual Allerton Conference on Communication, Control, and Computing, Monticello, USA*, pp. 1308–1315.

Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science*, pp. 1094–1096. Springer.

Juels, A. and R. L. Rivest (2013). Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 145–160. ACM.

Kondor, D., I. Csabai, J. Szüle, and G. Pósfai, M.and Vattay (2014). Inferring the interplay between network structure and market effects in bitcoin. *New J. of Phys. 16*(12), 125003.

Kondor, D., M. Pósfai, I. Csabai, and G. Vattay (2014). Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one 9*(2), e86197.

Kotsiantis, S. B., I. Zaharakis, and P. Pintelas (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering 160*, 3–24.

Kristoufek, L. (2015). What are the main drivers of the bitcoin price? evidence from wavelet coherence analysis. *PloS One 10*(4).

Li, N., W. Qardaji, D. Su, and J. Cao (2012). Privbasis: Frequent itemset mining with differential privacy. *Proceedings of the VLDB Endowment 5*(11), 1340–1351.

Lichman, M. (2013). UCI machine learning repository.

Lischke, M. and B. Fabian (2016). Analyzing the bitcoin network: The first four years. *Future Internet 8*(1), 7.

Madan, I.and Saluja, S. and A. Zhao (2015). Automated bitcoin trading via machine learning algorithms.

Mattila, J. et al. (2016). The blockchain phenomenon–the disruptive potential of distributed consensus architectures. Technical report, The Research Institute of the Finnish Economy.

Milo, R., S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon (2002). Network motifs: Simple building blocks of complex networks. *Science 298*(5594), 824–827.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Nanda, V. (2017). Perseus: the persistent homology software. *http://people.maths.ox.ac.uk/nanda/perseus/index.html*.

Nerlove, M. and S. J. Press (1973). *Univariate and multivariate log-linear and logistic models*, Volume 1306. Rand Santa Monica.

Ober, M., S. Katzenbeisser, and K. Hamacher (2013). Structure and anonymity of the bitcoin transaction graph. *Future internet 5*(2), 237–250.

Park, M., J. Foulds, K. Chaudhuri, and M. Welling (2016a). Dp-em: Differentially private expectation maximization. *arXiv preprint arXiv:1605.06995*.

Park, M., J. Foulds, K. Chaudhuri, and M. Welling (2016b). Practical privacy for expectation maximization. *CoRR abs/1605.06995*.

Pascanu, R., T. Mikolov, and Y. Bengio (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318.

Qardaji, W., W. Yang, and N. Li (2014). Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1435–1446. ACM.

Ron, D. and A. Shamir (2013). Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pp. 6–24. Springer.

Rubin, D. B. (1993). Discussion statistical disclosure limitation. *Journal of official Statistics 9*(2), 461.

Rubinstein, B. I., P. L. Bartlett, L. Huang, and N. Taft (2009). Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*.

Schölkopf, B., J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson (2001, July). Estimating the support of a high-dimensional distribution. *Neural Comput. 13*(7), 1443–1471.

Shah, D. and K. Zhang (2014). Bayesian regression and bitcoin. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pp. 409–414. IEEE.

Shah, I. M. (2009). Introduction to nonparametric estimation. *Investigación Operacional 30*(3), 284–285.

Shervashidze, N., S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt (2009). Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495.

Song, S., K. Chaudhuri, and A. D. Sarwate (2013). Stochastic gradient descent with differentially private updates. In *GlobalSIP, 2013 IEEE*, pp. 245–248. IEEE.

Sorgente, M. and C. Cibils (2014). The reaction of a network: Exploring the relationship between the bitcoin network structure and the bitcoin price. *No Data*.

Spitzner, L. (2003). *Honeypots: tracking hackers*, Volume 1. Addison-Wesley Reading.

Swanson, T. (2014). Learning from bitcoin's past to improve its future.

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*(05), 557–570.

Topaz, C., L. Ziegelmeier, and T. Halverson (2015). Topological data analysis of biological aggregation models. *PLoS ONE 10*(5).

Tschorsch, F. and B. Scheuermann (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Comm. Surveys 18*(3), 2084–2123.

Vaidya, J., B. Shafiq, A. Basu, and Y. Hong (2013). Differentially private naive bayes classification. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, pp. 571–576. IEEE Computer Society.

Vogel, P., T. Greiser, and D. C. Mattfeld (2011). Understanding bike-sharing systems using data mining: Exploring activity patterns. *Procedia-Social and Behavioral Sciences 20*, 514–523.

Williams, C. K. and C. E. Rasmussen (1996). Gaussian processes for regression. In *Advances in neural information processing systems*, pp. 514–520.

Wong, C. (2017). Nyc taxi trips dataset. Online.

Yang, S. Y. and J. Kim (2015). Bitcoin market return and volatility forecasting using transaction network flow properties. In *IEEE SSCI*, pp. 1778–1785.

Yuill, J., M. Zappe, D. Denning, and F. Feer (2004). Honeyfiles: deceptive files for intrusion detection. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pp. 116–122. IEEE.

Zeng, C., J. F. Naughton, and J.-Y. Cai (2012). On differentially private frequent itemset mining. *Proceedings of the VLDB Endowment 6*(1), 25–36.

Zhang, J., G. Cormode, C. M. Procopiuc, D. Srivastava, and X. Xiao (2014). Privbayes: Private data release via bayesian networks. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 1423–1434. ACM.

Zhang, J., Z. Zhang, X. Xiao, Y. Yang, and M. Winslett (2012). Functional mechanism: regression analysis under differential privacy. *Proceedings of the VLDB Endowment 5*(11), 1364–1375.

Zomorodian, A. (2010). Fast construction of the vietoris-rips complex. *Computers and Graphics 34*(3), 263–271.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B 67*(2), 301–320.

BIOGRAPHICAL SKETCH

Nazmiye Ceren Abay received her bachelor's in Computer Engineering from Middle East Technical University in Ankara, Turkiye in 2011. After her graduation, she started her master's in Computer Engineering at Middle East Technical University in 2014. She also worked at Space and Defense Company, Havelsan, as a software engineer for four years. After she received her master's degree, she joined the doctoral program in Computer Science at The University of Texas at Dallas in 2015. Her research interests are investigating predictive accuracy and privacy preserving in data generation.

CURRICULUM VITAE

Nazmiye Ceren Abay

Contact Information:

Department of Computer Science
The University of Texas at Dallas
800 W. Campbell Rd.
Richardson, TX 75080-3021, U.S.A.

Email: crnabay@gmail.com,
nca150130@utdallas.edu

Educational History:

B.S., Computer Engineering, Middle East Technical University, Turkey, 2011

M.S., Computer Engineering, Middle East Technical University, Turkey, 2011
Thesis: ILP-Based and Graph-Based Concept Discovery Systems
Advisor: Dr. Pinar Karagoz

Ph.D., Computer Engineering, University of Texas at Dallas, USA, 2019
Dissertation: Machine Learning Techniques for Prediction and Data Generation with Privacy
Advisors: Dr. Bhavani Thuraisingham, Dr. Murat Kantarcioglu

Employment History:

Graduate Research Assistant, The University of Texas at Dallas, Sep 2015 – Sep 2019
Software Development Engineer, Havelsan Inc, Sep 2011 – Jun 2015

Honors & Accomplishments:

Women in Cybersecurity (WiCyS) Graduate Poster Competition Winner, 2019
Graduated as an honor student from Middle East Technical University, 2011
Ranked 1201^{st} among 1.7 million participants in University Entrance Exam in Turkey, 2006