TECHNICAL UNIVERSITY OF CRETE

Data Evaluation using Shapley Values

Author: Mavrogiorgis Dimitrios

Committee: **Garofalakis Minos (Supervisor)**

Samoladas Vasileios

Deligiannakis Antonios

Acknowledgements

After 5 years at the Technical University of Crete, I want to express my deepest gratitude to all the people that supported me throughout these difficult, yet fulfilling years.

First and foremost, I would like to thank all the professors and the assistants of Technical University of Crete because they have helped me gain some major knowledge in the field of Electrical and Computer Engineering.

Last but not least, this journey wouldn't be possible without the help of my family and friends. I owe a lot to my parents who supported me through every decision I have made and gave me the opportunity to pursue all of my goals. I also want to express my gratitude for my friends, who were there for me both in good and in bad times.

Abstract

Purpose: The main purpose of the current work is to calculate the Data Shapley value of specific data sets by leveraging the mechanisms of differential privacy algorithms to ensure some guarantee of privacy. To fulfil this objective, the current thesis first studies and presents the theoretical foundation of the Shapley value calculations. Calculation methods, such as the Truncated Monte Carlo - Shapley, Gradient - Shapley and Group - Shapley are studied and analyzed, while at the same time this thesis proposes how new approaches guarantee the correctness and accuracy of calculations without any information leakage.

Design / methodology / approach: This study first presents the theoretical foundation of both the Shapley valuation and the differential privacy scientific domains, including mathematical analysis and the relative practical research evidence. On top, a case study, i.e. a classification problem with logistic regression has been designed, implemented, analyzed and presented. The code of the problem includes set up activities, data set creation, execution of three data Shapley algorithms with and without embedded differential data privacy noise and graphical presentation of the results obtained. Convergence of the applied algorithms is proved, while at the same time the performance of the studied algorithms is also evaluated for both non-differential private and noisy versions of the implemented algorithms.

Findings: Both TMC-Shapley and G-Shapley methods outperform the LOO method, since the accuracy is reduced faster when bigger fractions of trained data are being removed. For the differential private TMC-Shapley and G-Shapley methods, a number of graphs with all marginals is being created which proves the convergence of the algorithms. The performance of the studied noisy data Shapley methods is also demonstrated. Both Noisy TMC-Shapley and G-Shapley methods outperform the Noisy LOO and Random versions.

Originality / value: The value added by the current work is coming from the following areas: (1) Provision of an efficient formulation of the important problem of calculating the fair value of data sets, known as Data Shapley valuation problem, by taking advantage of machine learning techniques, (2) Review of existing empirical studies that prove the efficiencies and all desirable properties of the Data Shapley valuation method, (3) Introduction of a Data Shapley calculation algorithm variations that are suitable to address the problem under study, (4) Application of

sensitivity and noise to ensure data privacy mechanism as well as to measure and analyze data Shapley value in the context of differential privacy, and (5) Experimental implementation developed using Python programming language.

Keywords: Data Shapley, Differential Privacy (DP), Truncated Monte Carlo Shapley (TMC-Shapley), Gradient Shapley (G-Shapley).

Table of Contents

List of Figures

List of Tables

Abbreviations

CA	Calculation Algorithm
DP	Differential Privacy
LOO	Leave One Out
SEA	Spatial Estimation Algorithm
TEA	Temporal Estimation Algorithm

1 INTRODUCTION

1.1 Problem statement and importance

In recent days, data has become a significant driver of technological and economic growth, triggering among others the fundamental challenge of how to materialize and measure its value in the context of algorithmic forecasts and decision making. As part of the discussions that have come up, researchers and practitioners propose that individuals that share their data in industries such as healthcare and retail markets, should be compensated; still, it is unclear what should be the fair price of this data.

A quite popular framework developed to address the data valuation challenge, in the context of supervised machine learning, is based on the data Shapley metrics. Given a number of data points used to produce a prediction value, the data Shapley metric quantifies the contribution of each individual datum to the overall prediction performance. Data Shapley metrics are suitable for data pricing purposes since they satisfy important properties of a fair data valuation approach. A large number of practical experiments have also proven that data Shapley methods present additional benefits; specifically, they are more effective compared to traditional "*leave one out*" or scoring methods, while they also produce outliers and insights on the type and nature of additional data that might be candidate to improve the produced predictions.

On the other hand, differential privacy could be defined as a set of methods and tools capable for quantifying and solving practical data privacy problems. Differential privacy practices could be applied in the context of various applications, including Machine and Deep Learning applications when private data are used. Combination of those practices with the Data Shapley approach applied to quantify the fair data value for individuals, is an interesting research topic that will be studied in the framework of the current work.

1.2 Aims and objectives

The main objective of the current work is to calculate the Data Shapley value of specific data sets by leveraging one of the mechanisms of differential privacy algorithms (i.e. Laplacian, Gaussian,

Exponential) and providing some guarantee of privacy, as defined in the differential privacy domain. To fulfil this objective, the current thesis first studies and presents the theoretical foundation of the Shapley value calculations, including mathematical analysis and relative practical research evidence. Calculation methods, such as the Truncated Monte Carlo (TMC) Shapley algorithm, as well as the Gradient (G) Shapley and the Group Shapley algorithms are studied and analyzed, while there is special focus on studying and demonstrating how Data Shapley values could be calculated with already existing algorithms. At the same time, sensitivity and noise of the applied data privacy mechanism are measured and analyzed in the context of Data Shapley metrics. This thesis proposes how new approaches guarantee the correctness and accuracy of calculations without any information leakage from the data set under consideration. Summarizing, in the context of the current work, the following main topics are studied:

- Definition of a measure suitable to track the fair value of each (x_i, y_i) of a calculation algorithm (CA) in respect to a defined performance metric V.
- Efficient calculation of the above fair data value.
- Study of sensitivity and noise of applied data privacy mechanisms and how those could be applied in the context of Data Shapley metrics, by guaranteeing the correctness and accuracy of calculations without any information leakage.

1.3 Contribution

This work aims to contribute to the following areas:

- Provision of an efficient formulation of the important problem of calculating the fair value of data sets, known as Data Shapley valuation problem, by taking advantage of machine learning techniques.
- Review of existing empirical studies that prove the efficiencies and all desirable properties of the Data Shapley valuation method.
- Introduction of a Data Shapley calculation algorithm variations that are suitable to address the problem under study.

- Application of sensitivity and noise to ensure data privacy mechanism as well as to measure and analyze data Shapley value in the context of differential privacy.
- Implementation of a representative use case where data Shapley value calculations, combined with differential privacy guarantees are applied in selected data sets.

1.4 Main Terminology

As already stated, in this thesis the Data Shapley valuation problem and the differential privacy topic are studied and tested in a combined way. The Shapley value was defined by Shapley (1953) and can be simply defined as the level of importance, or fair value of a given data set. The main components of a data valuation process are: (1) the underlying training data set, i.e. a set of n points, each one of which is also known as datum; (2) the calculation algorithm (CA), i.e. a procedure which takes as input an arbitrary training data set and produces a calculated predictor as an output; (3) a metric that defines performance, i.e. a metric that tests the performance of the CA on a specific metric (Ghorbani & Zou, 2019). One of the most popular methods to calculate the importance of a data point, is the Leave One Out (LOO) method (Cook, 1977), which is used in this thesis for comparing and evaluating the efficiency of some of the latest and best performing valuation methods, such as the Truncated Monte Carlo Shapley (TMC-S) and the Gradient Shapley (G-S) (Ghorbani & Zou, 2019).

Differential Privacy (DP) could be defined as the change in private information that can be derived from a data set, when a point that corresponds to an entity (e.g. individual) is removed from the particular data set. The following two important metrics are of particular importance in any DP algorithm (Dwork & Roth, 2014): (1) the privacy loss or privacy budget (denoted by ε), which counts the amount of privacy loss when a differential change takes place in data, e.g. when one entry is added to, or removed from, the original data set, and (2) the accuracy, which defines the level of closeness of a DP algorithm's output to the actual output and is used as a metric for evaluating a DP algorithm. Introduction of noise is of key importance in the DP algorithms and the way it is introduced defines significantly how the DP mechanism works; there are three main noise types usually applied in DP algorithms, namely the Laplace, the exponential, and the Gaussian noise types. Additionally, the local sensitivity metric is important for the performance of DP algorithms; it is a mechanism to reduce the noise, with the trade-off of increasing the risk of

reducing the protection of the information included in the dataset. Last, but equally important, the sequence of processing applied to transform a data set into a differential private data set before releasing it for access and use, is often known data release strategy (Zhang & Meng, 2014; Wang et al., 2015).

1.5 Dissertation Structure

This dissertation is organized as follows. The current Chapter states the research problem and its importance, outlines the dissertation aims and objectives, the contribution of the current research and the main terminology used. In Chapter 2, an extensive literature review is presented. Specifically, section 2.1 introduction the problem under research, section 2.2 provides the needed background and fundamentals, including definition and mathematical formulation of the Data Shapley problem, while in section 2.3 the Shapley value calculation methods under research are presented. Section 2.4 introduces Differential Privacy (DP) and the DP noise, while it also describes data processing models, data release strategies and data release types. Section 2.5 provides some necessary background on the combination of Shapley value calculations with differential privacy, introducing concepts such as sensitivity, noise, loss function and boundaries. In section 2.6, the topic is summarized, while challenges and areas of optimization are discussed.

Chapter 3 proposes the implementation of a use case to demonstrate the results of the current research. Specifically, section 3.1 introduces the Chapter, and section 3.2 presents the proposed design, including a description of the overall solution along with the used programming platform (Python). Section 3.3 describes the implementation of Differential Privacy with Laplacian noise, while in section 3.4 the core Data Shapley implementation is presented. The Chapter closes with the core implementation results and discussion (section 3.5). Finally, Chapter 4 includes the conclusions of the current research. In more detail, section 4.1 contains a summary and the main conclusions of the dissertation, section 4.2 outlines some of the dissertation limitations, and section 4.3 proposes a path of important future work.

2 CURRENT RESEARCH – LITERATURE REVIEW

2.1 Introduction

The current thesis studies and presents the theoretical foundation of the Shapley value calculations, including mathematical analysis and relative practical research evidence. Calculation methods, such as the Truncated Monte Carlo Shapley algorithm, as well as the gradient Shapley and the group Shapley algorithms are studied and analyzed. Specifically, the current work focuses on studying and demonstrating how Sharpley values could be calculated with already existing algorithms. It proposes how new approaches that combine Shapley value calculation algorithms with differential privacy algorithms could be applied. The proposed new approaches guarantee the correctness and accuracy of calculations without any information leakage from the data set under consideration. Sensitivity and noise of the applied data privacy mechanism are also measured and analyzed.

2.2 Background and fundamentals

2.2.1 Definition and mathematical formulation of the Data Shapley problem

The data generated by individuals is increasingly becoming a significant factor of the marketplace, as labor or capital factors (Posner & Weyl, 2018). Even regulatory directives such as GDPR consider individual data as personal property and state that individuals should be compensated for the provision of this data (GDPR, 2018). In this context, a fundamental question that should be responded is, how the fair value of individual data can be calculated. Various methods have been proposed to calculate this fair value; the current work focuses on exploitation of supervised machine learning methods.

The main components of a data valuation process are the training data set, the calculation algorithm itself, and a metric that defines performance (Ghorbani & Zou, 2019). The three components are mathematically described in detail below:

1. Training Data Set: a set of n points, x_i, y_i, where $i = 1, \dots, n$.

2. Calculation Algorithm (CA): a procedure which takes as input an arbitrary training data set and produces a calculated predictor as an output. An example of a CA is a risk minimization algorithm that calculates $\theta^* = arg\ min_\theta \sum l(f(x_i;\theta), y_i)$, where l is the loss function, θ is a parameter that defines a family of models and $f(;\theta^*)$ is the predictor function.

3. Performance Metric: For any prediction function *f*, a performance metric $V(f)$ can be defined to test the performance of *f* on a specific metric.

In the context of the current work, two main topics related to fair data valuation are studied:

- The definition of a measure that is suitable to track the fair value of each (x_i, y_i) for a Calculation Algorithm (CA) in respect to a defined performance metric *V*.
- The efficient calculation of the above fair data point value.

An indicative CA could be a machine learning model, that is capable to classify for example customers or patients based on a relevant data set, using prediction accuracy as a performance metric for the algorithm effectiveness. As already mentioned, quantification of the value that each customer's or patient's data contributes to the model performance, is the main objective of the current work. It is important to state that no universal definition of the value for the overall data set is provided; on the contrary, for every data point (datum), a separate value is provided depending on the CA, the performance metric, and the overall data set used. This happens because there are data points which are more significant than others in many cases, depending on the calculation algorithm used, e.g. logistic regression, neural networks, or similar.

The Shapley value was defined by Shapley (1953) in game theory related content; since then, it has been widely adopted and used in theoretical and applied research across various research fields (Shapley et al., 1988). Many problem types use Data Shapley as a basis for their foundation and analysis. Examples are resource allocation related problems, and voting application studies (Milnor & Shapley, 1978; Gul, 1989). A big family of Data Shapley value models have been proposed in recent years, focusing on feature scoring in predictive algorithms. In those studies, the main objective is quantification of features in order to decide which ones have the highest impact on the outputs of a model (Cohen et al., 2007; Strumbelj & Kononenko, 2010; Datta et al., 2016; Lundberg & Lee, 2017; Lundberg et al., 2018; Chen et al., 2018). Other very important studies

related to the Data Shapley value calculation takes advantage of Monte Carlo simulation methods or similar network algorithms, linear regression methods, or in some specific cases analytical solutions (Cook, 1977; Fatima et al., 2008; Castro et al., 2009; Maleki et al., 2013; Michalak et al., 2013). Lately, Ghorbani & Zou (2019) have proposed use of Data Shapley value in the framework of machine learning methods.

In similar directions Cook and Weisberg (1982) and Koh and Liang (2017) studied the leverage or influence that perturbations of data points have to algorithm's parameters and outputs. However, for those studies it has been shown that the proposed algorithms are not always efficient and might present stability issues, while they do not present some of the desirable properties that Data Shaplay valuation methods should have (Ghorbani et al., 2017). Other studies, focus on the financial aspects, from an angle that is of particular interest for policy makers and economists; in those studies, it is assessed how individuals or citizens should be compensated for the data they provide, while at the same time it is discussed how individuals should be incentivized to generate valuable data (Arrieta Ibarra et al., 2017; Posner & Weyl, 2018).

One of the most popular methods to calculate the importance of a data point, is the Leave One Out (LOO) method. The core functionality of LOO is based on performing a comparison between predictor's performance when the model is trained with the full data set, versus performance when it is trained with a subset of the full set which is smaller by just one point (Cook, 1977). The level of degradation of model's performance when it is applied to the subset of the original training set, is a measure of the value of the removed data point. Variations of the LOO, use the notion of a leverage or influence score, which tracks the change of predictor's level of performance when the weight of a data point slightly changes (Cook & Weisberg, 1982). As shown by Ghorbani and Zou (2019), the LOO method does not satisfy important desirable properties for implementing fair data calculation algorithms.

In what follows, application of Machine Learning techniques to calculate the fair value of data is extensively studied from both a theoretical and a practical perspective. Given a fixed training data set $D = \{(x_i, y_i)\}_1^n$, where each y_i is the categorical or actual value that corresponds to x_i, and a machine learning algorithm $\mathcal{A}$ that uses D as an input to produce prediction, it is of particular interest to calculate the predictor when training is applied on particular subsets S of D, i.e. $S \subseteq D$. If $V(S; \mathcal{A})$, or for simplicity $V(S)$, denotes the performance score for an algorithm $\mathcal{A}$

and a predictor based on the data set *S*, then the main objective is to calculate the fair value of the data point i, which is denoted as $\phi_i(D, \mathcal{A}, V) \in \mathbb{R}$, or symbolized as $\phi_i(V)$, or ϕ_i for simplicity. According to Ghorbani and Zou (2019), a well-defined fair value function ϕ should posses the following important properties:

(1) If for every $S \subseteq D - \{i\}$, $V(S) = V(S \cup \{i\})$ holds, then ϕ_i=0, which means that the value of any (x_i, y_i) that does not have any impact on the performance when added to any training subset, should be equal to zero.

(2) If for the data points *i* and *j* and any $S \subseteq D - \{i, j\}$, $V(S \cup \{i\}) = V(S \cup \{j\})$ holds, then $\phi_i = \phi_j$. This practically means that if the addition of data points *i* and *j* to any subset of the training set produces the same variation in the score of the predictor then they must be given the same value.

(3) Let $V = -\sum_k l_k$, where $k \in test\ set$ and l_k is the loss function of the predictor on the *k*-th data point of the test set. Also, let $V_k = -l_k$ be the performance of the predictor on the *k*-th data point of the test set, and $\phi_i(V_k)$ be the value of the *i*-th point to the *k*-th data point of the train set. If the data point *i* contributes to the predictors of test points 1 and 2 respectively values $\phi_i(V_1)$ and $\phi_i(V_2)$, then the value of *i* in the predictions of both test points is equal to $\phi_i(V_1) + \phi_i(V_2)$. This practically means that, when the overall prediction is the sum of *K* separate predictions, then the value of a data point equals the sum of its value for each prediction, i.e. for performance scores *V* and *W*, $\phi_i(V + W) = \phi_i(V) + \phi_i(W)$.

Ghorbani and Zou (2019) have proven that any calculated fair value $\phi(D, \mathcal{A}, V)$ for which the above properties (1)-(3) hold, can be mathematically described in the following form:

$$\phi_i = C \sum_{S \subseteq D - \{i\}} \frac{V(S \cup \{i\}) - V(S)}{\binom{n-1}{|S|}} \qquad \text{(eq. 2.1)}$$

where C is a constant and the above sum aggregates all subsets of D that do not contain i.

The value ϕ_i calculated with the above relation is known as the data Shapley value of point i. The above equation practically aggregates the weighted contributions of i, with the weight being the inverse of the number of subsets with size $|S|$ in the $D - \{i\}$ universe. This formulation is very close to the LOO formula where every point's contribution to the fair value is based on a random

subset of the training set, instead of the whole training set. In other words, Shapley equation captures all scenarios of potential subsets instead of random selections.

Equation (2.1) calculates and assigns to all data points a fair value. However, the computational effort required to calculate every marginal contribution is very high and heavily impacted by the size of the training data set, while for every $S \subseteq D$ the calculation of $V(S)$ also implies that a predictor on S should be computed with the aid of algorithm $\mathcal{A}$. Consequently, calculation of the accurate Data Shapley value is not realistic for actual use cases that include large data sets. For this reason, approximations are used in practice.

2.3 Shapley value calculation methods

In this section, popular approximation methods commonly used to calculate the Data Shapley value are presented and discussed. Specifically, the Truncated Monte Carlo Shapley algorithm and the Gradient Shapley algorithm. The above methods will be used as a basis for the implementation proposed in this study.

2.3.1 Truncated Monte Carlo Shapley Algorithm

According to Ghorbani and Zou (2019), equation (2.1) can be reformulated if C is set equal to $\frac{1}{n!}$. Specifically, if Π is a uniform distribution function in the universe of all $n!$ data point permutations, then the following equation holds:

$$\phi_i = \mathrm{E}_{\pi \approx \Pi}\left[V\left(S_\pi^i \cup \{i\}\right) - V(S_\pi^i)\right] \qquad \text{(eq. 2.2)}$$

with S_π^i being the data set that includes only the data points that are positioned before data point i in permutation π. Obviously, $S_\pi^i = \emptyset$ when i denotes the first data point of the data set. The above notation implies that equation (2.2) in fact formulates and represents an expectation computation problem.

Variations of the Monte-Carlo method have been introduced and studied in the framework of the Data Shapley Value problem (Mann & Shapley, 1962; Castro et al., 2009; Maleki et al., 2013). In principle the steps followed in those variations are the below:

1. *A sample of random permutations of the data points is selected.*
2. *For each permutation the marginal contribution of each additional data point is calculated.*
3. *The overall estimation is calculated by averaging all the marginal contributions produced by step (2).*

The above steps conclude to an estimation of the Data Shapley value which is generally unbiased. Following such an approach, the Monte Carlo calculations are iteratively repeated until the algorithm converges. According to Maleki et al. (2013), convergence is achieved by generating $3n$ samples, which simply means that the computational complexity is $O(n)$.

To reduce computational complexity and optimize algorithm performance, Ghorbani and Zou (2019) introduced the idea of truncation in the above Monte Carlo iterative approach. The method the concluded with, was named Truncated Monte Carlo Shapley (TMC-Shapley). The idea behind what they proposed is that when a sample permutation is processed to compute its marginal contributions, calculations can be truncated as soon as $V(S)$ reaches a point with an acceptable performance tolerance $V(D)$. Calculations on remaining data points for the specific permutation could be then ignored (i.e., marginal contribution can be set equal to zero for those points), without any significant impact on the produced outcomes. Applying the idea of truncation in Data Shapley Monte Carlo saves significant computational effort and at the same time does introduce only immaterial bias into the estimation produced.

The above calculation is feasible to implement in the framework of Machine Learning methods. Validity of the idea can be explained by the fact that, since the data set used for testing is finite, $V(S)$ is in fact just an estimation of the actual performance of the trained model. It is thus adequate to approximate the Data Shapley value by taking into consideration a tolerance that equals the intrinsic noise; this noise could be measured by quantifying the performance variations for the same predictor across the samples of the testing data set (Hastie et al., 2001). At the same time, Mahajan et al. (2018) and Beleites et al. (2013) have proven that when S becomes larger, further addition of training points gradually has smaller impact on the performance of the calculation.

2.3.2 Gradient Shapley Algorithm

Even by applying the efficiencies stated in the previous section, Data Shapley calculations remain still expensive, especially when big data sets are considered, or high complexity predictive models such as deep learning models are used. For this reason, the Gradient Shapley Algorithm has been proposed by Ghorbani and Zou (2019). In most of the cases, predictive models require that algorithm A performs random iterations on batches of D to appropriately adjust the model parameters. These iterations are the basis of the stochastic Gradient Shapley, or G-Shapley algorithm. In this case, the model is gradually trained, by iteratively considering simple approximations of the model produced by a single parse of the training data. Obviously, this approach is quite close to the one followed in the Monte Carlo Shapley algorithm. The main difference is that, in each sample permutation, the model is updated by applying a gradient descent on each data point for every time step. The marginal contribution can then be calculated as the change that takes place in the model performance in every iteration. Additionally, to produce the best approximation, Ghorbani and Zou (2019) applied a hyper-parameter search during the learning process, which resulted into the best performing model when this is trained with only one parse of the data.

2.3.3 Variations

In some cases, especially when very large data sets are used as an input, calculation of the Data Shapley values become extremely demanding in terms of computational resources, and algorithm's performance significantly declines. An approach usually proposed in such cases is the grouping of data points and application of the Data Shapley algorithms on those groups instead of individual datums. Representative examples that take advantage of the data grouping approach can be found in the health industry applications such as the prediction of heart diseases, breast cancer, skin cancer etc., where patients could be grouped into separate bins using as grouping criteria features like gender, age, etc. Other examples could be found in relative research efforts (Ghorbani & Zou, 2019).

2.4 Differential privacy

During the last two decades, emblematic data breaches have led private and public Organizations to put high in their priority agendas, data privacy initiatives. In exactly the opposite direction, the increasing usage of Machine Learning algorithms has generated strong demand on large volumes of data sets to get trained and be executed to produce valuable information insights. Important portions of the required data may contain personal, sensitive and in any case private information; potential disclosure of such data could generate unpredictable and possibly unmanageable consequences for the organizations that are responsible to collect, store and manage it.

Differential Privacy (DP) is a relatively new research field, that aims to respond to information privacy challenges and enable Organizations and Companies to effectively manage private and sensitive information included in the managed data, as can be seen in Figure 2.1. DP was established as a research domain in 2006 by the seminal work of Dwork (2006), but has become popular during the last years, among others because it provides the so-called privacy guarantees as part of security frameworks and their implementation.

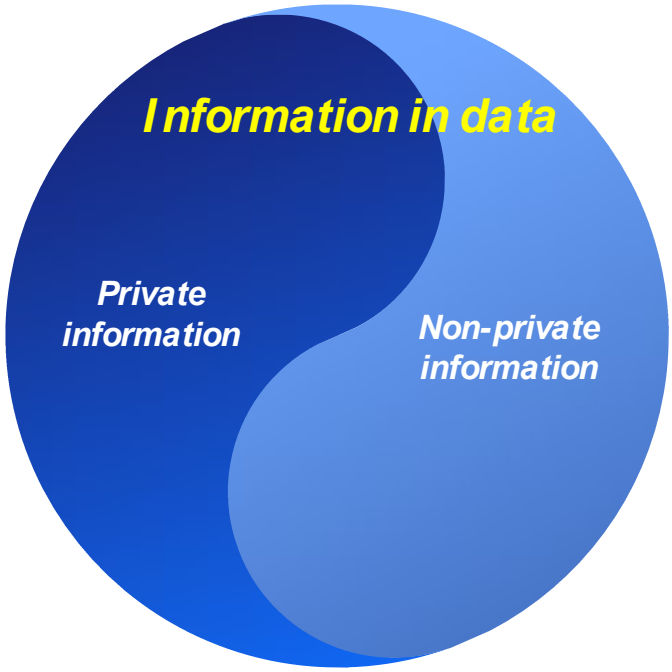

Figure 2.1: Information from a Differential Privacy perspective.

DP could be viewed as an approach to mathematically define information privacy in the context of machine learning and statistical analysis techniques (Nissim et al., 2018). It can be simply defined as the change in private information that can be derived from a data set, when a point that corresponds to an entity (e.g individual) is removed from the particular data set. DP carries the unique property of guaranteeing that the outcome of a calculation will remain essentially unchanged when a particular entity's private information, contained in the original input is removed (Nissim et al., 2018). This unique property can guarantee information privacy protection against a variety of privacy attacks such as differencing, linkage and reconstruction attacks (Dwork & Roth, 2014).

The following two important metrics are of particular importance in any DP algorithm (Dwork & Roth, 2014):

- Privacy loss or privacy budget (denoted by ε): This is a metric of privacy loss when a differential change takes place in data, e.g. when one entry is added to, or removed from, the original data set. Information privacy protection is inversely propotional to the size of ε.
- Accuracy: Given a data set, accuracy refers to the level of closeness of a DP algorithm's output to the actual output and is used as a metric for evaluating a DP algorithm.

From a mathematical perspective, the DP problem can be formulated as follows. A calculation function (or algorithm) K provides ε-differential privacy when for any data sets D and D' that vary by one row at most (i.e $d(D, D')$=1), and for any $S \subseteq Range(K)$, the following relation holds:

$$Pr[K(D) \in S] \leq e^{\varepsilon} Pr[K(D') \in S] \quad \text{(eq. 2.3)}$$

It is important to note that decreasing ε leads to declining accuracy. Moreover, an algorithm that is of 0-differential privacy, even if it protects the information privacy adequately, it will be of limited usage if it has very low accuracy, since it would produce only noise as an output. From another angle, having an ε which is equal to 0, makes the DP algorithm being independent of the data set used as an input, and consequently protects information privacy in a perfectly. Typical values for ε are thus small but non-zero, e.g 0.5, 0.1, 0.001.

Based on eq. (2.3) and the above discussion, it can easily be understood that the stronger the privacy guarantee provided the greater the noise added to the outcome of a DP algorithm. To

mathematically justify the trade off between privacy guarantee and noise, the (ε, δ)-differential privacy scheme was introduced in Dwork et al. (2006) and Dwork (2011a). Given two data sets D and D' that vary by one row at most (i.e $d(D, D')$=1), and a calculation function (or algorithm) K, $Range(K)$ being the set of all possible output alternatives of K, then for any $S \subseteq Range(K)$, the (ε, δ)-differential privacy could be defined as follows:

$$Pr[K(D) \in S] \leq e^{\varepsilon} Pr[K(D') \in S] + \delta \qquad \text{(eq. 2.4)}$$

There is not any hard requirement to select any specific values for ε or δ, which practically means that they could be selected based on the requirements or specific properties of the problem that needs to be tackled each time. In general, appropriate values for δ are usually smaller than 10^{-4}, while (ε, δ)-differential privacy degenerates to ε-differential privacy if δ is equal to zero.

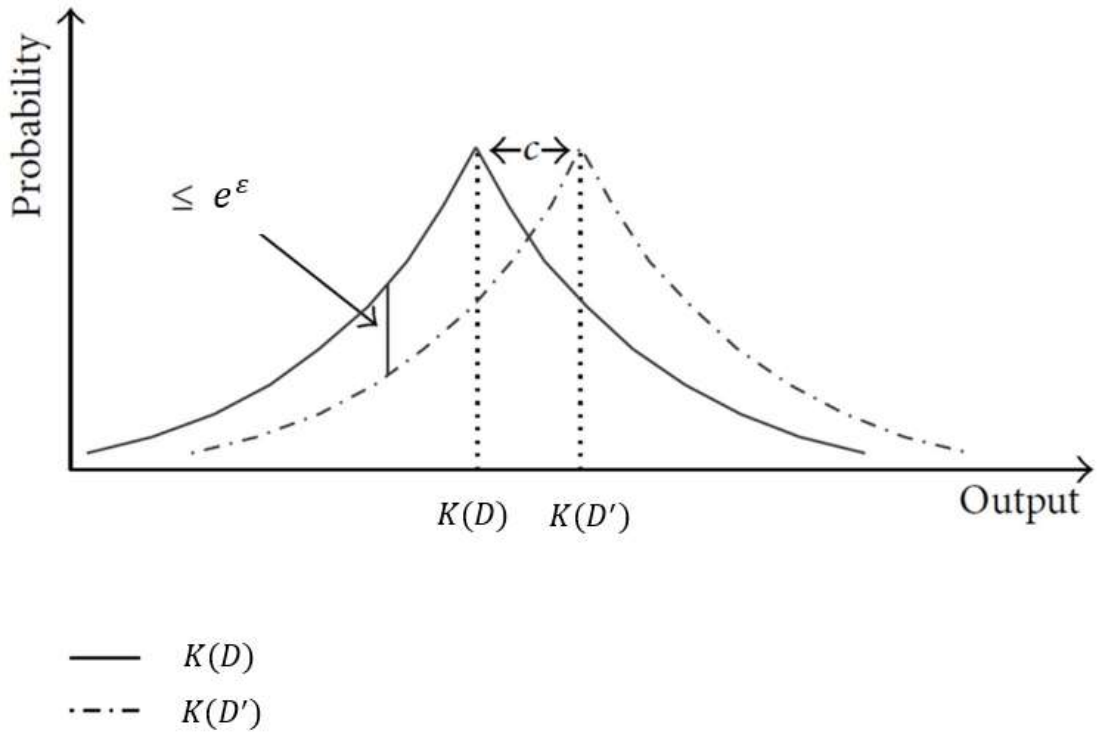

Figure 2.2: Probability distributions for the outputs of the function (or algorithm) K for datasets D and D'.

(Source: Wang et al., 2015)

According to Dwork and Roth (2014), DP has the following four (4) important properties that make it feasible to analyze personal or sensitive information taking into consideration protection of information privacy:

1. Ability to quantify information privacy loss. DP makes it feasible to measure information privacy loss and produce meaningful comparisons in the context of machine learning techniques.
2. Information privacy loss composition. Quantification of information privacy loss, makes it feasible to analyze and control cumulative impact of privacy loss when multiple computations are gradually applied. This quantification facilitates the proper design and analysis of DP algorithms when they are composed from smaller building blocks.
3. Immunity on post-processing. DP is not affected by any type of post-processing. This practically means that outputs of differential privacy algorithms cannot be used to "re-create" the relevant inputs, by using any type of computation.
4. Information privacy loss on groups of individuals. DP makes it feasible to analyze privacy loss incurred in groups of individuals e.g. families, when data are provided at individual level.

Especially for property (2) above, there are two separate types of composition that could be met, sequential and parallel. Both play a significant role in providing evidence on whether a specific algorithm satisfies DP or not. On top, they could also be used to quantify the differential privacy loss amount, providing thus significant input on the budgeting strategy effectiveness. The sequential composition variation shows that both the differential privacy budget and the error are evolving in a cumulative linearly manner when multiple differential privacy approach is used to release data for a given dataset. The parallel composition variation shows that the level of differential privacy guarantee is decreasing (or in other words the security is decreasing) when ε_i grows (Li et al., 2012; McSherry, 2010). To balance the required budget with the realized security, a number of methods have been developed based on the calculation of optimal values for the absolute or the relative error (Qardaji et al., 2013), the variance or the standard deviation (Cormode et al., 2012; Xiao et al., 2014; Qardaji et al., 2013), and the false negatives (Lee & Clifton, 2014).

In the rest of the section, the most important differential privacy concepts are studied and critically analyzed, starting with the definition and detailed presentation and mathematical formulation of the DP noise.

2.4.1 Differential Privacy noise

As already explained, introduction of noise is of key importance in the DP algorithms and the way it is introduced defines significantly how the DP mechanism works. There are three main noise types usually applied in DP algorithms, namely the Laplace, the exponential, and the Gaussian noise types. To better understand how the noise mechanism works, the global sensitivity metric should be introduced. For a function $f: D \rightarrow R^d$, the *global sensitivity* is defined as (Dwork, 2006):

$$\Delta f = \max_{D,D'} \|f(D) - f(D')\|_1. \quad \text{(eq. 2.6)}$$

where D and D' are two adjustment data sets, d is the data set dimension and $\|f(D) - f(D')\|_1$ the first order norm of the distance between $f(D)$ and $f(D')$.

Depending on the set of operations performed by function *f*, the global sensitivity metric could take different values, e.g, if f represents a simple count function, then $\Delta f = 1$, while Δf normally takes larger values for other, more complex, operations. Nissim et al. (2007), proposed a mechanism to better manage outputs when the noise increases. Specifically, they introduced the metric of local sensitivity defined as follows. For a function $f: D \rightarrow R^d$, the *local sensitivity* is defined as:

$$LS_f = \max_{D'} \|f(D) - f(D')\|_1 \quad \text{(eq. 2.7)}$$

where D and D' are two adjustment data sets, d is the data set dimension and $\|f(D) - f(D')\|_1$ the first order norm of the distance between $f(D)$ and $f(D')$.

The local sensitivity metric normally reduces the noise, but this comes with an increasing risk of reducing the protection of the information included in the specific dataset. For this reason, Nissim et al. introduced the β-smooth sensitivity metric, that makes it feasible to add noise proportionally to a smooth upper boundary of the local sensitivity. The β-smooth sensitivity is defined as follows (Nissim et al., 2007). For a function $f: D \rightarrow R^d$ and a $\beta > 0$, the *β-smooth sensitivity* is given by:

$$S_{f,\beta}(D) = \max_{D'}\left(LS_f(D')e^{-\beta d(D,D')}\right) \quad \text{(eq. 2.8)}$$

where D and D' are two adjustment data sets, d is the data set dimension and LS_f the local sensitivity of f.

In what follows the variations of DP noise are presented in detail.

2.4.1.1 Laplacian DP noise

The Laplacian DP noise has been proposed by Dwork et al. (2006b) as a proper mechanism for achieving differential privacy. The way this mechanism works is that it adds noise to the DP algorithm outputs, which follows Laplacian distribution. Using mathematical formulation, for a data set D, a function $f: D \rightarrow R^d$ with global sensitivity Δf, and a calculation algorithm $K(D) = f(D) + n$ that satisfies ε-differential privacy, the noise n follows the Laplacian distribution, i.e $n \sim Lap\,(\frac{\Delta f}{\varepsilon})$ with location and scale parameters respectively equal to 0 and $\frac{\Delta f}{\varepsilon}$. Let $Lap\,(b)$ denote the Laplacian distribution with location and scale parameters respectively equal to 0 and b, and $p(x) = \frac{e^{-\frac{|x|}{b}}}{2b}$ denote its probability density function. If $\sigma(x)$ is the standard deviation and $D(x)$ the variance, and $b = \frac{\Delta f}{\varepsilon}$, then:

$$D(x) = 2b^2 = 2\frac{(\Delta f)^2}{\varepsilon^2} \quad \text{(eq. 2.9a)}$$

$$\sigma(x) = \sqrt{D(x)} = \sqrt{2}\frac{\Delta f}{\varepsilon} \quad \text{(eq. 2.9b)}$$

As it is depicted in Figure 2.3, for larger values of noise, the larger the value of b and the value of ε (Wang et al., 2015).

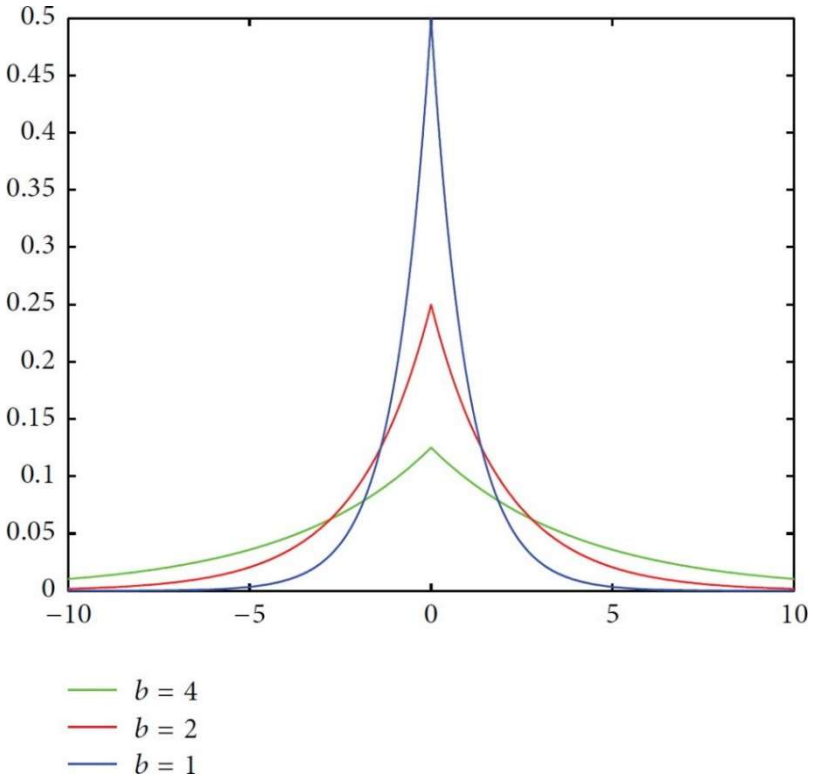

Figure 2.3: Visualization of the probability density function for different calls of b.
(Source: Wang et al., 2015)

2.4.1.2 Exponential DP noise

The exponential DP noise has been proposed by McSherry & Talwar (2007) as another mechanism for achieving differential privacy. In similarity with the Laplacian DP noise, the way this mechanism works is that it adds noise to the DP algorithm outputs, which follows exponential distribution. Their main differentiation point is that the Laplacian mechanism is usually being applied when the DP problem outputs are numerical, while the exponential mechanism is applied when the outputs are not numerical. Using mathematical formulation, for an input data set denoted as D, an output denoted as $r \in R$, and a score function $v: D \times R \to R$, if a calculation algorithm $K(D, v)$ decides a potential answer based on the following exponential probability defined in (eq. 2.10), then algorithm K satisfies ε-differential privacy:

$$K(D, v) = \left\{ r: Pr[\, r \in R] \propto e^{\frac{\varepsilon\, v(D,r)}{2\Delta v}} \right\} \qquad \text{(eq. 2.10)}$$

where Δv denotes the sensitivity of the score function v, defined as follows:

$$\Delta v = \max_{r \in R} \max_{\|D \Delta D'\|=1} |v(D,r) - v(D',r)| \quad \text{(eq. 2.11)}$$

As already mentioned, the exponential DP noise mechanism can be used for non-numerical results, using values produced by the score function. The highest the score of an output, the higher probability to be outputted when ε is larger. Additionally, when the difference between the output probabilities increases, the produced security decreases; for smaller values of ε, the produced security increases.

2.4.1.3 Gaussian DP noise

The Gaussian DP noise is another mechanism for achieving differential privacy. The way this mechanism works is that it adds noise to the DP algorithm outputs, which follows Gaussian distribution. However, this mechanism requires a slightly different notion of sensitivity. For a function $f: D \rightarrow R^d$, the l_2 *sensitivity* is defined as:

$$\Delta f_2 = \max_{D,D'} \|f(D) - f(D')\|_2 \quad \text{(eq. 2.12)}$$

where D and D' are two adjustment data sets.

The l_2 sensitivity and l_1 norms enjoy the following relationship:

$$\|D\|_2 \leq \|D\|_1 \leq \sqrt{d}\|D\|_2 \quad \text{(eq. 2.13)}$$

for a vector $D \in R^d$. Thus, the l_2 sensitivity might be greater by a factor of $\sqrt{d}$ than l_1 sensitivity.

The univariate Gaussian distribution $N(\mu, \sigma^2)$ with mean and variance parameters respectively equal to μ and σ^2 has probability density function which is the following:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{(eq. 2.14)}$$

2.4.2 Data processing models and data release strategies

Two different data processing models exist in the framework of differential privacy, i.e the interactive or on-line query model (see Figure 2.4) in which the data requester can access the data through an interface provided by the owner of the data, and the non-interactive model or offline query model (see Figure 2.5) in which the data requester can directly access only sanitized data sets as they are released by the data owner (Dwork et al., 2006; Xiong et al., 2014).

In the interactive model (see Figure 2.4) the data owner provides a DP-based query algorithm to the data requester, while the data requester requests the needed data using a query. Upon receipt of the query request, the algorithm brings the requested raw data from the original data source and performs sanitization before returning it to the requester. In this case, the number of performed queries is inversely proportional to the privacy budget ε, which means that the bigger the number of queries the smaller the budget for each query and the larger the noise added to the query result. Consequently, in this type of model, it is of key importance to design the query algorithm in such a way, so that it provides the largest number of queries that could be applied under the restriction of the budget ε (Wang et al., 2015).

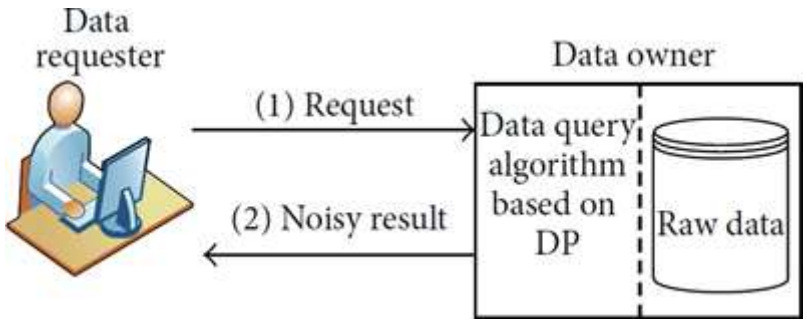

Figure 2.4: Interactive or on-line query model.

(Source: Wang et al., 2015)

In the non-interactive model (see Figure 2.5) the data owner releases a sanitized data set since he/she is considered a trusted curator. Upon receipt of a query request from the data requester, the sanitized data set is used to formulate and return the noisy result. In this type of model, it is of key importance to design the query algorithm in such a way, so that it is capable to enhance the accuracy and efficiency of the query; even if a high number of queries are required this is not prohibitive, since the number of queries are not related to the privacy budget ε (Wang et al., 2015).

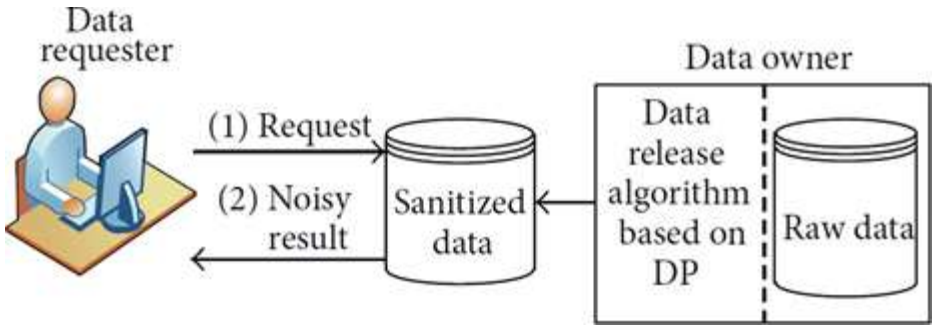

Figure 2.5: Non-Interactive model or offline query model.

(Source: Wang et al., 2015)

Based on the above models, three different data release strategies can be applied (Zhang & Meng, 2014; Wang et al., 2015):

1. **Data Release Strategy 1:** In this alternative, ε represents a uniform privacy budget and the noise that follows Laplace distribution $Lap\ (\frac{\Delta f}{\varepsilon})$ is added to the original raw data $D = \{x_1, x_2, \dots, x_n\}$, giving a noisy transformation D_{noisy}. Post-processing can be applied to the noisy transformation D_{noisy} to improve query accuracy, with a method such as least square, and finally release the transformed post-processed data set $D_{released}$.

2. **Data Release Strategy 2:** In this alternative, ε represents again a uniform privacy budget. First a processing is applied to the original raw data $D = \{x_1, x_2, \dots, x_n\}$, incuding transformation (e.g a graph structure is converted to a tree structure) and compression, to reduce the sensitivity of function f, Δf. The outcome is a synthetic data set $D_{synthetic}$ in which then noise that follows Laplace distribution $Lap\ (\frac{\Delta f}{\varepsilon})$ is added, to finally release the data set $D_{released}$.

3. **Data Release Strategy 3:** In this alternative, ε represents a non-uniform privacy budget, i.e $\varepsilon_\iota \neq \varepsilon_j$ and the noise that follows Laplace distribution $Lap\ (\frac{\Delta f}{\varepsilon})$ is added to the original raw data $D = \{x_1, x_2, \dots, x_n\}$, giving a noisy transformation D_{noisy}. Finally, reasonable budgeting strategy techniques are used on the noisy transformation D_{noisy} to improve query accuracy and release the data set $D_{released}$.

The above data release strategies can be applied either autonomously or in combination. For example, Data Release Strategy 3 could be initially used to allocate a reasonable budget and then post-processing of Data Release Strategy 1 could be applied to improve the query accuracy.

2.4.3 Data Release Types

Ensuring high levels of privacy guarantee and at the same time providing highly usable released data, is apparently the main objective of any differential privacy approach. For this reason, a significant number of data release methods have been proposed, defining different types of differential privacy. Although the research in the DP domain is still at the beginning, the data release methods applied using the strategies mentioned above, lead to different DP approaches. A classification of these approaches in five (5) separate types, namely *Histograms*, *Tree structures*, *Time series*, *Graphs*, and *Pattern mining*, has been proposed by Wang et al. (2015). The advantages and defects of each one of those types, along with the most important representative methods and some typical applications are summarized in Table 2.1.

Table 2.1: Main types of DP data release methods.

Classification	Advantage	Defect	Representative methods	Typical Applications
Histogram	Supports any range query	The noise is large in high-dimensional data	DPCube NoiseFirst Structure First	Statistical analysis of disease and search history
Tree structure	Supports multiple-dimensional data and data-dependent or data-independent query	The noise is large in high-dimensional data	Quad-opt AG SEA	Location query of user and device, transport planning
Time series	Supports time series query	It is hard to balance the utility and security in high-dimensional data	TEA FAST U-KF	Real time traffic, disease surveillance
Graph	Supports the analysis and query of graph data	Sensitivity is high, and node-differential privacy is difficult to achieve	DP2K(ε) LNPP hrg-ε_1-e-ε_2	Relationship analysis of user in health social network
Pattern mining	Supports differential privacy pattern mining, and original data can be constructed by frequent patterns	Long pattern leads to large noise	PrivBasis NoiseCut Diff-FPM	User behavior, DNA sequences, trajectory and disease trend analysis, recommended system

In the following sections, each one of the above five (5) classifications, is described in detail and analyzed.

2.4.3.1 Histogram release type

The histogram release type is based on histograms created from the input data set by splitting it into disjoint data subsets, which are known as "*buckets*" or "*bins*". Bucketing is based on a given set of rules or attributes. To access the raw data, a DP interface is commonly used, through which users send queries on the data and the histograms are used to formulate responses. The process workflow is depicted in Figure 2.6 (Xiao et al., 2010).

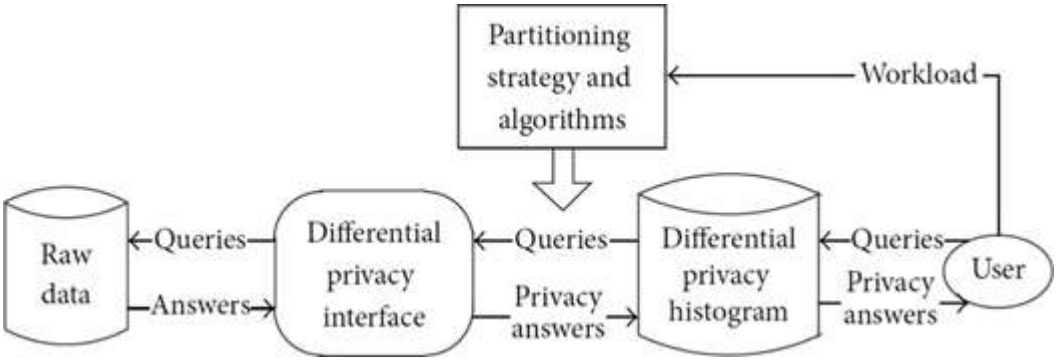

Figure 2.6: DP using the histogram data release type.

(Source: Xiao et al., 2015)

One of the most common methods to implement the histogram approach, is by adding Laplace noise equal to $Lap\ (\frac{1}{\varepsilon})$ (function sensitivity $\Delta f = 1$) to all histogram bins. This selection is appropriate when the requests are based on short range queries, since for larger ones the error increases significantly (error is proportional to $\frac{2n}{\varepsilon^2}$). To reduce the error, a grouping of the buckets can be applied; an example with grouping of age bins is demonstrated in Figure 2.7 below (Xiao et al., 2010). In this example, by merging the seven (7) buckets into three (3), the total noise is reduced in 3/7=0.43 of the initial noise. The side effect of this noise improvement is that an approximation error is introduced which implies that to keep the DP approach effective, the number of bucket mergers should be limited. Since the finer the partitioning is, the smaller the approximation error will be, but at the same time the larger the noise will be, it is of key importance to find the proper balance between introduced noise and approximation error (Xiao et al., 2010).

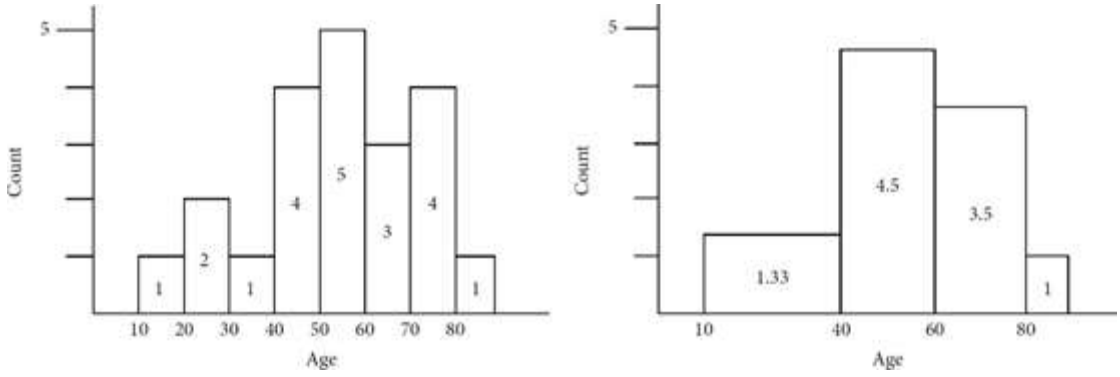

Figure 2.7: Grouping of buckets in the histogram data release type.

(Source: Xiao et al., 2015)

One of the most popular partitioning strategies was proposed by Xiao et al. (2010, 2012) and is based on the notion of the *kd*-tree. Specifically, the *kd*-tree histogram data release process is consisted of the following five (5) steps:

1. The histogram structure is constructed based on the *k*-th different attributes of the original raw data.
2. A noisy *k*-dimensional data set is generated, by adding noise to the original histogram using half of the privacy budget (i.e. $\varepsilon/2$).
3. Partitioning of the noisy *k*-dimensional data set is conducted with the use of the *kd*-tree algorithm.
4. Noise is added to each one of the partitions using the other half of the privacy budget ($\varepsilon/2$).
5. The generated noisy histogram is released.

A number of algorithms have been based on the *kd*-tree histogram data release method. Xiao et al. (2010) proposed *DPCube* which follows Data Release Strategy 1, i.e. the *kd*-tree is applied for post-processing purposes after addition of noise. The *DPCube* algorithm can handle long-range query and multidimensional data, while the query error can be optimized with the proper parameter value selection.

Xu et al. (2013) proposed the idea of using the notion of Sum of Squared Error (SSE) to appropriately balance the partitioning and noise error. Based on this notion, two important methods have been proposed, namely methods *NoiseFirst* and *StructureFirst*. *NoiseFirst* follows Data Release Strategy 1, i.e. first adds noise and then uses dynamic programming to construct the optimal histogram; *NoiseFirst* is suitable for short-range queries. *StructureFirst* follows Data Release Strategy 2, i.e. first generates the optimal histogram and then adds noise. *StructureFirst* uses privacy budget ε_1 in the first step as a privacy guarantee mechanism for the sensitive information in the histogram. In the second step, noise is added to the different partitions using the rest of the privacy budget (ε- ε_1). *StructureFirst* is suitable for long-range queries and is similar to en extend with *DPCube*. Since both *NoiseFirst* and *StructureFirst* reconstruct the histogram, large numbers of partitions negatively affect performance.

Hay et al. (2010) proposed another approach that reduces the error in long-range queries by applying a hierarchical tree structure, transforming thus the histogram into a hierarchical tree structure. By applying this transformation, all queried intervals can be organized as trees and query accuracy is enhanced (Qardaji et al., 2013b). Further work done by Qardaji et al. (2013b), based on the hierarchical tree notion, concluded to the *Flat* method which works well for larger dimensions compared to the original hierarchical approach.

A summary of all histogram data release methods is depicted in Table 2.2.

Table 2.2: Histogram data release methods.

Method	Strategy	Advantage	Defect	Budgeting
DPCube	Data Release Strategy 1	Supports multidimensional and long-range query	Parameter affects query accuracy heavily	Uniform budgeting
NoiseFirst	Data Release Strategy 1	Fit for short-range query	Histogram reconstruction is expensive	Uniform budgeting
Structure First	Data Release Strategy 2	Fit for long-range query	Histogram reconstruction is expensive	Near-optimal budgeting
Flat	Data Release Strategy 2	Fit for multidimensional data	Low-dimensional data query error is larger	Uniform budgeting

2.4.3.2 Tree structure release type

The tree structure methods are targeting to the reduction of query errors and are based on the idea of splitting the data into tree structures. They are also met in the relevant literature in another form, as private spatial decompositions where the geospatial data are split into sub-regions enabling the generation of statistics within each one of them. In case that a partition might disclose sensitive information when such a split is performed, the method is called data-dependent decomposition. If it might not, it is called data-independent decomposition. In that sense for example, *kd*-tree splits are based on the median value, which implies disclosure of the median value itself; that is why it is a data-dependent decomposition (Cormode et al., 2012).

In what follows the two main types of decomposition are presented in detail.

(1) **Data-dependent decomposition.** As depicted in the example of Figure 2.8, the node with grid coordinates (5, 4) discloses information of itself during the splitting, leading to noise addition in order to protect actual information disclosure. Given a data set $D = \{x_1, x_2, \dots, x_n\}$, $x_i \in [1, r]$ with *n* data points sorted in ascending order and x_{median} being the median value, then by adding $noise \sim Lap\left(\frac{\Delta f}{\varepsilon}\right)$ the noise median is obtained, i.e $M(D) = x_{median} + noise$. Given that $M(D)$ might take values not belonging to $[1, r]$ it has been proposed by Inan et al. (2012) that the mean value x_{mean} could be selected instead of the median when numerical data are used; the mean value could be then approximated by dividing the sum of the noisy counts to the number of the counts. Xiao et al. (2010, 2012) used initially half of the privacy budget on the original data and the rest of the budget to construct a *kd*-tree structure and compute the median on noisy data, guaranteeing thus the differential privacy of the median.

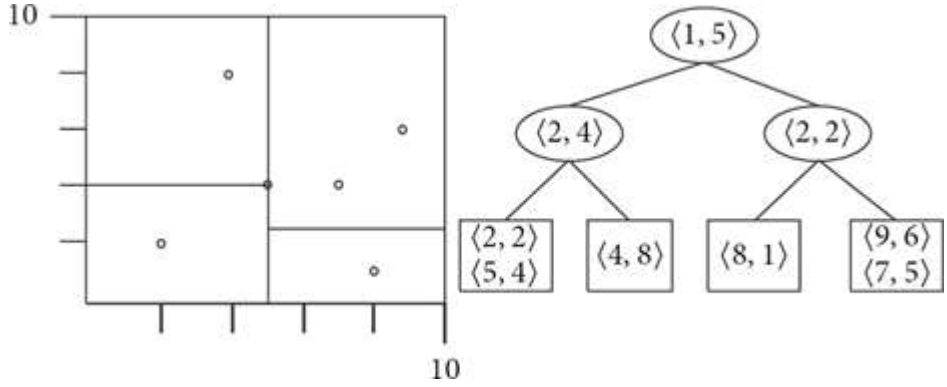

Figure 2.8: A private *kd*-tree.

(Source: Wang et al., 2015)

Cormode et al. (2012) proposed a method based on *kd*-tree structure and *EM* medians, known as *kd*-standard. This method uses *EM* (exponential mechanism) to check and validate the median by applying an exponential mechanism and calculates differential privacy as $Pr[EM(D) = x] \propto e^{-\varepsilon|rank(x)-rank(x_{median})|/2}$. In this way the rank of the returned x is the same with the rank of x in D. Since x is almost equal to x_{median}, the median is guaranteed. The method uses a privacy budget ε_{median} for calculating x_{median}, and a privacy budget ε_{count} for the computation of the count. Since $\varepsilon = \varepsilon_{median} + \varepsilon_{count}$, the larger the value of ε_{median} the more accurate the median and the larger the count error, while the smaller the value of ε_{median} the more accurate the count computation but the larger the median error. This practically means that it is of key importance in the *kd*-standard method to find the proper balance between the budget that should be allocated to the count computation and the budget that should be used for the median computation.

(2) **Data-independent decomposition.** Method Quad-opt which is based on the notion of quadtree was proposed by Cormode et al. (2012). The main mechanism on which this method is based is the recursive decomposition of the data set into equal quadrants, that do not disclosure any data information. As depicted in the example of Figure 2.9, the data set ***a*** is decomposed into four quadrants of equal size, i.e. $\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_3$ and $\boldsymbol{b}_4$; all ***b*** data sets are then decomposed to four (4) equal quadrants and this continues until a predefined depth ***h*** of the scetched tree is met.

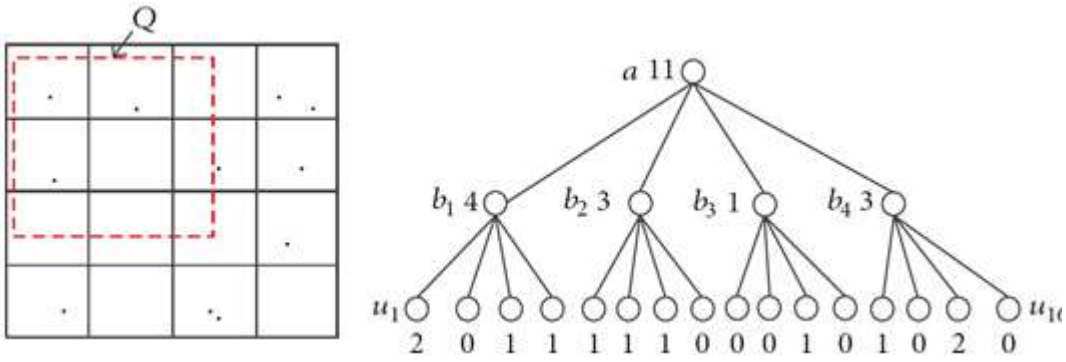

Figure 2.9: An example of the Quad-opt method.

(Source: Wang et al., 2015)

Going beyond the obvious choice of a uniform budget, i.e. $\varepsilon_i = \frac{\varepsilon}{h+1}$, Cormode et al. (2012) proposed a geometrical strategy, with a factor of $\sqrt[3]{2}$. Based on this strategy, budget $\varepsilon_i = 2^{\frac{h-i}{3}}\varepsilon\left(\frac{\sqrt[3]{2}-1}{2^{\frac{h+1}{3}}-1}\right)$, with $i = 0, \ldots, h$ and $\varepsilon = \sum_{i=0}^{h}\varepsilon_i$. For a query Q, the error accumulated is equal to the sum of the errors occurring in each node and for the upper error boundary in query Q, it holds that $Error(Q) = Var(Q) \leq \frac{2^{h+7}}{\varepsilon^2}$. Method Quad-opt can further enhance the accuracy of the query, by applying post-processing on the noisy data with the aid of the least square method. When the entire privacy budget is allocated to the leaf nodes, the method implies decomposition of the data set into $w \times w$ cells.

Fan et al. (2013) proposed another data-independent decomposition method, called Spatial Estimation Algorithm (SEA), which is merging similar cells into groups, proved to be effective especially in cases of sparse data sets. The SEA method starts with modeling the cells, by grouping them into different cell types (dense or sparse) that depend on the knowledge of the domain before decomposition. Any group of cells is characterized as homogeneous when every cell of the group is of the same type (dense or sparse) and no need for further decomposition exists. Further decomposition is needed in case that a group is not homogeneous, if the predefined depth of the tree has not been reached. In SEA, noise is added in each one of the groups after completion of the required decomposition, and the total cell noise is given by the formula $\frac{noise}{g_i \times size}$, where $g_i \times size$ is the number of cells in group g_i. The main advantage of

SEA is that it enhances the accuracy of the query, while at the same time manages to provide a high-level privacy guarantee by merging the cells, or in other words by reducing the noise added to each cell.

While the decomposition methods *kd*-standard and quadtree are suitable for one-dimensional or two-dimensional data sets, they are not appropriate for multi-dimensional data sets. Qardaji et al. (2013) proposed UG which is based on a uniform grid that simulates an n-tree structure. The error generated in the UG method is coming from two separate sources. First, a random noise error is introduced following Laplace distribution and second a non-uniformity error is added as a result of the assumption that that the data points are uniformly distributed. The non-uniformity error is mainly driven by the data points that are positioned in the border cells of the query rectangle. Since the UG method decomposes the data set into w×w cells, for a query Q the error is given by the formula $Error(Q) = \frac{\sqrt{2r}}{\varepsilon} + \frac{\sqrt{r}N}{wc_0}$, with N being the number of data points, r the ratio of the query area Q to the overall domain area, and c_0 a constant. It is important to notice that the granularity $w = \sqrt{\frac{N\varepsilon}{\sqrt{2}c_0}}$ of the decomposed space is the main factor that effects the accuracy of the query results.

The UG method treats similarly dense and sparse cells. The result of this is that, in cases where a cell is sparse, the noise error is large, while for dense cells the non-uniformity error gets larger. This was the main trigger for Qardaji et al. (2013) to propose the Adaptive Grid (AG) method that targets in balancing the above two errors. To achieve this balance, AG initially decomposes the data space into $w_{init} \times w_{init}$ cells with non-fine granularity. Then, cells that are too dense are further decomposed into $w_{gran} \times w_{gran}$ more granular cells. After applying the decomposition, a count query is created for each cell, using privacy budget $a \times \varepsilon$ for the initial coarse cells and $(1 - a) \times \varepsilon$ for the more granular cells. To optimize the error, Qardaji et al. (2013) used the following values:

$$w_{init} = max\left(10, 0.25\sqrt{\frac{N\varepsilon}{c}}\right) \quad \text{(eq. 2.15)}$$

$$w_{gran} = \sqrt{\frac{2\tilde{x}_i(1-a)\varepsilon}{\sqrt{2}c_0}} \quad \text{(eq. 2.16)}$$

where $\tilde{x}_i$ is the noisy count of the cell c_i.

The main drawback of AG is that it still lacks a solid adaptive rule to decide if a cell is dense or sparse. Other issues with both UG and AG are that they are both using the assumption that the non-uniformity error is driven by the number of data points included in the cells, the rectangular shape of the grid, and the heavy dependency of the query results on the selection of c_0.

Overall, it has become evident from the above analysis that in data-dependent decomposition methods, such as *kd*-tree, a part of the overall privacy budget is required to guarantee the privacy. Minimizing the privacy budget to increase the query accuracy is one of the main challenges and future research topics. In the case of data-independent decomposition methods there is not privacy risk, but the reduction of the noise error is a significant challenge. On top, the level of granularity of the performed decomposition, is one of the main challenges and future research topics. A summary of all tree structure data release methods is depicted in Table 2.3.

Table 2.3: Tree structure data release methods.

Method	Strategy	Advantage	Defect	Budgeting
Quad-opt	Data Release Strategies 2, 3	Computational efficiency is high; support any range query	Error is large	Geometric budgeting
SEA	Data Release Strategy 2	Noise error is small	Approximate error is large	Uniform budgeting
kd-standard	Data Release Strategy 2	Query accuracy is high	Only fit for low-dimensional data	Median and count use privacy budget together
UG	Data Release Strategy 1	Support any range query	Neglects the balance between noise error and nonuniformity error	Uniform budgeting
AG	Data Release Strategy 1	Balance the noise error and the nonuniformity error	Lack of the adaptive judging criterion	Uniform budgeting

2.4.3.3 Time series release type

The time series release type methods are using time series data to reduce the DP error. One of the main representatives of this category is the DFT_k method, proposed by Rastogi and Nath (2010). For time series D, DFT_k executes the following steps:

1. Applies a discrete Fourier transform $F=DET(D)$ from the output of which only the first k *DET* coefficients are kept.
2. Adds Laplace noise to the k coefficients that have been kept.
3. Executes inverse Fourier transform on the noisy coefficients $\widetilde{D} = IDET(\tilde{F})$.
4. Releases the perturbed data $\widetilde{D}$.

Although from a result production point of view DFT_k works well, from a performance point of view it cannot be used in real time applications since both *DET* and *IDET* operations are required to be applied on the full time series (Fan et al., 2013).

To overcome the performance issues of DFT_k, Fan et al. (2013) proposed the Temporal Estimation Algorithm (*TEA*). For a space G, this method executes the following steps:

1. Splits G into smaller $w \times w$ cells and discretizes the time span T using a time index $k \geq 0$, and $k < T$. With this split, the frequency series D^c for cell c can be defined as $\{x_k^c | 0 \leq k < T\}$, x_k^c being the frequency series of cell c at time step k and $\{D^c | \, c \in G\}$ being all frequency series G^c in G.
2. Constructs the model to be dependent on the broader domain knowledge, i.e. population, network, and locations. Mathematically, this can be formulated in the following way: for every cell c, $x_{k+1}^c = x_k^c + \omega^c$, and $p(\omega^c) \sim N(0, Q^c)$, Q^c being the level of variation between neighbouring time steps, or the noise added to x_k^c. This is $\widetilde{x_k^c} \sim x_k^c + noise$, where $noise \sim Lap\left(0, \frac{1}{\varepsilon_0}\right)$, $\varepsilon_0 = \frac{\varepsilon}{T}$, ε being a uniform budget for the overall time span T. To facilitate calculations, *noise* can be replaced by white Gaussian noise, i.e. $noise \sim N(0, R)$.
3. Releases the noisy data set $\widetilde{D^G}$.

The main advantage of TEA compared to DFTk is that the computational complexity of TEA is lower ($O(kw^2)$ with $k = O(1)$) since it takes advantage of an "*educated guess*" to extend the model accuracy.

The performance of TEA in real time applications depends heavily on the sampling rate; the higher the sampling rate is, the larger amount of noise should be added to achieve differential privacy. To appropriately adapt the sampling rate, an adaptive sampling method known as FAST was proposed (Fan et al., 2013b; Fan & Xiong, 2014). The main idea on which FAST is based is, that the sampling rate *F* can be adjusted automatically according to the dynamics of the data, i.e. the sampling increases when the data changes fast while it decreases when the data changes slow. The dynamics of the data speed could be described by the *feedback error* $E_{k_n} = \frac{|\hat{x}_{k_n} - \hat{x}^-_{k_n}|}{\max(\hat{x}_{k_n}, \delta)}$, with $\hat{x}_{k_n}$ and $\hat{x}^-_{k_n}$ being respectively the post-state and the pre-state estimates of the n^{th} sampling point at the k_n time step and δ being a user defined parameter (normally δ=1). To reach the appropriate sampling rate, FAST uses a specific algorithm, known as PID, that controls and adjust the rate. With the adaptation described above, FAST manages to reduce the order of the noise error from $O(T)$ to $O(F)$.

Fan et al. (2014) proposed an extension of the FAST method, known as *U-KF* to further improve it by utilizing the idea of session-level differential privacy, applicable especially in cases of web browsing. *U-KF* works only for browsing requests in a single server, but still worths mentioning since in the single server case is very effective. To release the DP aggregates, *U-KF* uses a real time state-space approach and applies a Kalman filter (Kalman, 1960) in a post-processing stage. For a data set $x_k^i, i = 1, \dots, m$ with x_k^i being the number of sessions that browse $page_i$ at time step *k* the *U-KF* executes the following steps:

1. Prediction step: predict a pre-estimate of the current step $\widehat{x_k^i}$; this estimate is calculated by using the previous step's post-estimate x_{k-1}^i.
2. Computation step: Compute the perturbed value $\widetilde{x_k^i} = x_k^i + noise_k^i$, and $noise_k^i \sim Lap\left(0, \frac{l_{max}}{\varepsilon}\right)$, with l_{max} being the sensitivity of the query.

3. Correction step: the post-estimate value of $\widehat{x_k^i}$ is calculated by using the perturbed value $\widetilde{x_k^i}$ and the pre-estimate value $\widehat{x_k^{i-}}$, i.e $\widehat{x_k^i} = \widehat{x_k^{i-}} + K_k^i\left(\widetilde{x_k^i} - \widehat{x_k^{i-}}\right)$, with K_k^i being the Kalman gain.

4. Release step: The final calculated value $\left\{\widehat{x_k^i}, i = 1, \dots, m\right\}$ is released.

The main advantage of *U-KF* is that this method is a univariate time series algorithm with high query accuracy and computational effort proportional to the number of web pages, i.e. $O(m)$.

A summary of all time series data release methods is depicted in Table 2.4.

Table 2.4: Time series data release methods.

Method	Strategy	Advantage	Defect	Budgeting
DFT_k	Data Release Strategy 2	Supports long-range query	Parameter affects query accuracy	Uniform budgeting
TEA	Data Release Strategies 1 and 2	Computing complexity is low, query accuracy is high	Affected by data type	Uniform budgeting
FAST	Data Release Strategies 1 and 2	Query accuracy is high, adaptive data changes	Add the cost of feedback control	Uniform budgeting
U-KF	Data Release Strategy 2	Computing complexity is low, query accuracy is high	Only supports requests in signal server	Uniform budgeting

2.4.3.4 Graph release type

Given the high sensitivity of network data when noise is added, a number of specialized data release methods have been proposed to enhance the usability of sanitized data for this category. In this context, the edge DP approach for networks was proposed by Hay et al. (2009) as follows. Given two neighbouring data sets Gr_1 and Gr_2 whose difference is (at most) one edge, i.e $|Gr_1 \Delta Gr_2| = 1$, and a random function $F: D \to R$, then $Range(F)$ can be defined as the data set that contains all potential outputs of F in Gr_1 and Gr_2; for any $S \subseteq Range(F)$, F is defined as the *ε-edge DP* (or *ε-DP*) if the following relation holds:

$$\Pr[F(D) \in S] \leq e^{\varepsilon} \Pr[\, F(D') \in S] \qquad \text{(eq. 2.17)}$$

The above definition can be generalized for neighbouring data sets whose difference is k edges, i.e $|Gr_1 \Delta Gr_2| = k$. In this case the function F is defined as *ε-k edge DP*. Even further, in case that neighbouring data sets that differ up to all the edges are linked with a single node (i.e. $|Gr_1 \Delta Gr_2| = \forall k$), the function F is defined as *ε-node DP*. Even though achieving an *ε-node DP* looks very desirable, it has the disadvantage that the noise introduced in this case is too high resulting into issues in the quality of the data.

To increase the usability of the graph for a given level of DP, its structure should be appropriately captured and then, with the addition of noise, it should be converted to a synthetic graph equivalent to the original one. Based on this approach, the Pygmalion method was proposed by Sala et al. (2011); in Pygmalion graph capturing is being done by using a *dK*-series graph model (Mahadevan et al., 2006). Figure 2.10, explains the transformation for two different values of *d*, i.e. *d* =1 and *d*=2. A synthetic graph is then being generated from the *dK*-series by using a matching generator. As explained in the figure, Pygmalion captures the *dK*-series from the original graph, clusters the *dK*-series to sets of sub-series and then perturbates the sub-series with a random noise and using a local sensitivity; as a final step of the method, the perturbated sub-series are combined to create a synthetic graph. The drawback of Pygmalion is that the local sensitivity might cause reveal of sensitive information as a price for reducing the magnitude of the noise.

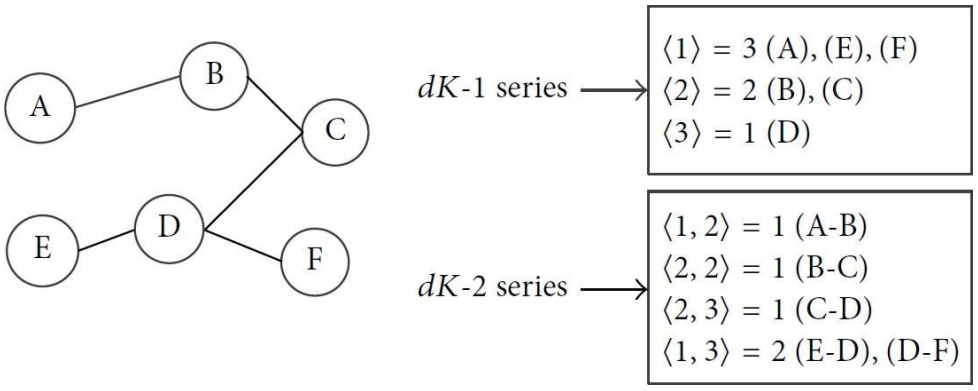

Figure 2.10: An example of the Graph release method (*dK* series).

(Source: Wang et al., 2015)

To resolve the drawback of Pygmalion method, Wang and Wu (2013) introduced a "*smooth sensitivity*" approach and proposed method *DP2K(ε)* which satisfies *(ε,δ)*-DP for *2K*-graphs. *DP2K(ε)* is based on the execution of the following steps:

1. Use parameter *(ε,δ)* to calculate *(β,α)*, with $a = \frac{\varepsilon}{2}$, and $\beta = \frac{\varepsilon}{2ln\left(\frac{2}{\delta}\right)}$.

2. Calculate the *β*-smooth sensitivity $S_{f,\beta}(G)$ and the sensitivity $LS_f(G)$.

3. With the aid of $S_{f,\beta}(G)$ obtain a random noise and add it to the *2K*-series graph.

4. Generate a new graph by perturbating the one created in step (3) above.

Wang et al. (2103) proposed a different approach, known as LNPP method, which is based on the use of a spectral graph for guaranteeing the data. The LNPP method first decomposes the matrix A that represents graph G (i.e. the adjacent matrix that encodes the topological structure of the graph) and then calculates its eigenvectors and eigenvalues. A Laplace based perturbation is applied on the calculated values of eigenvectors which are then post processed by applying vector orthogonalization since the previous transformations conclude to non-orthonormal eigenvectors. By utilizing the spectral graph and the *dK*-graph models for privacy guarantee privacy, noise of order $O\left(\sqrt{n}\right)$ and sensitivity of order $O(n)$ is introduced respectively in the calculations, *n* being the number of the vertices of the graph.

To improve the drawbacks of *DP2K(ε)* and LNPP methods, Xiao et al. (2014) proposed a method based on the Hierarchical Random Graph (HRG) model, known as $hrg - \varepsilon_1 - e - \varepsilon_2$. The HRG model represents graph *G* as a hierarchical structure (a binary tree with *n* leafs that represent the *n* vertices of *G*), including also the information of connection probabilities; for every internal node *r*, a probability p_r applies, while for any two vertices *i* and *j*, the connection probability is provided by the following formula (eq. 2.18):

$$p_{i,j} = p_r, \quad p_r = \frac{e_r}{n_{L_r} * n_{R_r}} \qquad \text{(eq. 2.18)}$$

with *r* being the two-vertices lowest common ancestor in *T*, L_r and R_r being respectively the left and the right sub-trees of the internal node *r*, n_{L_r} and n_{R_r} being respectively the number of leaves of the left and the right sub-trees, and e_r being the number of edges for which the end points are leaves of the two sub-trees of *r* in *T*. With the notation just explained, the HRG can be defined as $(T, \{p_r\})$.

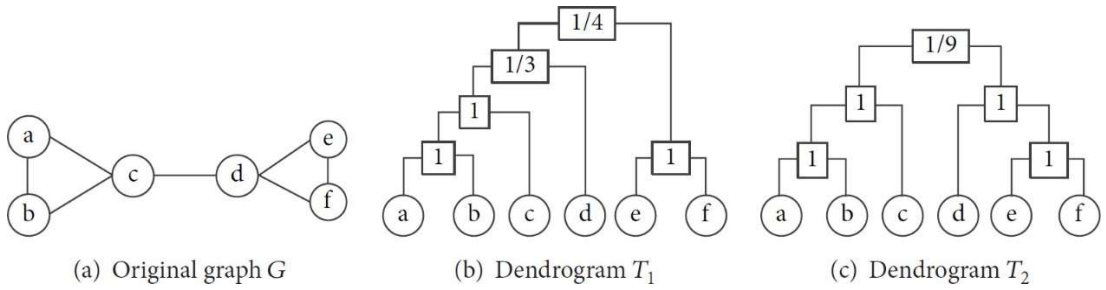

Figure 2.11: An example of HRG model.

(Source: Wang et al., 2015)

Figure 2.11 above shows how the original graph G could be represented by treegrams, or dendrograms T_1 and T_2. For a graph G, to select the proper dendrogram, the following formula (eq. 2.19) has been proposed for the calculation of the likelihood of each dendrogram:

$$L(T, \{p_r\}) = \Pi_{r \epsilon T} \, p_{p_r}^{e_r} (1 - p_r)^{n_{L_r} * n_{R_r} - e_r} \qquad \text{(eq. 2.19)}$$

The calculated likelihoods are then compared and the dendrogram with the larger one is selected since it better represents the original graph.

In the $hrg - \varepsilon_1 - e - \varepsilon_2$ method, the calculation of the DP budget ε is split into two parts. For the first part an exponential calculation mechanism is being used; specifically, a Markovian Monte Carlo procedure is utilized to select a good candidate dendrogram $T_{candidate}$, for which any single edge change might affect only one probability. The DP budget consumed in this part is ε_1, while the score function is given by the following formula (eq. 2.20):

$$logL(T, \{p_r\}) = -\Sigma_{r \epsilon T} \, n_{L_r} * n_{R_r} * h(p_r), \qquad \text{(eq. 2.20)}$$

where

$$h(p_r) = -p_r log p_r - 1 - p_r (1 - log p_r) \qquad \text{(eq. 2.21)}$$

For the second part, a Laplace calculation mechanism is applied and the p_r set of dendrogram $T_{candidate}$ is perturbed thus using the remaining budget ε_2. After completion of the budget split, the sanitized graph $\tilde{G}$ is created using the HRG model. With the above steps, the sensitivity of the $hrg - \varepsilon_1 - e - \varepsilon_2$ method costs $O(\log n)$ which is significantly better from the previously mentioned methods.

A summary of all graph data release methods is depicted in Table 2.5 below.

Table 2.5: Graph data release methods.

Method	Strategy	Advantage	Defect	Budgeting
Pygmalion	Data Release Strategy 2	Reducing the noise by subsection	Having the risk of privacy disclosure using local sensitivity	Uniform budgeting
DP2K(ε)	Data Release Strategy 2	Guaranteeing the privacy by smooth sensitivity	Noise is large	Uniform budgeting
LNPP	Data Release Strategies 1, 2	Enhancing the utility by postprocessing	Noise is large	Uniform budgeting
hrg-ε_1-e-ε_2	Data Release Strategy 2	Utility of the data is high	Result affected by different budgeting strategies	Sampling and perturbation use privacy budget together

2.4.3.5 Pattern mining release type

A frequent pattern can be defined as a subset of data items which appear frequently in a data set and is the basis of data mining techniques such as classification and clustering. Since the existence of frequent patterns may reveal privacy of an individual during data mining, the study of differential privacy in the framework of data mining has become a significant research topic. There are three different pattern mining release methods, measured by the degree of support (Shen & Yu, 2013; Li et al., 2012; Lee & Clifton, 2014), the occurrence (Luca & Li, 2013), or the stay point (Ho & Ruan, 2011; 2013). For a transaction data set $D = \{tran_1, tran_2, \dots, tran_n,\}$, $tran_i \in T$, given that every transaction is consisted of items, i.e. $tran_i = \{item_1, item_2, \dots, item_n,\}$, $item_m \in I$, where I is the space of all items and $itemset_j$ is a subset of item space I, if $tran_n$ includes $itemset_j$ then $tran_n$ supports $itemset_j$. In this case as support degree of $itemset_j$ is defined the ratio of the number of transactions supporting $itemset_j$ to the total number of transactions. Obviously, the higher the support degree of $itemset_j$, the more frequently it appears in the transaction data set.

To mine useful patterns for geographic locations of interest, Ho & Ruan (2011) proposed the *BuildDPQuadTree* method which is based on quadtrees and the density-based clustering algorithm (DBSCAN), and it takes advantage of the notion of stay point mentioned above. Let the

data trajectory vector be defined as $traj_i^{k_i} = \{x_1, x_2, \dots, x_{k_i}\}$, $x_j = (latt_j, long_j, t_j)$, t_j being a timestamp, $latt_j$ the latitude, and $long_j$ the longitude of x_j. In case that for a period equal to ΔT, a trajectory *stays* within a cyclic region with radius ρ, the center of this circle $(latt_j, long_j)$ can be defined as a stay point. On top, for a given a set of trajectories $TJ = \{traj_1^{k_1}, traj_2^{k_2}, \dots, traj_s^{k_s}\}$, the region is known as location of interest, if it includes more than *r* stay points.

The *BuildDPQuadTree* method (Ho & Ruan, 2011) repeatedly splits the data space *G* into subspaces, taking into consideration a stay-point set *S*, a threshold T_{thres}, and a privacy budget ε_{q_t}. Splitting continues until $|S| + Lap\left(\frac{3T_{thres}}{\varepsilon_{q_t}}\right) \leq 3T_{thres}$. After this iterative splitting process, a set of clusters G_j is obtained by applying density-based spatial clustering using the noise clustering method (Ester et al., 1996). For a given threshold *r*, and privacy budgets ε_{cg} and ε_{cts}, then G_j is called a region of interest, if the following relation (eq. 2.22) holds (Ho & Ruan, 2011):

$$\left|G_j\right| + Lap\left(\frac{\Delta f_{c_{ts}}^j}{\varepsilon_{cts}}\right) \geq r \qquad \text{(eq. 2.22a)}$$

where

$$\Delta f_{c_{ts}}^j = max_{i \in D} \#\{s \in G_j | s \; is \; a \; stay \; point \; for \; i\} \qquad \text{(eq. 2.22b)}$$

D being the set of all *is* with records in cluster G_j.

In this case, the center location of the region of interest is given by the following formula (eq. 2.23) (Ho & Ruan, 2011):

$$\frac{\sum_{k=1}^{|G_j|}(latt_k, long_k)}{Lap\left(\frac{\Delta f_{cg}^j}{\varepsilon_{cts}}\right)} \qquad \text{(eq. 2.23a)}$$

with

$$\Delta f_{cg}^j = \frac{maxlengt\ \{point_x, point_y\}}{2}, \quad point_i \in G_j \qquad \text{(eq. 2.23b)}$$

and

$$\varepsilon = \sum_{t=1}^{h} \varepsilon_{qt} + \varepsilon_{cg} + \varepsilon_{cts} \qquad \text{(eq. 2.23c)}$$

where h is the depth of quad-tree.

Since in their initial work where the *BuildDPQuadTree* method was established, the level of noise is significantly affected by parameter T_{thres}, in research that followed the original one the algorithm *DP-ILD* was proposed, that utilized the idea of *β-smooth* sensitivity satisfying *(ε,δ)*-DP (Ho & Ruan, 2013).

When the *Top-k* frequent pattern mining approach is applied, the sensitivity of the transaction data set depends on the number of dimensions. Thus, to reduce the sensitivity, the original data set is usually projected into a dimensional space with lower number of dimensions. In this direction, the *PrivBasis* method was proposed by Li et al. (2012); *PrivBasis*, as an initialization step, targets to the reduction of the original data space by identifying smaller data sets I_B that contain the top k frequent item sets. As a next step, a binary support counting technique is applied (Chen & Xiao, 2010) on the identified data sets that produces the supports of the I_B subsets. The drawback of the *PrivBasis* method is the accuracy of the final frequent patterns obtained.

To resolve the accuracy issue of *PrivBasis*, the method *NoiseCut*, which was based on the notion of the *FP-tree* and the sparse vector technique, was proposed by Lee & Clifton (2014). The *NoiseCut* in its first step, identifies all frequent data subsets L by applying a sparse vector technique that reduce the consumption of the DP budget; for a noisy threshold $\hat{\tau}$ and a count-type query q, privacy budget is consumed only in case that $q(D) + Lap\frac{2k}{\varepsilon} \geq \hat{\tau}$, with $\hat{\tau} = \sigma_k + Lap(.)$, σ_k being the support of the k-th in series most frequent item set. To decide if an item set is frequent, *NoiseCut* compares its noisy support $\tau + \alpha$ with $\hat{\tau}$ and if $\tau + \alpha > \hat{\tau}$, then this item set it considered to be frequent. In its second step, *NoiseCut* builds the noisy *FP-tree* utilizing L and can then decide the supports of item sets and consequently the *Top-k* frequent item sets. The false negative of this calculation is $e^{-\frac{\alpha\varepsilon}{2}\left(\frac{\alpha\varepsilon}{2}+2\right)}{4}$, and the smaller it is the better frequent patterns are obtained.

Leveraging the idea of the method they developed for the graph data release type, Shen & Yu (2013) proposed a frequent graph pattern mining method based on a Markov chain and a Monte Carlo random walk, known as *Diff-FPM*. This method bases the *Top-k* frequent sub-graph selection on the result of a random walk. If the random walk reaches a steady state, then the actual counts of the sub-graphs are perturbed with the aid of a Laplacian random noise and *Diff-FPM*

satisfies ε-DP; if no steady state can be reached, *Diff-FPM* satisfies (ε,δ)-DP and in this case data security issues might appear.

Luca & Li (2013) highlighted in their research that for sequential data, e.g. DNA data sets, frequent mining patterns might fail to be calculated properly. To resolve this issue, the authors proposed the notion of occurrence and a two-phase algorithm for mining sequential patterns with differential privacy (Luca & Li, 2013b). For a data pattern $p = a_0 a_1 \dots a_{n-1},\ a_i \in \Sigma$ and a character string $x = x_0 x_1 \dots x_{m-1}$, in case that there is an integer $i \in (0, m-n)$ for which $x_{i+j} = a_j,\ j = 0, \dots, n-1$, then it is said that the pattern p appears in position i of the character string x and $f_x(p)$ is the number of all occurrences of p in x. The equivalent for a data set $D = \{x^0, x^1, \dots, x^{N-1}\}$, is the notion of $F_D(p) = \sum_{i=1}^{N-1} f_{x^i}(p)$ which denotes the number of occurrences of the sequential pattern p in the given data set D; in that case, if $F_D(p)$ is greater than the value of the threshold, then p symbolizes the frequent sequential pattern.

A summary of all pattern mining data release methods is depicted in Table 2.6.

Table 2.6: Pattern mining data release methods.

Method	Strategy	Advantage	Defect	Budgeting
BuildDPQuadTree	Data Release Strategy 2	Identify the interesting location using stay point	Noise is large	Partition and cluster use the privacy budget together
DP-ILD	Data Release Strategy 2	Utility of the data is high	Only supports offline data	Uniform budgeting
PrivBasis	Data Release Strategy 2	Mining speed is fast	The final frequent patterns may be imprecise	Uniform budgeting
NoiseCut	Data Release Strategies 1, 2	Utility of the data is high	Results are affected by different budgeting strategies	Privacy budget is allocated for two steps of the algorithm
Diff-FPM	Data Release Strategy 2	Query accuracy is high	The drop in utility with the increase of the number of outputs	Sampling and perturbation use the privacy budget together

2.5 Calculation of Shapley value with differential privacy

In this section, the way that Shapley calculation algorithms could work when combined with a DP mechanism is described. Especially, sensitivity and noise formulation for this case are stated, while the impact on the loss function and boundaries are also presented.

2.5.1 Sensitivity and noise

In algorithms such as the truncated Monte Carlo Shapley algorithm, the privacy of the data is under risk during the Shapley value computations, since to compute the Shapley value for a point i, requires all other data points to be processed, putting thus at risk their privacy. By design, DP protects data privacy and acts proactively to avoid data leaks. It thus provides a privacy guarantee for every data point of the data set, by making sure that the Shapley value is not statistically sensitive to the addition or removal of individual data points. Under that perspective, the Shapley value is assumed to be a function and its sensitivity (and consequently the noise added by any mechanism, e.g. Laplacian) can be calculated. Of course, transforming a Data Shapley calculation to differential private, comes always with the challenge to achieve in parallel the highest possible Shapley calculation accuracy.

2.5.2 Loss function and boundaries

Let L represent the loss function of the related logistic regression, i.e.:

$$L(w,x) = \frac{1}{n}\sum log\left(1 + e^{-w^T x_i y_i}\right) \quad \text{(eq. 2.24)}$$

where *w* is the vector of logistic regression parameters, x_i is each datum and y_i is its label.

To calculate the above loss function for a test set, other than the train set (used to calculate w), an upper boundary *M* should be assumed, so that:

$$|L(w,x) = L(w',x)| \leq M \quad \text{(eq. 2.25)}$$

In the framework of differential privacy, it is interesting to examine what happens to the loss function if a datum *x* is replaced with a datum x' in the trainset. In such a case, calculating the loss function for each *x* that belongs to the test set, will cause change of the vector of the parameters *w*, to w'. The upper boundary M can be calculated based on relative research (Bousquet & Elisseeff,

2002). Additionally, a linear regression can be performed for with the loss function providing another approximate upper boundary M':

$$L(w,x) = \frac{1}{2n}\sum(w^T x_i - y_i)^2 \qquad \text{(eq. 2.26)}$$

In any case, identification of an upper boundary for the loss function is instrumental for applying any differential privacy strategy and that is why researchers are focusing on identifying such a boundary.

2.6 Discussion, challenges and areas of optimization

The current research proposes the extension of Shapley value calculation with differential privacy embedded. Three different data Shapley valuation algorithms have been implemented and tested, suitably transformed to preserve differential privacy, namely the TMC-Shapley, G-Shapley and for comparison reasons the classic LOO. By executing the implemented algorithms, it has been validated that both differentially private and non-differentially private Shapley values calculated are meaningful in terms of data set accuracy, which practically means that their outputs could be used in the same way.

Even though the theoretical results studied around the scientific domain of differential privacy injected Shapley valuations, it is evident that relevant research is still far from establishing widely acceptable and usable results. One of the most significant challenges in the specific domain is to connect data sets with actual meaningful use cases, translating thus the scientific results to socially valuable values and metrics.

From a technical perspective, proving that the extension of data Shapley properties to Monte Carlo based approximations still needs further research efforts to be investigated. One of the reasons why this might be a challenge is that calculation of Shapley value for data points that are part of potentially large data sets can prove highly complex and demanding. Another topic of interest is the intuition behind the data Shapley calculated values, i.e. if the calculated Shapley values is aligned with what humans expect that in some cases might be valid while in others not (Kumar et al. 2020; Fryer et al., 2021).

Although the current study is contributing to the enrichment of the data Shapley valuation approaches with the preservation of privacy, its contribution in the provision of meaningful and actionable insights and thus its impact in the social society, remains still ambiguous. The main reason is that both Shapley valuation and differential privacy are domains without straightforward interpretation of their outcomes, making thus unclear if algorithms or mechanisms that combined the two domains could help humans in practice to take better and more accurate decisions, instead of just adding complexity, uncertainty and the fear of technology. The following issues that differential private data Shapley calculations might create, are indicative (Bagdasaryan & Shmatikov, 2019; Ganev et al., 2021):

- Differential private methods do not provide perfect privacy; instead, they produce probabilistic results that might lead to misunderstandings.
- Differential private transformations might become suspicious for unethical and dangerous usage such as cyber-attacks, fraud, or abuse.
- Unethical parties may intentionally provide fake data points targeting in increasing their benefit or reducing the benefits of others.
- Valuation applications may preserve and propagate fraud and create the need for continuous monitoring of the conducted valuations.
- Possible leakage of data with high value, could become an opportunity for attackers and put at risk data holders.
- Incentivizing the data might become a reason for potential abuse, since depending on the circumstances, it could create pressures to data owners to contribute this data or receive a penalty for not contributing it.
- In case that highly valuable data is privacy sensitive, connecting it with incentives might be a good reason for someone to act towards loss of data privacy.

3 IMPLEMENTATION

3.1 Introduction

As part of the current work, calculation methods, such as the truncated Monte Carlo Shapley algorithm, as well as the Gradient Shapley and the Group Shapley algorithms are studied and analyzed. At the same time, sensitivity and noise of the applied data privacy mechanism are measured and analyzed in the context of Data Shapley metrics. Specifically, sensitivity and noise of applied data privacy mechanisms and how those could be applied in the context of Data Shapley metrics, by guaranteeing the correctness and accuracy of calculations without any information leakage, have been implemented and are presented in this chapter.

3.2 Proposed Design

3.2.1 Overall solution

The proposed implementation has been developed exclusively using Python programming language. In particular, two Jupyter notebooks have been developed; the first has been used for executing and demonstrating three Data Shapley algorithms (Truncated Monte Carlo Shapley, Gradient Shapley, Leave One Out) without differential privacy (named *DataShapley_v2.ipynb*), while the second, named *DataShapley_v3.ipynb*, has been used to embed differential privacy with Laplacian DP noise in the Data Shapley algorithms.

Additionally, three python files have been developed and contain all the code required to calculate the Data Shapley values for a given data set, namely *Data_Shapley.py*, *Shapley_Class.py* and *utilities.py*. Finally the Jupyter notebook *DP_v1.ipynb* has been created and used as an independent example to demonstrate how differential privacy with Laplacian DP noise could be applied to a data set. In the following sections the detailed code design is presented, explained and analyzed.

3.2.2 Programming with Python

Python is a programming language developed by Guido van Rossum, starting in 1989. It is a relatively simple and easily understood programming language, which owes its name not to the well-known snake (python) but to a comedy TV show (Monty Python's Flying Circus). The key features of the Python language are presented in Figure 3.1 and can be summarized as follows:

Figure 3.1: Main features of programming language Python.

1) Reliability: This is a truly reliable language with very comprehensive commands, which perform the expected functions in a reliable and effective way.

2) Easy to learn: Python's structure and operation is relatively simple, while its commands are high-level and highly expressive. Therefore, understanding and learning it can be done quickly and relatively easily.

3) Cross Platform: Python can work across all popular operating systems (Mac, Windows, Linux, Unix, etc.) without variations and issues. This makes Python a language with a high degree of portability.

4) Open Source: Python is an open source programming language with all the advantages that this gives to the developers who use it.

5) Wide variety of libraries: There is a wide variety of standard libraries with prepared code and important pre-implemented functions, which can be deployed and utilized without large-scale coding.

6) Free of cost: Python language is provided for free to download and use, which makes it even more attractive.

7) Exception Handling: This feature increases the tolerance of executable code as it runs even when there are errors, which the developer can handle at a later stage of development and debugging.

8) Automatic memory management: Python supports automatic memory management, which practically means that memory is automatically cleared and released, without any need for developer intervention.

9) Advanced features: Python supports, among others, generators and list comprehensions which are very advanced and useful features.

There are multiple areas in which programming with Python is applicable. The following list is indicative, while Figure 3-2 summarizes those areas:

1) Web application development: Popular web frameworks such as Django and Flask have been developed based on Python. With their help, server-side code can be written for operations such as database management, back end logic, etc.

2) Machine Learning: There are many machine learning applications written in Python. Examples are product recommendations on popular sites like Amazon, eBay, etc. and voice-facial recognotion.

3) Data Analysis: Data analysis and visualization using interactive charts is particularly popular in Python.

4) Scripting: Using small programs (scripts) to automate simple tasks such as sending automated emails is very common in Python programming language.

Figure 3.2: Popular applications of programming language Python.

5) Game Development: This is an area where the use of Python is gradually becoming more and more popular.

6) Embedded Applications: The development of embedded applications using Python is rapidly increasing.

7) Desktop Applications: Office applications using Python have already become popular, using libraries such as TKinter and QT.

As already mentioned, one of the most important advantages of Python is the wide range and variety of libraries that contain significant ready-to-use functionality, without requiring the developer to write extensive code. For example, Matplotlib which is Python's traditional design library has strong functional features and its use is typically given for specialized libraries such as NetworkX and Pandas data frames. In areas such as integration and interactivity, Bokeh and Plotly are also top, presenting the advantage of being built upon Javascript, which makes the ability to convert graphics from Javascript an easy task.

3.3 Implementation of Differential Privacy with Laplacian Noise

To demonstrate how differential privacy with Laplacian noise is applied to an algorithm, a python program based on *PyDP* python library has been developed. Using that program, the way an ε-differentially private algorithm works is demonstrated through graphical representations of aggregated statistics over a potential numeric data set which might contain sensitive or private information. As already explained, to preserve data privacy it is fundamental to add noise, transforming in that way the original data set to prevent a potential attacker from realizing the actual original data set. The most popular choice of distribution to add noise is the Laplace distribution because it works effectively with the differential privacy parameter ε.

The Laplace distribution is practically consisted of two adjustment exponential distributions, that mirror the positive values to their negative equivalents as it is depicted in the chart of Figure 3 which is produced by use of the following python code:

```
def show_laplace(x1):
    return npy.exp(-npy.abs(x1))
x1 = npy.arange(-10, 10, 0.01)
```

pltt.plot(x1, show_laplace(x1));

If we read this under the prism of differential privacy, the peak in the middle of the graphical representation could be read as follows: it is probable that someone would choose a number close to zero in which case the result will be close to the actual original result. In case that the selection is a number far from zero, a potential attacker can identify the actual result with a smaller probability.

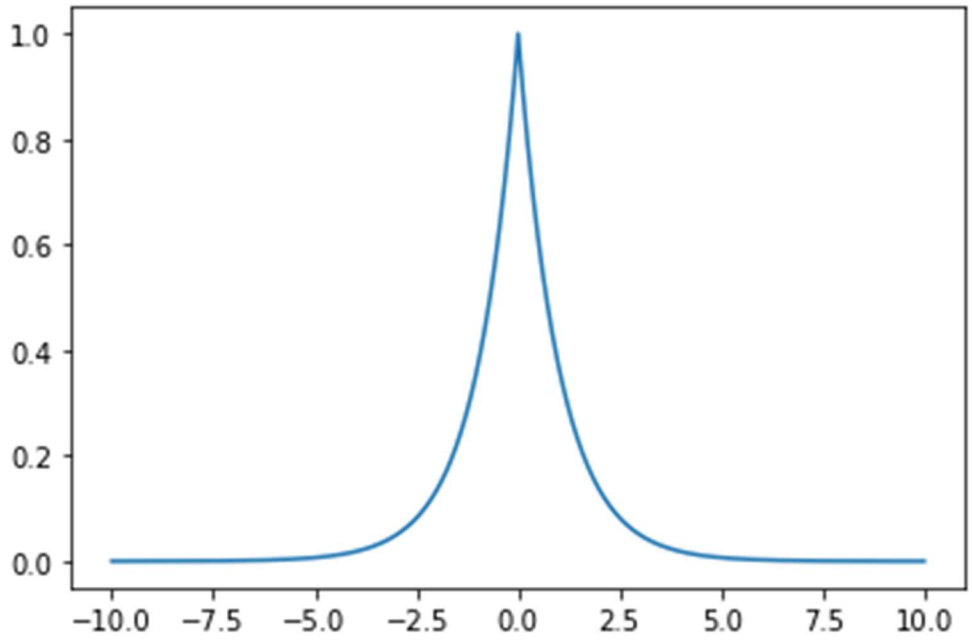

Figure 3.3: Typical Laplace Distribution.

To achieve better results in terms of differential privacy, the Laplace distribution could be adjusted by introducing a scaling parameter, typically defined with the introduction of two additional parameters, namely *m* and *b* as follows:

$$f(x|m,b) = \frac{1}{2b} e^{-\frac{|x-m|}{b}} \qquad \text{(eq. 3.1)}$$

m being the mean of the distribution and *b* the scaling parameter.

From a differential privacy perspective, increasing parameter *b* results to more privacy, since by flattening out the distribution graph, it makes it more probable to choose a higher value

for the noise; this practically implies that potential attackers will be less certain for the accuracy of the results they obtain. The python code below produces the graph depicted in Figure 3.4.

```
def laplace(x1, m, b):
    return 1 / (2 * b) * npy.exp(-npy.abs(x1 - m) / b)
def plot_laplace(x1, m, b):
    pltt.plot(x1, laplace(x1, m, b), label="m={}, b={}".format(m, b))
x1 = npy.arange(-20, 20, 0.01)
plot_laplace(x1, -5, 2)
plot_laplace(x1, 5, 4)
pltt.legend();
```

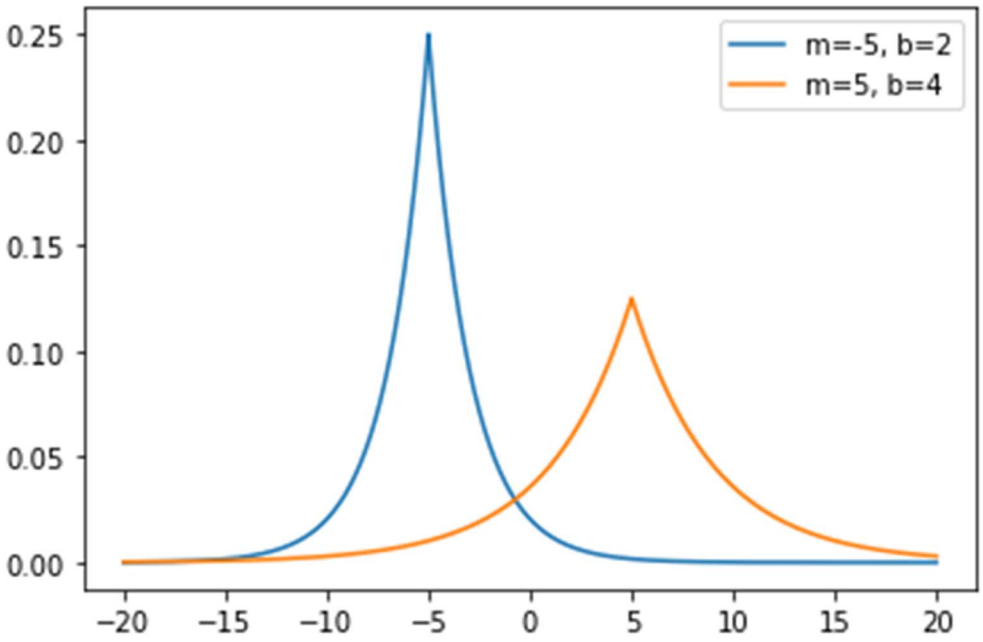

Figure 3.4: Laplace Distribution with different values of *m* and *b*.

As already explained in the previous Chapter, the ε-differential privacy can be defined with ε being a parameter that determines the acceptable levels of privacy loss. More accurately, ε-differential privacy means that the probability to produce the actual result when using this distribution should be smaller by e^{ε} times from the probability when using the original distribution. By setting parameter *b* equal to 1/ε in the Laplace distribution the ε-differential privacy is being achieved as depicted in Figure 3.5 which is produced by the following python code:

```
output_value = 100
epsilon = 0.5
x1 = npy.arange(95, 105, 0.1)
distribution_1 = laplace(x1, output_value, 1 / epsilon)
distribution_2 = laplace(x1, output_value + 1, 1 / epsilon)
pltt.plot(x1, distribution_1, label="distribution_1")
pltt.plot(x1, distribution_2, label="distribution_2")
pltt.axvline(x=output_value, c="red", dashes=(1, 2), label="output_value")
pltt.legend()
probability1 = laplace(output_value, output_value, 1 / epsilon)
probability2 = laplace(output_value, output_value + 1, 1 / epsilon)
probability1, probability2
```

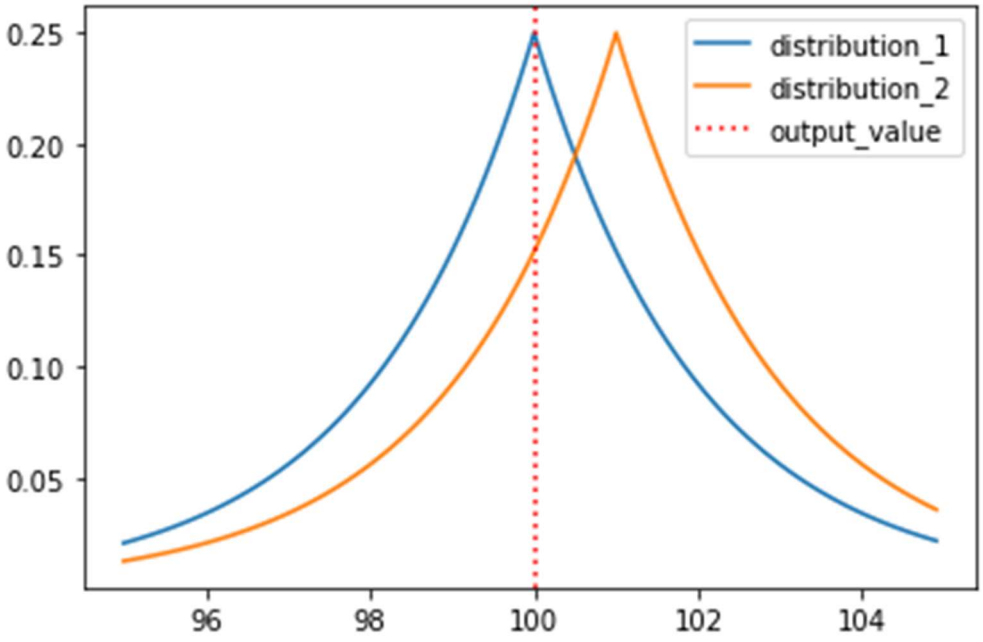

Figure 3.5: Two different ε-Differential Private Laplace Distributions vs the original output value.

In Figure 3.5, the dotted line shows the output value. As could be easily understood, it is more probable that the output value might came from distribution_1 rather than from distribution_2.

3.4 Core Data Shapley Implementation

In the case study proposed in this thesis, a classification problem with logistic regression has been designed, implemented, analyzed and presented. The code of the problem includes set up activities, data set creation, execution of three data Shapley algorithms with/ without embedded differential data privacy noise and graphical presentation of the results obtained. In detail, its design is consisted of the following steps:

1. Create a synthetic data set: A synthetic data set is created based on Bernouli's formula, i.e $y = Bernouli(f(x))$, $f(x)$ being a polynomial of order defined by variable "*difficulty*" and $x \in \mathbb{R}^d$; variable "*important_dims*" determines the number of non-null dimensions d in x

2. Execution stage: An instance that takes care of running all algorithms for the specific synthetic data set, is created. Following this instance creation, runs are repeated multiple times sequentially (in a real-world scenario they should run in parallel). More specifically, this step includes the following sub steps:

    a. Initialization and execution of the instance, is supported by the following function:

    *def __init__(self, X, y, X_test, y_test, num_test, sources=None, sample_weight = None, directory = None, problem = 'classification', model_family = 'logistic', metric = 'accuracy', seed = None, overwrite = False, **kwargs)*

    The arguments used by the initialization function are:

    - *X: represents the data covariates*
    - *y: stores the data labels*
    - *X_test: holds the test and held-out covariates*
    - *y_test: holds the test and held-out labels*
    - *sources: stores the mapping of each point to its group. If it takes the value "None", then each points gets its individual value*
    - *samples_weights: contains the weight of the train samples in the loss function for algorithms where weighted training method is applicable*

- *num_test: indicates the number of data points used for the calculation of the evaluation metric*
- *directory: names the directory where results and figures are saved*
- *problem: defines the problem type to be solved, i.e. "Classification"*
- *model_family: defines the family of the model used for the learning algorithm*
- *metric: represents the evaluation metric used*
- *seed: holds the "random seed" needed to initialize the Monte-Carlo simulations*
- *overwrite: facilitates deletion of existing data to re-execute computations*
- ***kwargs: the arguments of the model*

b. Calculation the data point values is based on the following function:

def run(self, save_every, err, tolerance=0.01, g_run=True, loo_run=True)

The arguments used for the definition of the above function are:

- *save_every: defines the saving frequency of the marginal contributions*
- *err: indicates algorithm's exit criteria*
- *tolerance: defines the truncation tolerance used*
- *g_run: when this variable takes the value "True", the function calculates the G-Shapley values*
- *loo_run: when this variable is "True", the function calculates the leave-one-out scores*

c. Execution of the TMC-Shapley algorithm is supported by the following function:

def _tmc_shap(self, iterations, tolerance=None, sources=None)

The arguments used for the definition of the above function are:

- *iterations: defines the number of iterations for execution of the TMC-Shapley algorithm*

- *tolerance: defines the truncation tolerance ratio of the TMC-Shapley algorithm*
- *sources: indicates if the values used are coming from sources of data points rather than being individual points*

d. Execution of the G-Shapley algorithm is supported by the following function:

def _g_shap(self, iterations, err=None, learning_rate=None, sources=None)

The arguments used for the definition of the above function are:

- *iterations: defines the number of iterations for execution of the G-Shapley algorithm*
- *err: indicates algorithm's exit criteria*
- *learning_rate: defines the learning rate used for the algorithm; when this variable takes the value "None" the algorithm calculates the best learning rate*
- *sources: indicates if the values used are coming from sources of data points rather than being individual points*

e. Execution of the LOO algorithm is supported by the following function:

def _calculate_loo_vals(self, sources=None, metric=None)

The arguments used for the definition of the above function are:

- *metric: defines the metric to be used; when this variable takes the value "None" the algorithm uses the default metric*
- *sources: indicates if the values used are coming from sources of data points rather than being individual points*

The outcome of the algorithm is the calculated "leave-one-out" scores.

3. Merge results of parallel runs: in this step the results from the different runs are merged. The function on which this merge is based is the following:

def merge_results(self, max_samples=None)

The outcome of this step is the merged marginals, sample indices and values calculated for the algorithm used.

4. Convergence plots of the algorithms: in this step the convergence plots produced from the different runs, showing the effect of removing points contributing in the overall value, are presented. The function on which drawing of those plots is based, is the following:

   def performance_plots(self, vals, name=None, num_plot_markers=20, sources=None)

   The arguments used for the definition of the above function are:

   - *vals: provides a list of different valuations of data points*
   - *name: defines the name of the saved plot*
   - *num_plot_markers: indicates the number of points used to draw each plot*
   - *sources: indicates if the values used are coming from sources of data points rather than being individual points*

   The outcomes of this step are the plots that show the performance variations that when data points are removed from the data set, in an order starting from most valuable datum and proceeding to the least valuable ones.

3.5 Core Implementation Results and Discussion

In this section, the results of applying Data Shapley techniques coupled or not with differential privacy noise, are presented and discussed. The first part of the section covers presentation and analysis of the results produced by Data Shapley algorithms without differential privacy, while in the second part the impact of Laplacian noise is also presented and compared to the previous ones.

3.5.1 Results and Discussion for non-Differential Private Data Shapley Methods

To prove convergence of the applied algorithms, the proposed implementation takes care of plotting the marginals. For the Truncated Monte Carlo method, by making use of the relevant python statement *convergence_plots(dshap.marginals_tmc)*, a number of graphs with all marginals is being created as depicted in Figure 3.6. It can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the TMC method for the specific use case.

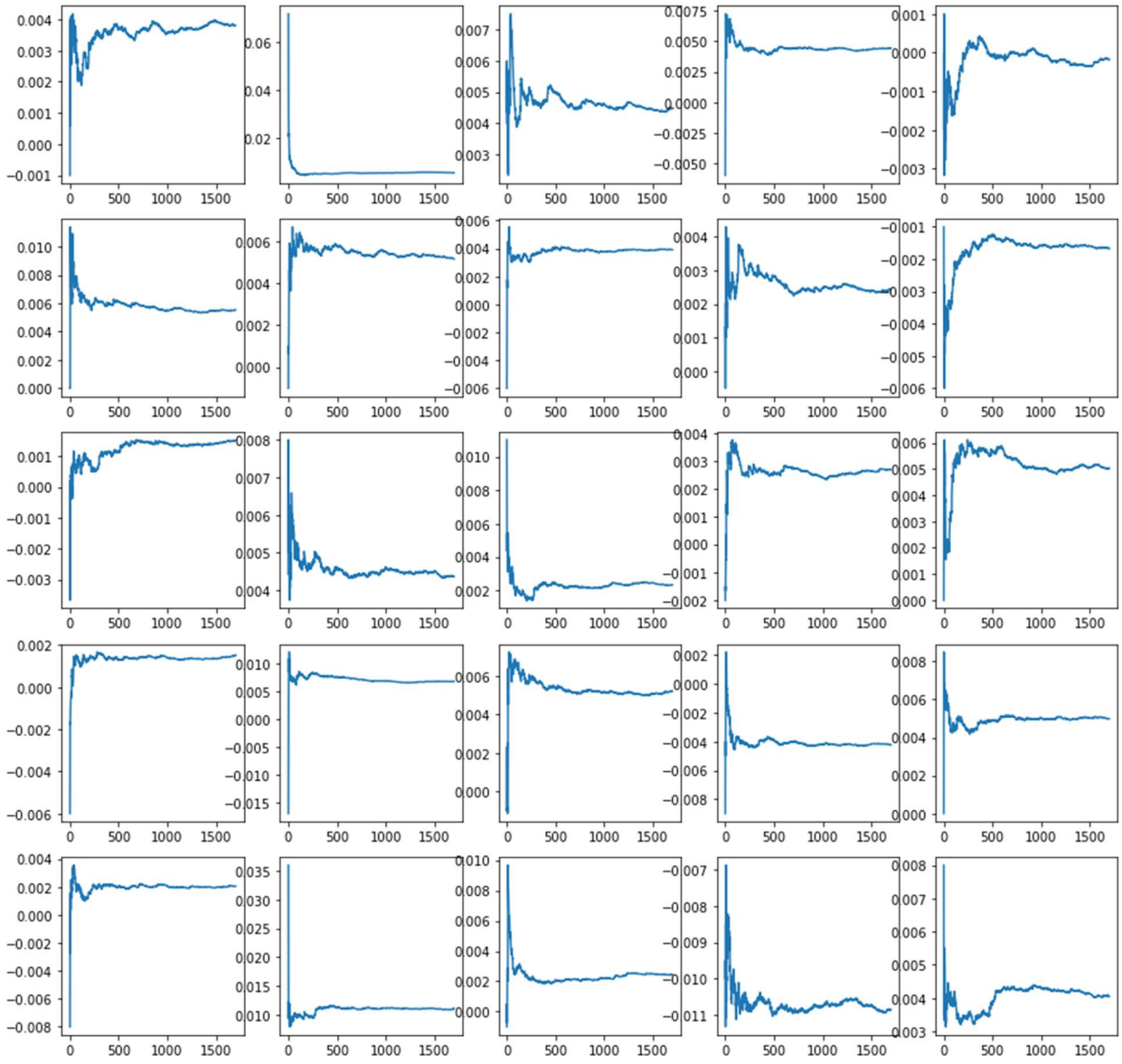

Figure 3.6: Convergence of marginals for the Truncated Monte Carlo method.

For the G-Shapley method, by making use of the relevant python statement *convergence_plots(dshap.marginals_g)*, a number of graphs with all marginals is being created as depicted in Figure 3.7. Again, it can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the G-Shapley method for the specific use case.

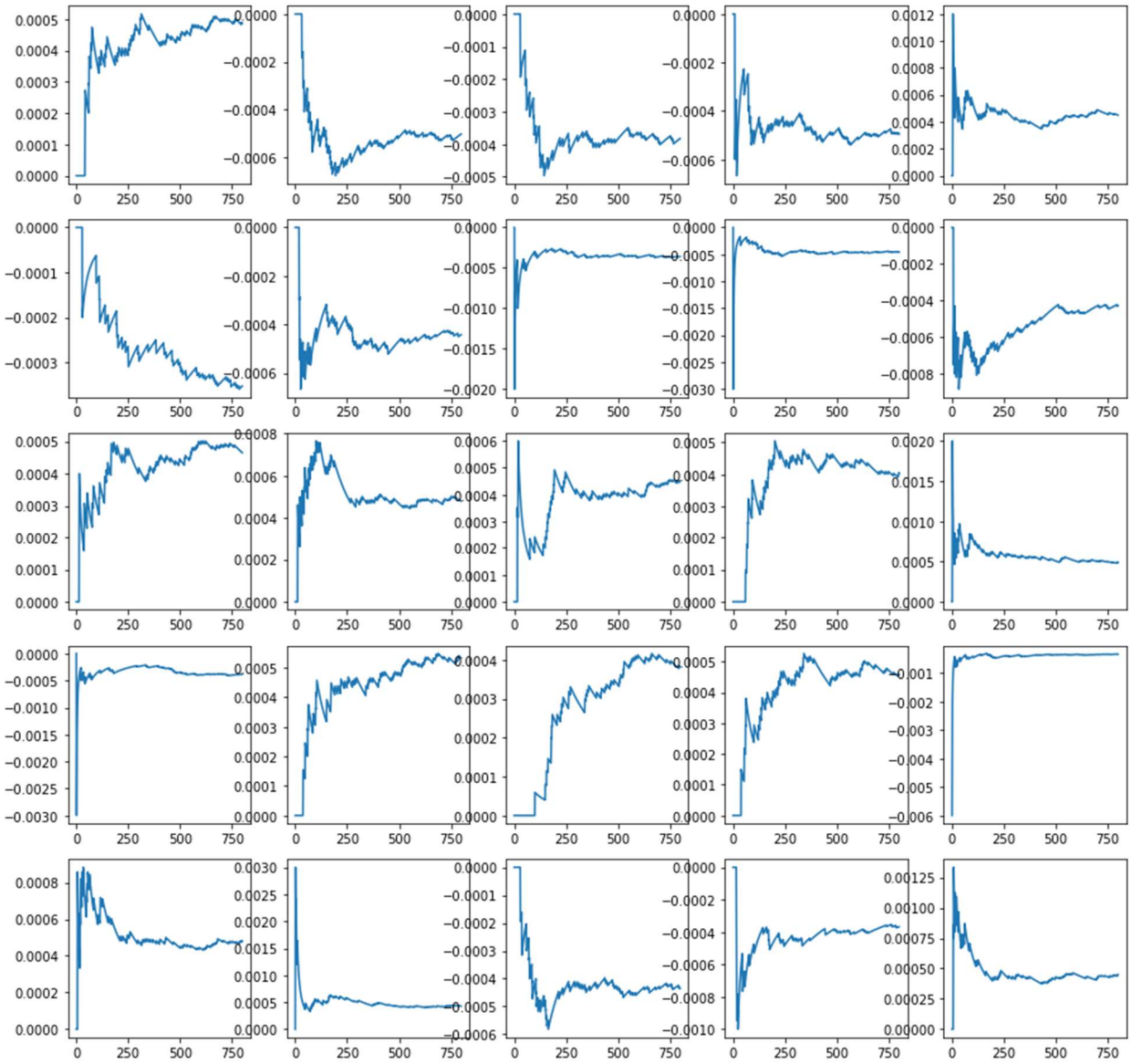

Figure 3.7: Convergence of marginals for the G-Shapley method.

The performance of the studied Data Shapley methods is demonstrated in Figure 3.8 by making use of the relevant python statement *dshap.performance_plots([dshap.vals_tmc, dshap.vals_g, dshap.vals_loo], num_plot_markers=20, sources=dshap.sources)*. It is evidend that both TMC-Shapley and G-Shapley methods outperform the LOO method as well as the Random method, since the accuracy is reduced faster when bigger fractions of trained data are being removed.

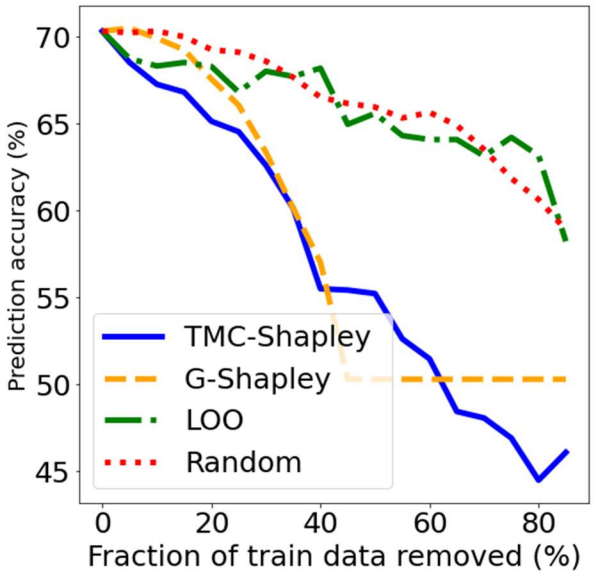

Figure 3.8: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random.

3.5.2 Results and Discussion for NoisyData Shapley Methods

To prove convergence of the applied algorithms, the proposed implementation takes care of plotting the marginals. For the Noisy Truncated Monte Carlo method, by making use of the relevant python statement *convergence_plots(dshap.marginals_tmc)*, a number of graphs with all marginals is being created as depicted in Figure 3.9. It can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the Noisy TMC method for the specific use case.

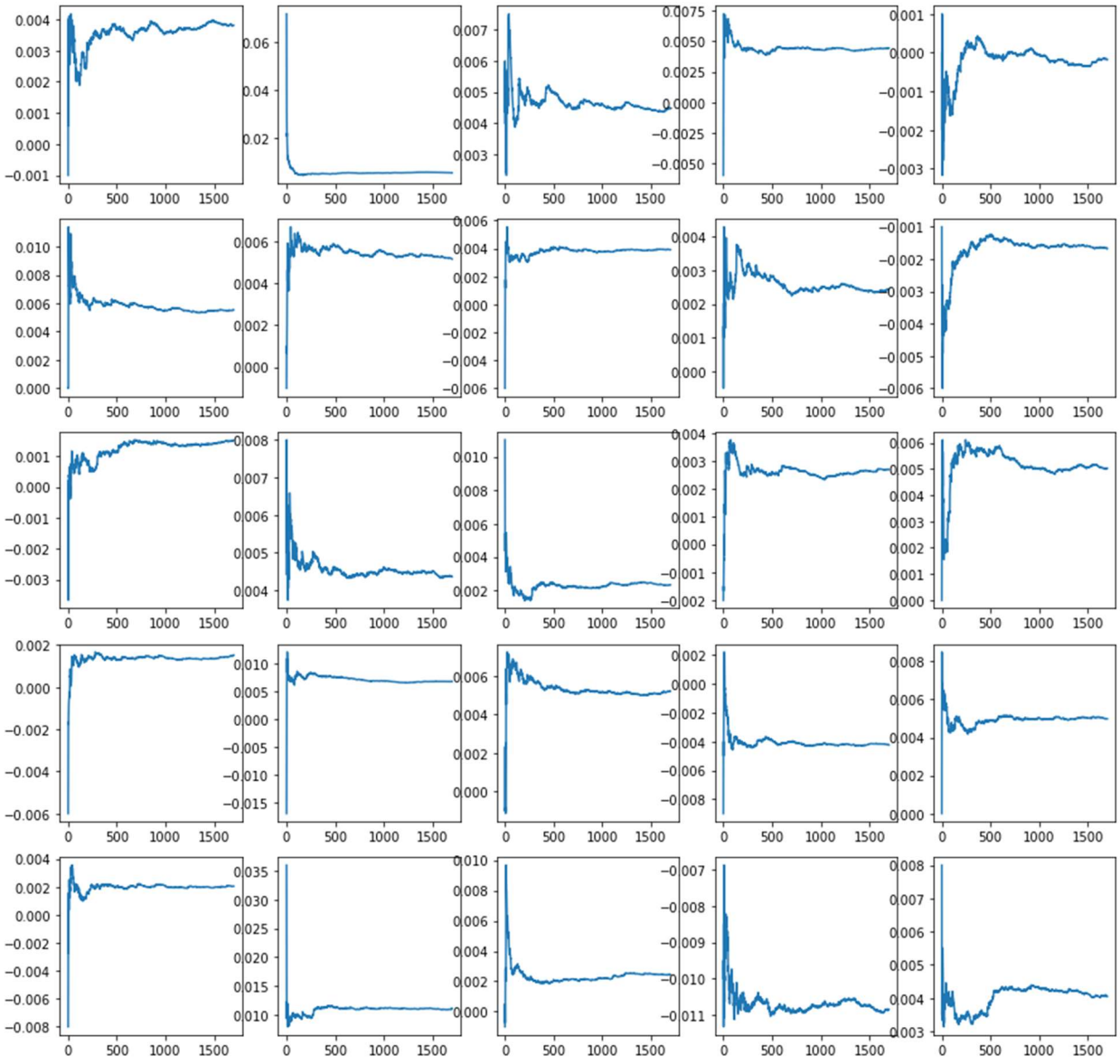

Figure 3.9: Convergence of marginals for the Noisy Truncated Monte Carlo method.

For the Noisy G-Shapley method, by making use of the relevant python statement *convergence_plots(dshap.marginals_g)*, a number of graphs with all marginals is being created as depicted in Figure 3.10. Again, it can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the Noisy G-Shapley method for the specific use case.

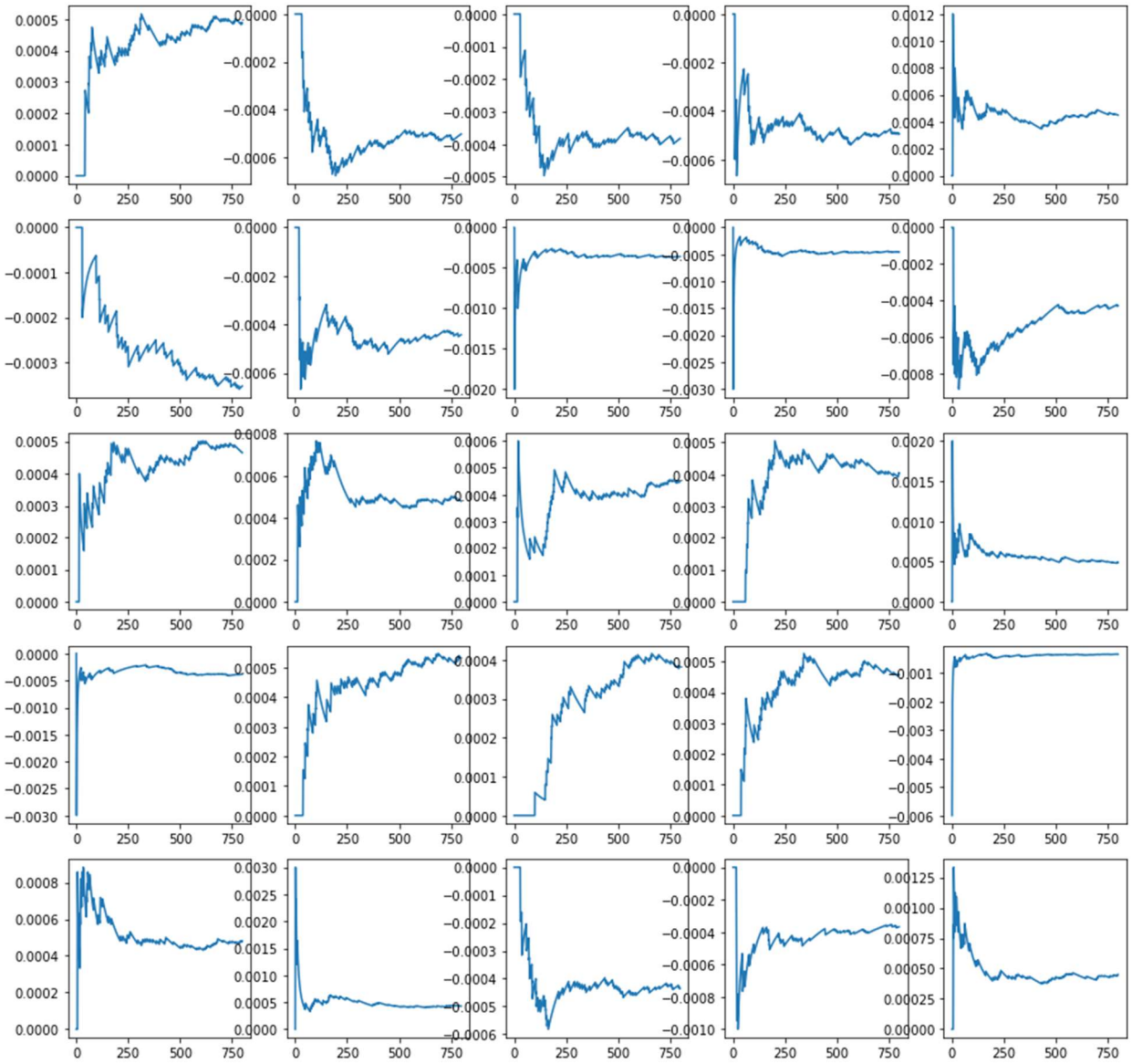

Figure 3.10: Convergence of marginals for the Noisy G-Shapley method.

The performance of the studied Noisy Data Shapley methods is demonstrated in Figure 3.11 by making use of the relevant python statement *dshap.performance_plots([dshap.vals_tmc, dshap.vals_g, dshap.vals_loo], num_plot_markers=20, sources=dshap.sources)*. It is evident that both Noisy TMC-Shapley and G-Shapley methods outperform the Noisy LOO method as well as the Noisy Random method, since the accuracy is reduced faster when bigger fractions of trained data are being removed.

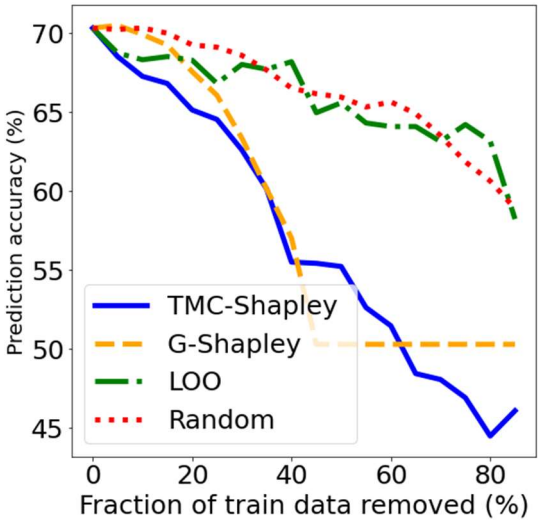

Figure 3.11: Data Shapley performance plots for the Noisy methods TMC-Shapley, G-Shapley, LOO and Random.

For the Noisy TMC method as well as the Noisy G-Shapley method, can be noted that all the marginals converge to very small (close to zero) values for i=1/4, 1/2, 2, 4, 8 and 16, which proves the convergence of these noisy methods.

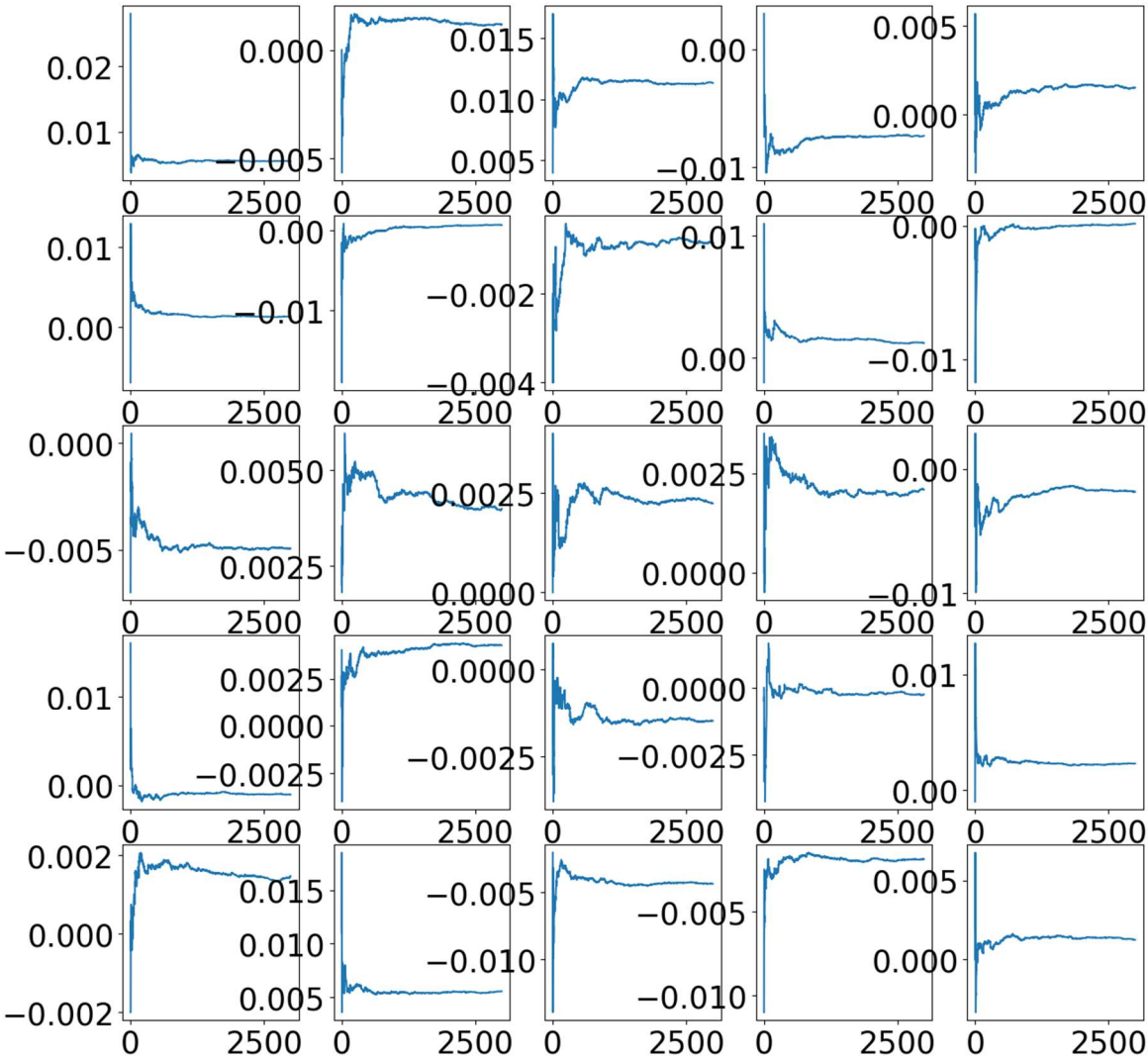

Figure 3.12: Convergence of marginals for the Noisy TMC-Shapley method (i=1/4).

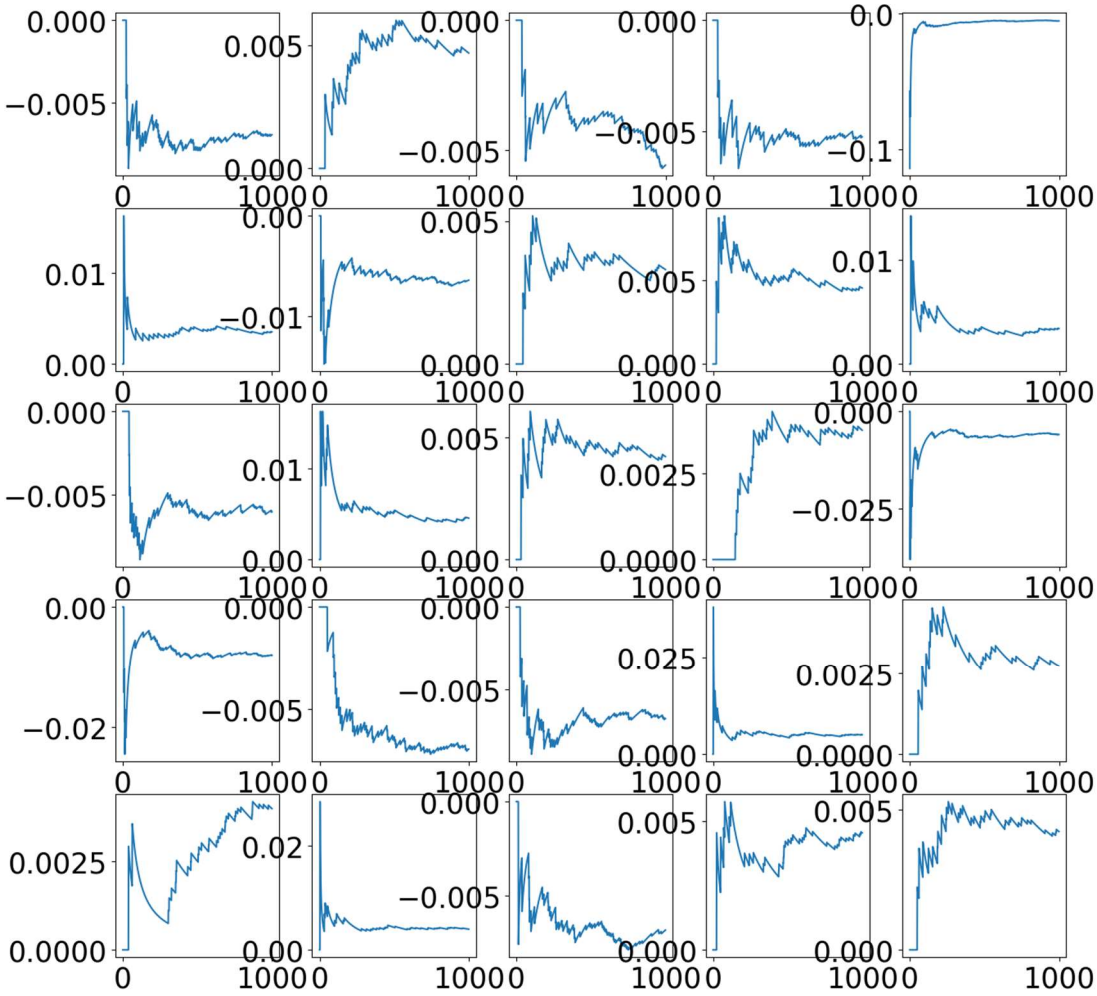

Figure 3.13: Convergence of marginals for the Noisy G-Shapley method(i=1/4).

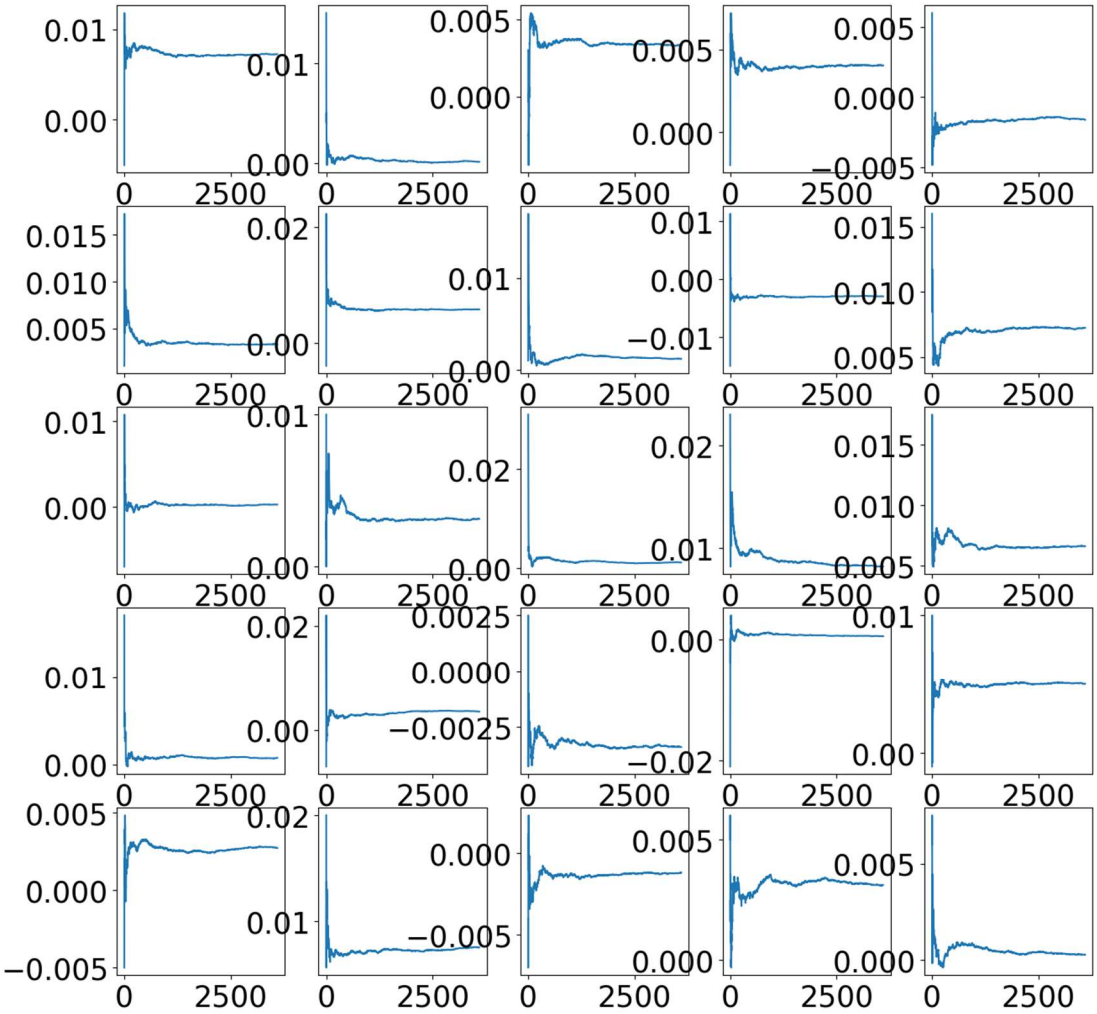

Figure 3.14: Convergence of marginals for the Noisy TMC-Shapley method (i=1/2).

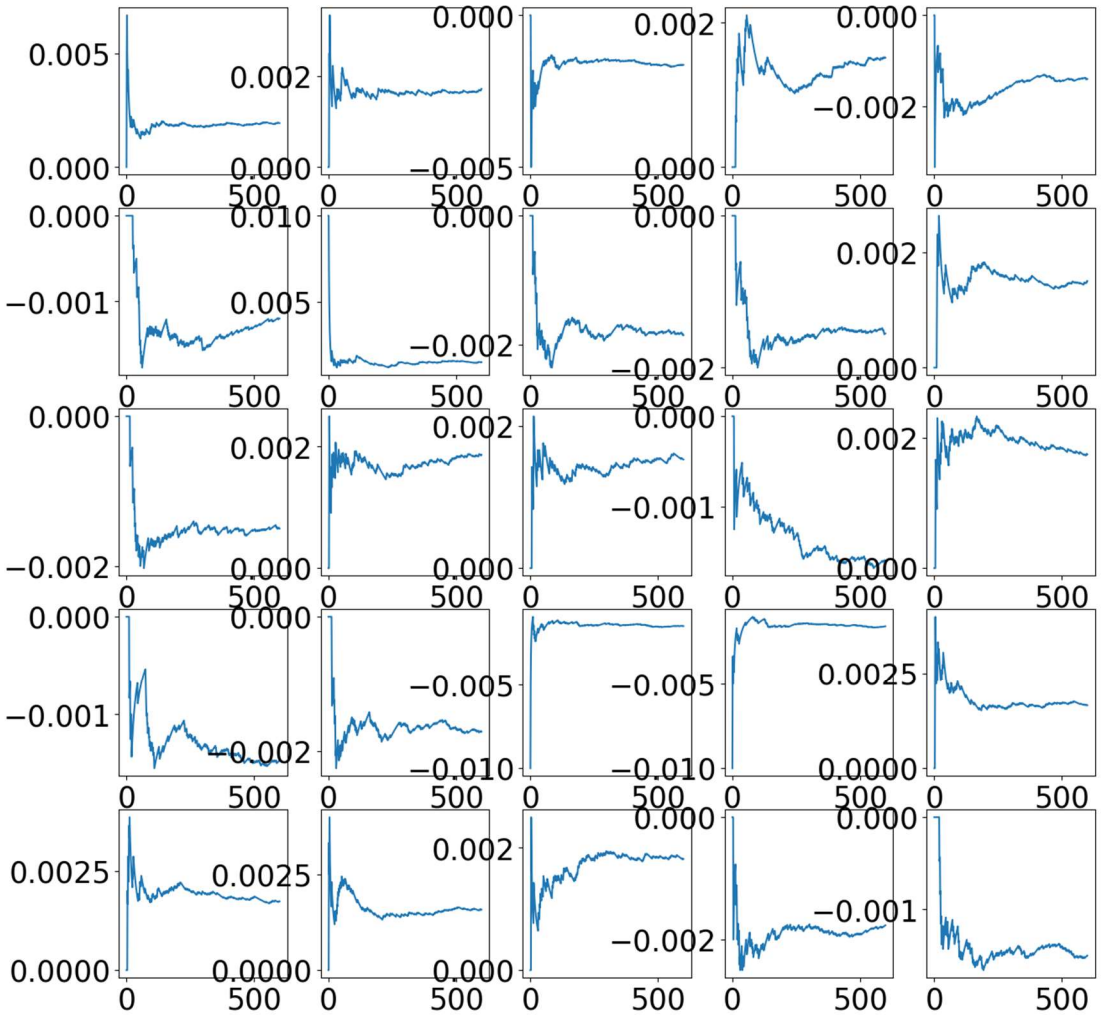

Figure 3.15: Convergence of marginals for the Noisy G-Shapley method(i=1/2).

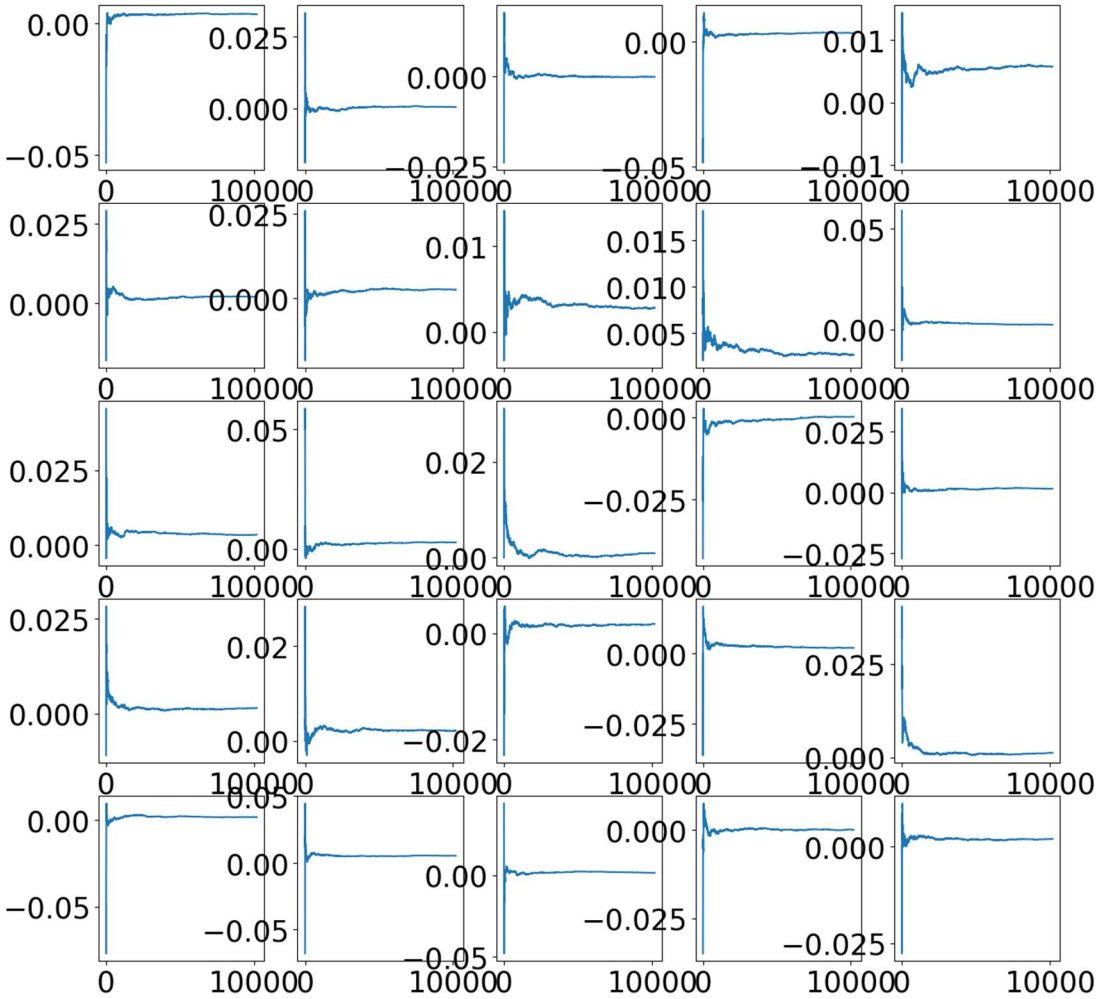

Figure 3.16: Convergence of marginals for the Noisy TMC-Shapley method (i=2).

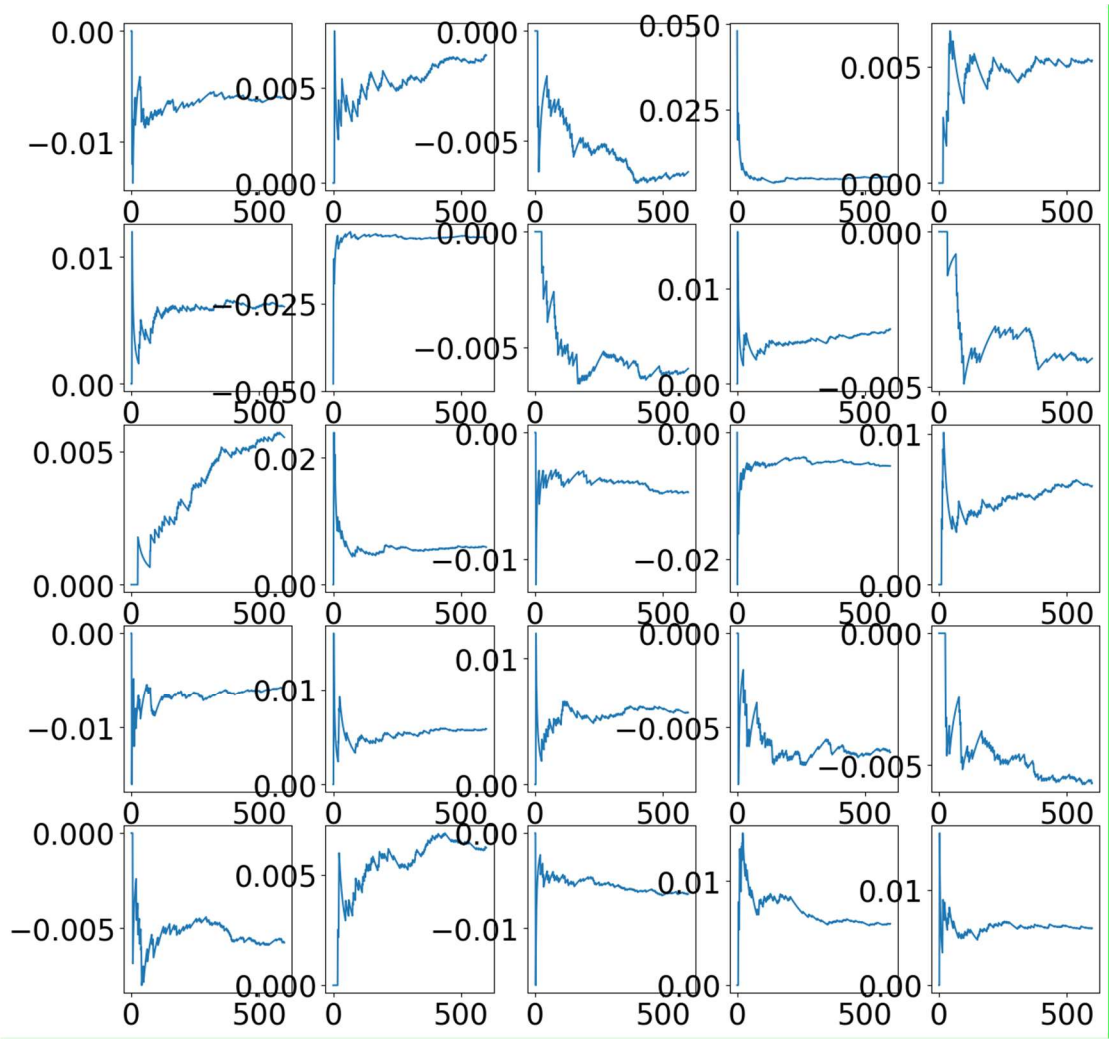

Figure 3.17: Convergence of marginals for the Noisy G-Shapley method (i=2).

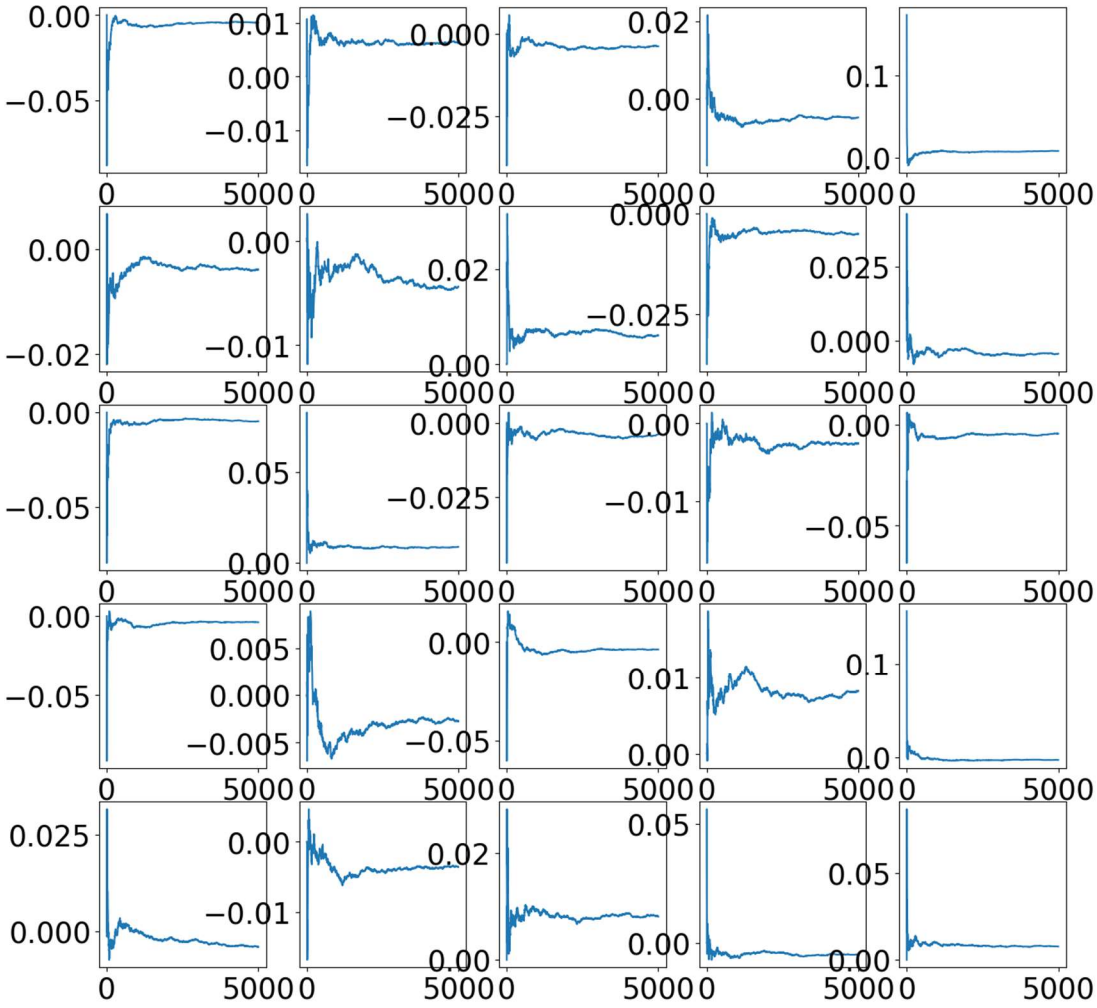

Figure 3.18: Convergence of marginals for the Noisy TMC-Shapley method (i=4).

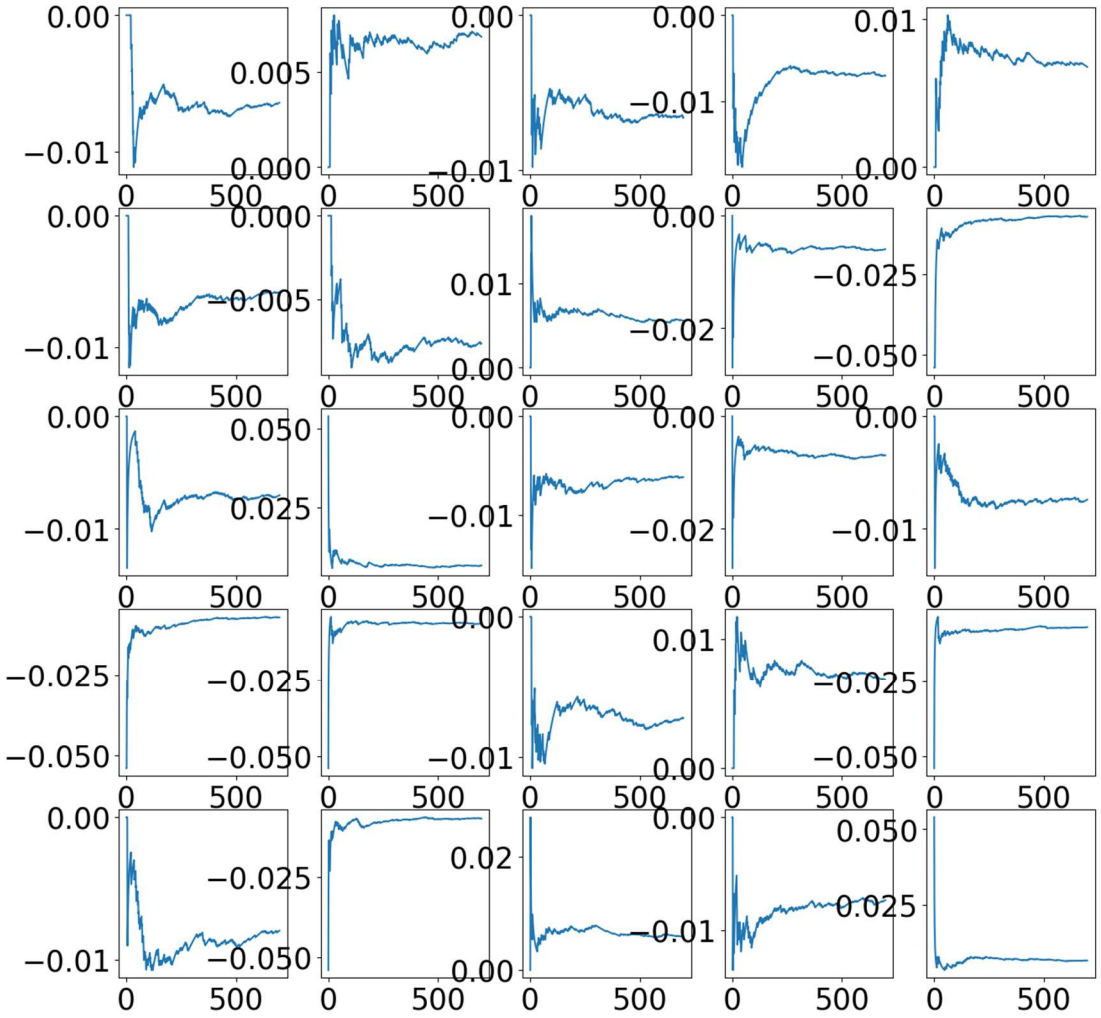

Figure 3.19: Convergence of marginals for the Noisy G-Shapley method (i=4).

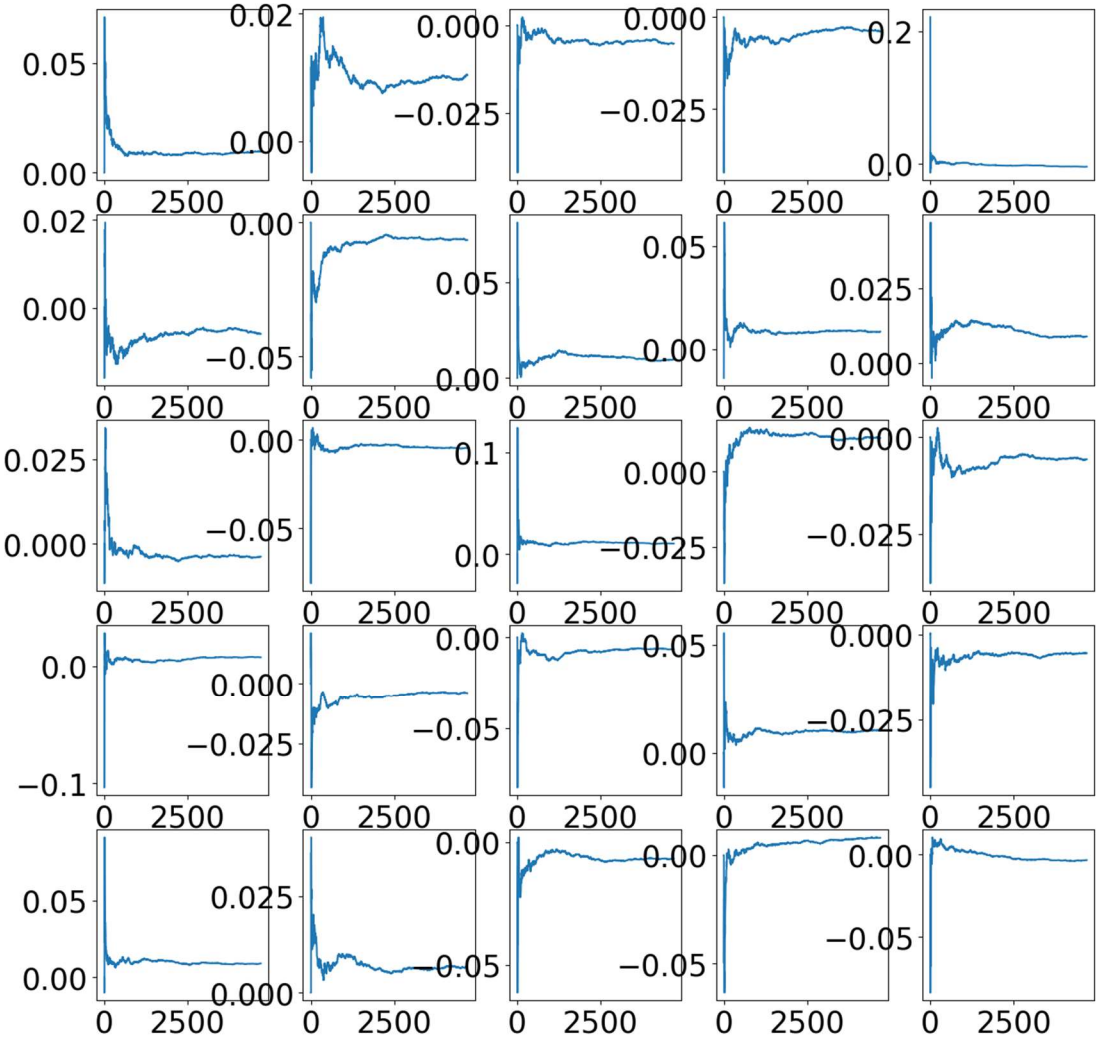

Figure 3.20: Convergence of marginals for the Noisy TMC-Shapley method (i=8).

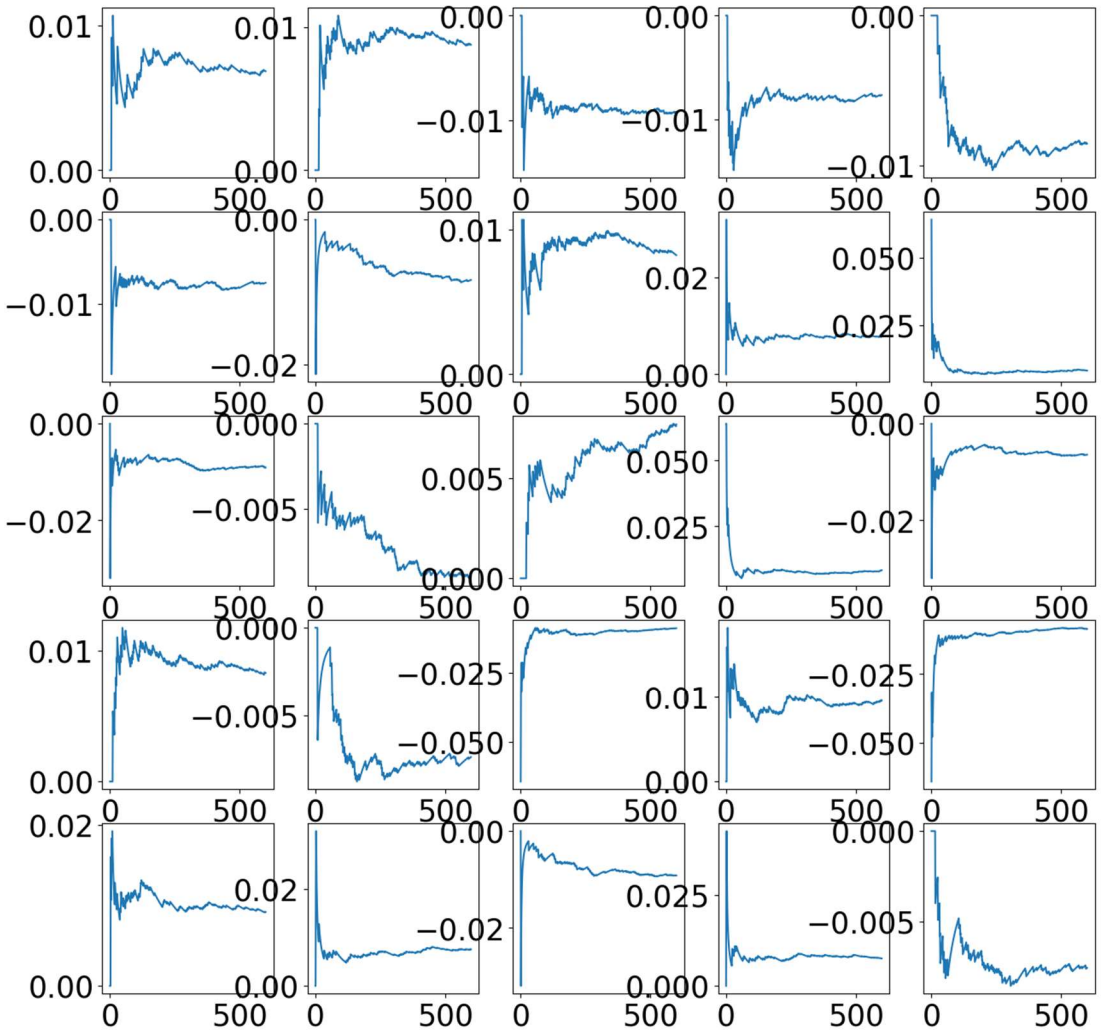

Figure 3.21: Convergence of marginals for the Noisy G-Shapley method (i=8).

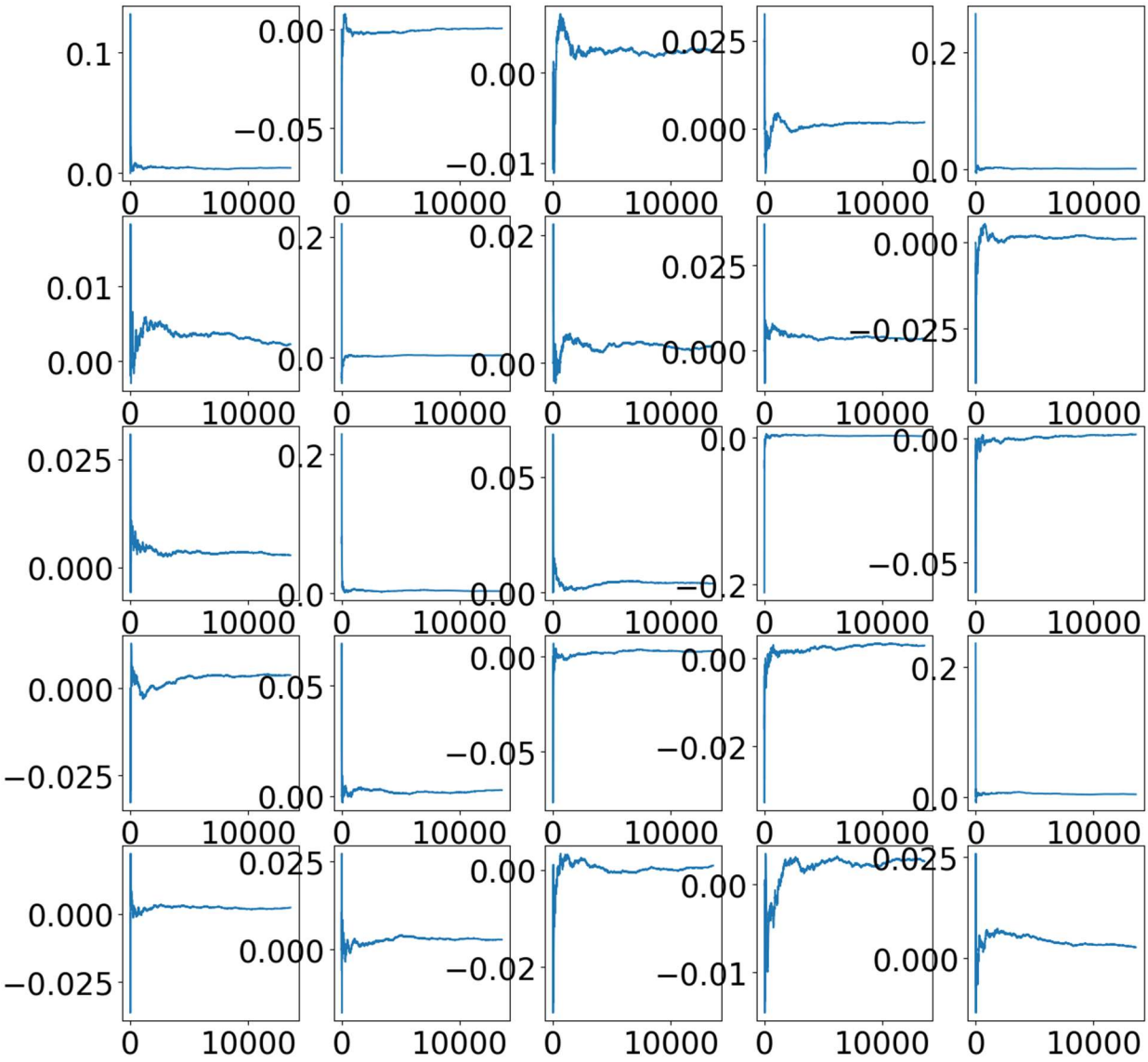

Figure 3.22: Convergence of marginals for the Noisy TMC-Shapley method (i=16).

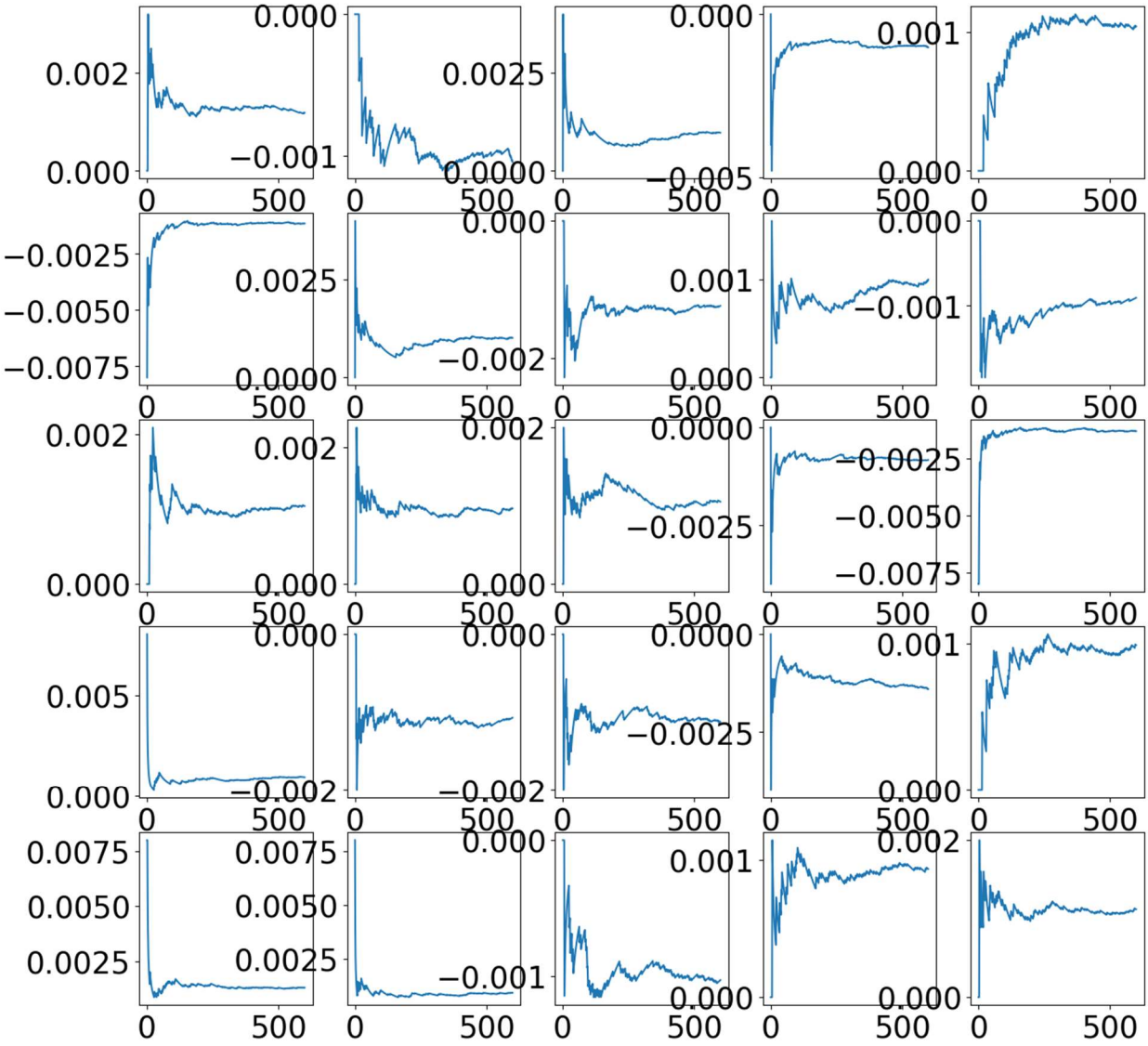

Figure 3.23: Convergence of marginals for the Noisy G-Shapley method (i=16).

The performance of the studied Noisy Data Shapley methods is demonstrated in Figure 3.24 through Figure 3.29 i=1/4, 1/2, 2, 4, 8 and 16. It is evident that as Laplacian noise increases the Noisy TMC-Shapley and G-Shapley methods as well as the Noisy LOO method have identical performance as the Noisy Random method, since the accuracy is reduced even faster when bigger fractions of trained data are being removed.

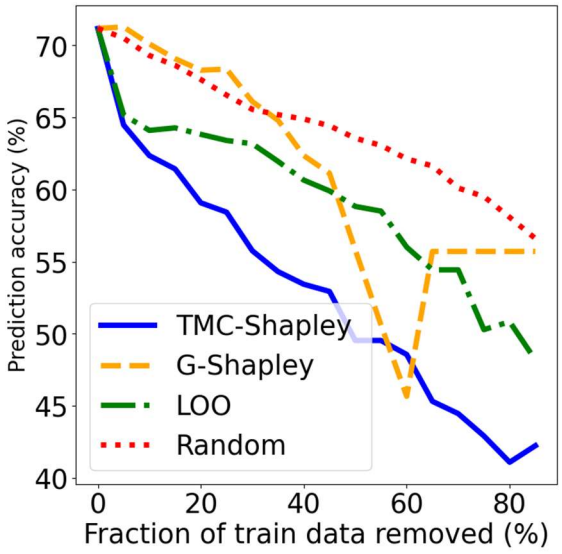

Figure 3.24: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=1/4).

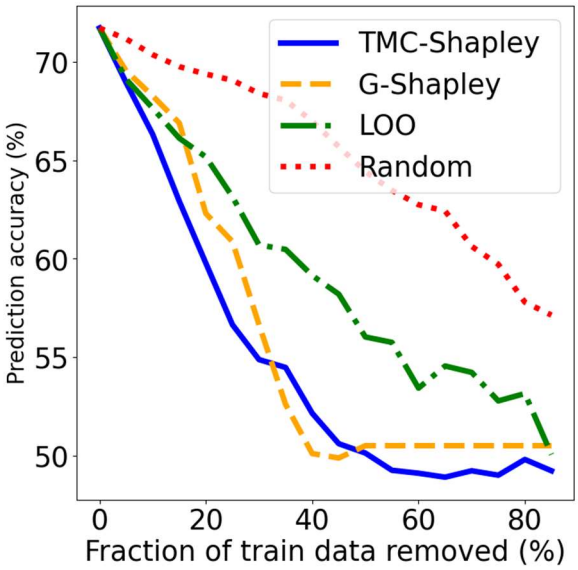

Figure 3.25: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=1/2).

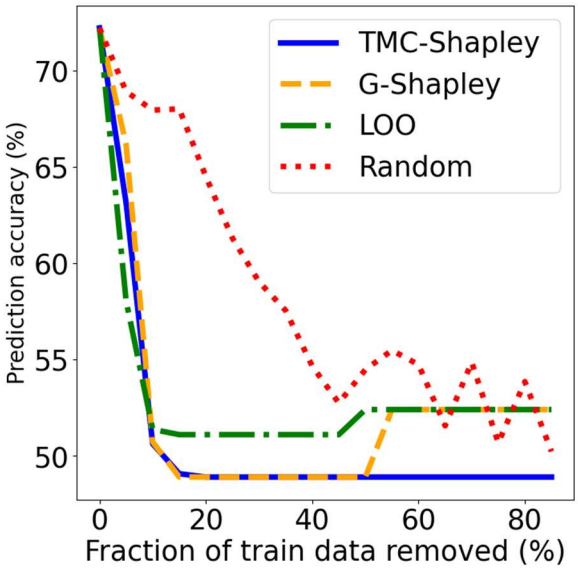

Figure 3.26: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=2).

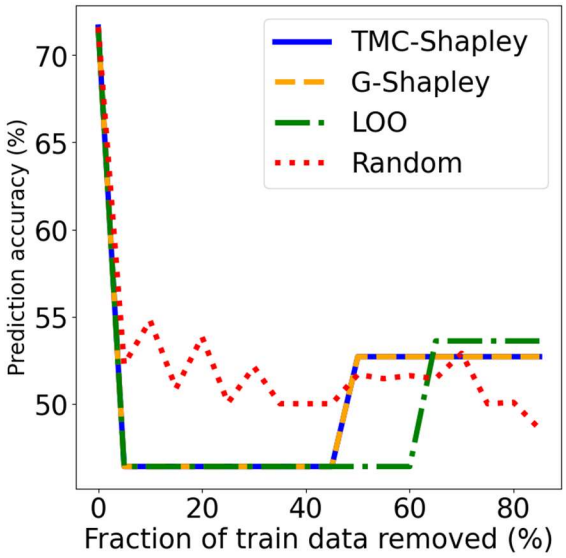

Figure 3.27: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=4).

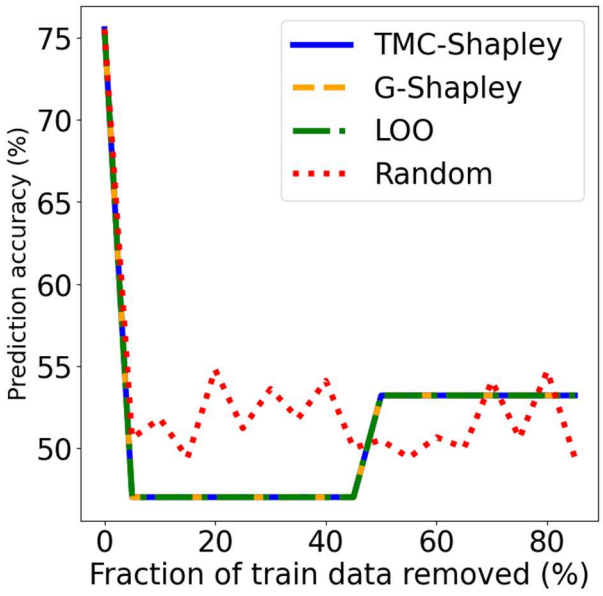

Figure 3.28: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=8).

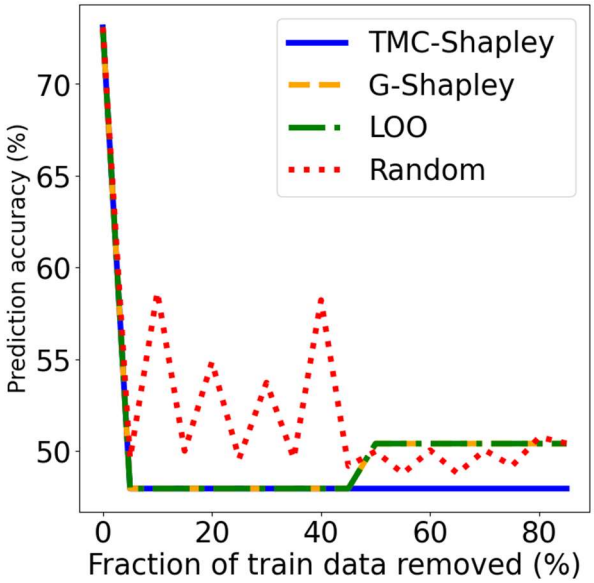

Figure 3.29: Data Shapley performance plots for methods TMC-Shapley, G-Shapley, LOO and Random (i=16).

4 CONCLUSIONS

4.1 Summary and conclusion

This study first presents the theoretical foundation of the Shapley value calculations, including mathematical analysis and the relative practical research evidence. Calculation methods, such as the TMC-Shapley, the G-Shapley and the Group-Shapley algorithms are studied and analyzed. The main components of a data valuation process are the training data set, the calculation algorithm itself, and a metric that defines performance (Ghorbani et al., 2019). In the context of the current work, two main topics related to fair data valuation are studied, the definition of a measure that is suitable to track the fair value of each (x_i, y_i) for a Calculation Algorithm (CA) in respect to a defined performance metric V, and the efficient calculation of the above fair data point value.

Differential Privacy (DP) is a relatively new research field, that aims to respond to information privacy challenges and enable Organizations and Companies to effectively manage private and sensitive information included in the managed data. DP was established as a research domain in 2006 by the seminal work of Dwork (2006), but has become popular during the last years, among others because it provides the so-called privacy guarantees as part of security frameworks and their implementation. Introduction of noise is of key importance in the DP algorithms and the way it is introduced defines significantly how the DP mechanism works. There are three main noise types usually applied in DP algorithms, namely the Laplace, the exponential, and the Gaussian noise types. Two different data processing models exist in the framework of differential privacy, namely the interactive or on-line query mode in which the data requester can access the data through an interface provided by the owner of the data, and the non-interactive model or offline query model in which the data requester can directly access only sanitized data sets as they are released by the data owner (Dwork et al., 2006; Xiong et al., 2014). To demonstrate how differential privacy with Laplacian noise is applied to an algorithm, a python program has been developed. Using that program, the way an ε-differentially private algorithm works is demonstrated through graphical representations of aggregated statistics over a potential numeric data set which might contain sensitive or private information.

In algorithms such as the truncated Monte Carlo Shapley algorithm, the privacy of the data is under risk during the Shapley value computations, since to compute the Shapley value for a point *i*, requires all other data points to be processed, putting thus at risk their privacy. By design, DP protects data privacy and acts proactively to avoid data leaks. It thus provides a privacy guarantee for every data point of the data set, by making sure that the Shapley value is not statistically sensitive to the addition or removal of individual data points. Under that perspective, the Shapley value is assumed to be a function and its sensitivity (and consequently the noise added by any mechanism, e.g. Laplacian) can be calculated. Of course, transforming a Data Shapley calculation to differential private, comes always with the challenge to achieve in parallel the highest possible Shapley calculation accuracy.

As part of the current study, an experimental implementation has been developed using Python programming language. In particular, Jupyter notebooks have been developed for executing and demonstrating three Data Shapley algorithms (Truncated Monte Carlo Shapley, Gradient Shapley, Leave One Out) without differential privacy, while at the same time a variation of the algorithms which embeds differential privacy with Laplacian DP noise in the Data Shapley calculations, has been implemented.

Specifically, in the case study proposed in this thesis, a classification problem with logistic regression has been designed, implemented, analyzed and presented. The code of the problem includes set up activities, data set creation, execution of three data Shapley algorithms with/ without embedded differential data privacy noise and graphical presentation of the results obtained. To prove convergence of the applied algorithms, the proposed implementation takes care of plotting the marginals. For the Truncated Monte Carlo method, a number of graphs with all marginals is being created. It can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the Noisy TMC method for the specific use case. For the Noisy G-Shapley method, a number of graphs with all marginals is being created. Again, it can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the G-Shapley method for the specific use case.

The performance of the studied Data Shapley methods is also demonstrated. It is evident that both TMC-Shapley and G-Shapley methods outperform the LOO method as well as the Random method, since the accuracy is reduced faster when bigger fractions of trained data are being removed.

To prove convergence of the applied algorithms, the proposed implementation takes care of plotting the marginals. For the Noisy Truncated Monte Carlo method, a number of graphs with all marginals is being created. It can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the Differential Private TMC method for the specific use case. For the Differential Private G-Shapley method, a number of graphs with all marginals is being created. Again, it can be easily noted that the marginals converge to very small (close to zero) values, which proves the convergence of the Noisy G-Shapley method for the specific use case. The performance of the studied Noisy Data Shapley methods is also demonstrated. It is evident that both Noisy TMC-Shapley and G-Shapley methods outperform the Noisy LOO method as well as the Noisy Random method, since the accuracy is reduced faster when bigger fractions of trained data are being removed.

In summary, this work aims to contribute to the following areas:

- Provision of an efficient formulation of the important problem of calculating the fair value of data sets, known as Data Shapley valuation problem, by taking advantage of machine learning techniques.
- Review of existing empirical studies that prove the efficiencies and all desirable properties of the Data Shapley valuation method.
- Introduction of a Data Shapley calculation algorithm variations that are suitable to address the problem under study.
- Application of sensitivity and noise to ensure data privacy mechanism as well as to measure and analyze data Shapley value in the context of differential privacy.
- Implementation of a representative use case where data Shapley value calculations, combined with differential privacy guarantees are applied in selected data sets.

4.2 Dissertation limitations

This study is primarily addressed to researchers who deal with the important thematic fields of Data Shapley valuations and differential privacy. It presents a broad bibliographic analysis of this scientific domain, including empirical studies that have been conducted to support and substantiate

findings on the application and impact of differential private Shapley calculations. Additionally, the current research went beyond the limits of simple literature review and included quantitative research and analysis to evaluate how the two scientific domains could be combined, with the aid of appropriate programming tools. Overall, therefore, one can easily understand that the limitations set in the context of this study are those that may be set by the used tools of literature search, and of conducting quantitative study and analysis.

Especially for the quantitative analysis, it is obvious that the actual conditions of the experimentation, i.e. the selected data set, the parameters use for experiments configuration, and the tested algorithms themselves, put specific limitations on the breadth and depth of the research as well as to the interpretation of the produced results. One of the most significant limitations of the current study is the connection of the data sets used for experimentation with actual and meaningful use cases, and the ability to translate the scientific results to socially understandable and valuable metrics. In other words, although the current study is contributing to the enrichment of the data Shapley valuation approaches with the embedded differential privacy, its contribution in the provision of meaningful and actionable insights and thus its impact in the social society, remains limited. Another significant limitation, which is connected to the previous one, is the computational effort and resources required to handle especially large data sets that in practice connect with real life applications. Additional work should be done at algorithm level to remove this limitation and expand the usage and applicability of the current study.

4.3 Future work

The current study has captured a number of aspects around the calculation of the Data Shapley value and differential privacy, but there are still many open questions for further research. Regarding the data Shapley calculations, some of the future work could be the following:

Data Shapley calculations satisfy three important properties for the fair valuation of data. Undoubtedly, there are many different Machine Learning settings where these properties are desirable or not, while there are other settings where additional properties might become important. Additional work will be needed to identify what are the desirable properties and under what scenarios the calculations of Data Shapley value is more effective.

- Combinations of fair data value calculation with other important areas such as data privacy, and personal association are also candidates for future research directions.
- Applied learning metrics and functions in the framework of Data Shapley value calculations need to also be further studied and are proposed for future work.

At the same time differential privacy has become the standard approach to guarantee privacy and is one of the most popular research topics in domains where the sensitivity of exchanged information is high. Important work has been conducted, but research efforts should further evolve and could be directed in the following directions:

- Since differential privacy is a highly sensitive problem, reduction of the sensitivity to further enhance useability, require on-going research.
- Privacy guarantee in cases of online data transfer is another significant question which worths additional study.
- Since computational complexity of differential privacy algorithms remains high in many cases, improvement of algorithm efficiency is another area where further research should be done.
- Distributed differential privacy, which means sharing data among multiple parties and at the same time guaranteeing privacy, is another research area that deserves additional attention (Liu et al., 2013; Friedman et al., 2014).
- Designing optimal privacy budgeting strategies in structures such as trees, is a great challenge that would need significant future work (Cormode et al., 2012).
- Balancing of the noise and nonuniformity errors, by choosing the optimal partition granularity, especially for geospatial data is also a challenge that requires future research (Qardaji et al., 2013a).

No need to explain, that future research efforts on the combined problem, i.e. Data Shapley valuations combined with Differential Privacy guarantees, is of great importance to further evolve the current research in both directions in a way that adds value in both directions.

References

Arrieta-Ibarra, I., Goff, L., Jimenez Hernandez, D., Lanier, J., & Weyl, E. (2017). Should we treat data as labor? Moving beyond "free". *AEA Papers and Proceedings, 108*, 38-42. https://doi.org/10.1257/pandp.20181003

Bagdasaryan, E., & Shmatikov, V. (2019). Differential privacy has disparate impact on model accuracy. *Advances in Neural Information Processing Systems, 32,* 1-10. https://doi.org/10.48550/arXiv.1905.12101

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., & Popp, J. (2013). Sample size planning for classification models. *Analytica Chimica Acta, 760*, 25-33. https://rb.gy/knslaq

Bousquet, O., & Elisseeff, A. (2002). Stability and Generalization. *Journal of Machine Learning Research, 2*, 499-526. https://cutt.ly/iN4h3pt

Castro, J., Gomez, D., & Tejada, J. (2009a). Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research, 36*(5), 1726-1730. https://doi.org/10.1016/j.cor.2008.04.004

Chen, J., Song, L., Wainwright, M., & Jordan, M. (2018). L shapley and c-shapley: Efficient model interpretation for structured data. *arXiv*, 1808.02610. https://rb.gy/qcdyk6

Chen, J., & Xiao, K. (2010). BISC: a bitmap itemset support counting approach for efficient frequent itemset mining. *ACM Transactions on Knowledge Discovery from Data, 4*(3), 1-37. https://doi.org/10.1145/1839490.1839493

Cohen, S., Dror, G., & Ruppin, E. (2007). Feature selection via coalitional game theory. *Neural Computation, 19*(7), 1939-1961. https://rb.gy/w2xzxl

Cook, R. (1977). Detection of influential observation in linear regression. *Technometrics, 19*(1), 15-18. https://cutt.ly/hN4js4Q

Cook, R., & Weisberg, S (1982). *Residuals and influence in regression*. New York: Chapman and Hall.

Cormode, G., Procopiuc, C., Srivastava, D., Shen, E., & Yu, T. (2012). Differentially private spatial decompositions. Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE '12), pp. 20-31. https://doi.org/10.1109/ICDE.2012.16

Datta, A., Sen, S., & Zick, Y. (2016). Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. *Security and Privacy (SP), 2016 IEEE Symposium*, 598-617. https://rb.gy/7vdnoz

Dong, J., Roth, A., & Su, W. (2022). Authors' Reply to the Discussion of 'Gaussian Differential Privacy'. *Journal of the Royal Statistical Society Series B: Statistical Methodology, 84*(1), 50-54. https://doi.org/10.1111/rssb.12463

Dwork, C. (2006). Differential privacy. *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP)* (2), 1-12. https://rb.gy/gvogzh

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., & Naor, M. (2006). Our Data, Ourselves: Privacy Via Distributed Noise Generation, in: Vaudenay, S. (eds) Advances in Cryptology - EUROCRYPT 2006. Lecture Notes in Computer Science, Vol 4004. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11761679_29

Dwork, C., McSherry, F., Nissim, K., & Smith, A. (2006b). Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds) Theory of Cryptography. TCC 2006. Lecture Notes in Computer Science, vol 3876. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11681878_14

Dwork C. (2011a). A firm foundation for private data analysis. *Communications of the ACM, 54*(1), 86-95. https://doi.org/10.1145/1866739.1866758

Dwork, C. (2011b). Differential Privacy, in: Van Tilborg, H.C.A., Jajodia, S. (eds), Encyclopedia of Cryptography and Security. Springer, Boston, MA. https://doi.org/10.1007/978-1-4419-5906-5_752

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science, 9*(3-4), 211-407. https://rb.gy/55urlt

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the International Conference on Knowledge Discovery & Data Mining* (KDD '96), 226-231. https://shorturl.at/jmW09

Fan, L., Xiong, L., & Sunderam, V. (2013). Differentially Private Multi-dimensional Time Series Release for Traffic Monitoring. In Wang, L., Shafiq, B. (eds) Data and Applications Security and Privacy XXVII, DBSec 2013. Lecture Notes in Computer Science, vol. 7964. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-39256-6_3

Fan, L., Xiong, L., & Sunderam, V. (2013b). FAST: differentially private real-time aggregate monitor with filtering and adaptive sampling. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 1065-1068. Association for Computing Machinery, New York, USA. https://doi.org/10.1145/2463676.2465253

Fan, L., & Xiong, L. (2014). An Adaptive Approach to Real-Time Aggregate Monitoring With Differential Privacy. *IEEE Transactions on Knowledge and Data Engineering, 26*(9), 2094-2106. https://doi.org/10.1109/TKDE.2013.96

Fan, L., Bonomi, L., Xiong, L., & Sunderam, V. (2014). Monitoring web browsing behavior with differential privacy. *Proceedings of the 23rd International Conference on World Wide Web (WWW '14), Seoul, Republic of Korea, pp. 177-187.* https://doi.org/10.1145/2566486.2568038

Fatima, S., Wooldridge, M., & Jennings, N. (2008). A linear approximation method for the shapley value. *Artificial Intelligence, 172*(14), 1673-1699. https://rb.gy/djt5aq

Friedman, A., Sharfman, I., Keren, D., & Schuster, A. (2014). Privacy-preserving distributed stream monitoring. *Network and Distributed System Security Symposium*, San Diego, California, USA, pp. 1-12. https://shorturl.at/npJM9

Fryer, D., Strümke, I., & Nguyen, H. (2021). Shapley Values for Feature Selection: the Good, the Bad, and the Axioms. *arXiv: 2102.10936*. https://doi.org/10.48550/arXiv.2102.10936

Ganev, G., Oprisanu, B., & De Cristofaro, E. (2021). Robin hood and matthew effects–differential privacy has disparate impact on synthetic data. *arXiv preprint arXiv:2109.11429*. https://doi.org/10.48550/arXiv.2109.11429

GDPR (2018). *Guide to the General Data Protection Regulation*. Information Commissioner's Office. https://cutt.ly/3N4j3XE

Ghorbani, A., Abid, A., & Zou, J. (2017). Interpretation of neural networks is fragile. arXiv: 1710.10547. https://rb.gy/njgyk8

Ghorbani, A., & Zou, J. (2019). Data Shapley: Equitable Valuation of Data for Machine Learning. *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. https://cutt.ly/wN4kQ52

Gul, F. (1989). Bargaining foundations of shapley value. *Econometrica: Journal of the Econometric Society, 57*(1), 81-95. https://rb.gy/deergi

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning* (2nd ed.). New York: Springer series in statistics. https://rb.gy/v12kk7

Hay, M., Li, C., Miklau, G., & Jensen, D. (2009). Accurate Estimation of the Degree Distribution of Private Networks. *2009 Ninth IEEE International Conference on Data Mining*, Miami Beach, FL, USA, 169-178, https://doi.org/10.1109/ICDM.2009.11

Hay, M., Rastogi, V., Miklau, G., & Suciu, D. (2010). Boosting the accuracy of differentially private histograms through consistency. *Proceedings of the VLDB Endowment, 3*(1-2), 1021-1032. https://doi.org/10.14778/1920841.1920970

Ho, S.-S., & Ruan, S. (2011). Differential privacy for location pattern mining. *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS* (SPRINGL '11), Chicago, Ill, USA, 17-24. https://doi.org/10.1145/2071880.2071884

Ho, S.-S., & Ruan, S. (2013). Preserving privacy for interesting location pattern mining from trajectory data. *Transactions on Data Privacy, 6*(1), 87-106. https://shorturl.at/EO257

Hsu, J., Gaboardi, M., Haeberlen, A., Khanna, S., Narayan, A., Pierce, B., & Roth, A (2014). Differential privacy: an economic method for choosing epsilon. *Proceedings of the IEEE 27th Computer Security Foundations Symposium (CSF '14),* Austria, IEEE, 398-410. https://rb.gy/r1s3b6

Inan, A., Kantarcioglu, M., Ghinita, G., & Bertino, E. (2012). A hybrid approach to private record matching. *IEEE Transactions on Dependable and Secure Computing, 9*(5), 684-698. https://doi.org/10.1109/TDSC.2012.46

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering, 82*(1), 35-45. https://doi.org/10.1115/1.3662552

Koh, P., & Liang, P. (2017). Understanding black-box predictions via influence functions. arXiv: 1703.04730. https://rb.gy/liyxlj

Kumar, I., Venkatasubramanian, S., Scheidegger, C., & Friedler, S. (2020). Problems with Shapley-value-based explanations as feature importance measures. *Proceedings of the 37th International Conference on Machine Learning, PMLR 119*, 5491-5500. https://shorturl.at/hqSW4

Lee, J., & Clifton, C. (2014). Top-k frequent itemsets via differentially private FP-tree. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), New York, USA, pp. 931-940. https://doi.org/10.1145/2623330.2623723

Li, N., Qardaji, W., Su, D., & Cao J. (2012). Privbasis: frequent itemset mining with differential privacy. *Proceedings of the VLDB Endowment, 5*(11), 1340-1351. https://doi.org/10.14778/2350229.2350251

Li, Y., Wen W., Xie G. Q. (2012). Survey of research on differential privacy, Application Research of Computers, 299, pp. 3201-3211

Liu, J., Xiong, L., Luo, J., & Huang, J. (2013). Privacy preserving distributed DBSCAN clustering. *Transactions on Data Privacy, 6,* 69-85. https://shorturl.at/kqPS2

Luca, B., & Li, X. (2013). Mining frequent patterns with differential privacy. *Proceedings of the VLDB Endowment, 6*(12), 1422-1427. https://doi.org/10.14778/2536274.2536329

Luca, B., & Li, X. (2013b). A two-phase algorithm for mining sequential patterns with differential privacy. *CIKM '13: Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 269–278. https://doi.org/10.1145/2505515.2505553

Lundberg, S., & Lee, S. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems,* 4765-4774. https://rb.gy/ujcvkp

Lundberg, S., Erion, G., & Lee, S. (2018). Consistent individualized feature attribution for tree ensembles. arXiv: 1802.03888. https://rb.gy/93gaue

Mahadevan, P., Krioukov, D., Fall, K., & Vahdat, A. (2006). Systematic topology analysis and generation using degree correlations. *ACM SIGCOMM Computer Communication Review, 36*(4), 135–146. https://doi.org/10.1145/1151659.1159930

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., & Van Der Maaten, L. (2018). Exploring the limits of weakly supervised pretraining. arXiv: 1805.00932. https://rb.gy/6s4ohs

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., & Rogers, A (2013). Bounding the estimation error of samplingbased shapley value approximation. arXiv: 1306.4265. https://rb.gy/g6sm1f

Mann, I., & Shapley, L. (1962). *Values of large games. 6: Evaluating the electoral college exactly.* Santa Monica: Rand Corp. https://rb.gy/nth7kh

McSherry, F. (2010). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Communications of the ACM, 539*, 89-97. https://doi.org/10.1145/1810891.1810916

McSherry, F., & Talwar, K. (2007). Mechanism Design via Differential Privacy. *48th Annual IEEE Symposium on Foundations of Computer Science* (FOCS'07), Providence, RI, USA, 94-103. https://rb.gy/cuncun

Michalak, T., Aadithya, K., Szczepanski, P., Ravindran, B., & Jennings, N. (2013). Efficient computation of the shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research, 46,* 607-650. https://rb.gy/nbuokf

Milnor, J., & Shapley, L. (1978). Values of large games ii: Oceanic games. *Mathematics of operations research, 3*(4), 290-307. https://rb.gy/ehes1w

Nissim K., Raskhodnikova S., & Smith A (2007). Smooth sensitivity and sampling in private data analysis. *Proceedings of the 39th Annual ACM Symposium on Theory of Computing* (STOC ′07), 75-84. https://doi.org/10.1145/1250790.1250803

Nissim, K., et al. (2018). *Differential Privacy: A Primer for a Non-technical Audience.* https://rb.gy/xiycvq

Posner, E., & Weyl, E. (2018*). Radical Markets: Uprooting Capitalism and Democracy for a Just Society*. Chicago: Princeton University Press.

Qardaji, W., Yang, W., & Li, N. (2013a). Differentially private grids for geospatial data. Proceedings of the 29th International Conference on Data Engineering (ICDE ′13), Brisbane, Australia, pp. 757-768. https://doi.org/10.1109/ICDE.2013.6544872

Qardaji, W., Yang, W., & Li, N. (2013b). Understanding hierarchical methods for differentially private histograms. Proceedings of the VLDB Endowment, 614, pp. 1954-1965. https://doi.org/10.14778/2556549.2556576

Rastogi, V., & Nath, S. (2010). Differentially private aggregation of distributed time-series with transformation and encryption. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Indianapolis, 735-746, https://doi.org/10.1145/1807167.1807247

Sala, A., Zhao, X., Wilson, C., Zheng, H., & Zhao, B. Y. (2011). Sharing graphs using differentially private graph models. *Proceedings of the ACM SIGCOMM Internet Measurement Conference* (IMC ′11), Berlin, Germany, 81-97. https://doi.org/10.1145/2068816.2068825

Shapley, L. (1953). A value for n-person games. *Contributions to the Theory of Games, 2*(28), 307-317. https://rb.gy/r0isrg

Shapley, L., Roth, A., et al. (1988). *The Shapley value: essays in honor of Lloyd S.* Cambridge: Cambridge University Press.

Shen, E., & Yu, T. (2013). Mining frequent graph patterns with differential privacy. *Proceedings of the 19th ACM SIGKDD International Conference,* Chicago, Ill, USA, 545-553. https://doi.org/10.1145/2487575.2487601

Strumbelj, E., & Kononenko, I. (2010). An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research, 11*, 1-18. https://rb.gy/ujvfwe

Wang, J., Liu, S., & Li, Y. (2015). A Review of Differential Privacy in Individual Data Release. *International Journal of Distributed Sensor Networks, 11*(10), 259682-2596. https://rb.gy/bbbmbr

Wang, Y. & Wu, X. (2013). Preserving differential privacy in degree-correlation based graph generation. *Transactions on Data Privacy, 6*(2), 127-145. PMID: 24723987; PMCID: PMC3979555. https://shorturl.at/ipqW4

Wang, Y., Wu, X., & Wu, L. (2013). Differential Privacy Preserving Spectral Graph Analysis. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2013. Lecture Notes in Computer Science, vol 7819. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37456-2_28

Xiao, Q., Chen, R., & Tan, K. (2014). Differentially private network data release via structural inference. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14), New York, USA, pp. 911-920. https://doi.org/10.1145/2623330.2623642

Xiao, Y., Gardner, J., & Xiong, L. (2012). DPCube: Releasing Differentially Private Data Cubes for Health Information. IEEE 28th International Conference on Data Engineering, Arlington, VA, USA, pp. 1305-1308, https://doi.org/10.1109/ICDE.2012.135

Xiao, Y., Xiong, L., & Yuan, C. (2010). Differentially Private Data Release through Multidimensional Partitioning. In: Jonker, W., Petković, M. (eds) Secure Data Management. SDM 2010. Lecture Notes in Computer Science, Vol 6358. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-15546-8_11

Xiong, P., Zhu, T. Q., Wang, X. F (2014). A survey on differential privacy and applications. Chinese Journal of Computers, 371, pp. 101-122. DOI: 10.3724/SP.J.1016.2014.00101

Xu, J., Zhang, Z., Xiao, X., Yang, Y., Yu, G., & Winslett, M. (2013). Differentially private histogram publication. *The VLDB Journal, 226*, 797-822. https://doi.org/10.1007/s00778-013-0309-y

Zhang, X. J., Meng, X. F. (2014). Differential privacy in data publication and analysis. Chinese Journal of Computers, 374, pp. 927-949.