Local Differential Privacy Preservation via the Novel Encoding Method

Niu Zhang; Shunwei Wang; Tianchong Gao; Feng Li; Agnideven Palanisamy Sundar; Xukai Zou

***Abstract*—In recent years, the application of big data analysis has seen a significant increase across various fields with the aim of extracting useful information from massive data to enhance human life. However, the use of data may lead to the exposure of sensitive information. To address this pressing issue, many scholars have incorporated the latest protection methodologies, such as local differential privacy (LDP), into big data analysis. LDP preserves privacy with a strict theoretical guarantee but is blamed for high utility loss. For example, existing LDP methods usually convert the data into binary string and then adding noise. However, the weight of each bit unevenly distributes the noise, and eventually increases the utility lose. Inspired by this, the paper investigates the reduction of error through alternative numeral systems, leading to the proposition of a segmented-based LDP data preservation mechanism (LDPseg), where each bit flip has an equal impact on the outcome. Theoretical analysis reveals that under certain conditions, this mechanism maintains a lower error expectation compared to binary systems. Real-world experimental results indicate that the proposed method exhibits positive performance in machine learning.**

***Index Terms*—Data privacy protection, Machine learning, Local differential privacy, Segmentation method**

I. Introduction

In recent years, the continuous in-depth research and development of machine learning technologies have yielded great success in various fields. The use of machine learning allows for the provision of significantly more precise and diverse services. However, the user data used for machine learning may typically contain a considerable amount of privacy data, and directly using this data for training model poses a significant risk of privacy leakage [1]. Addressing this issue requires an effective privacy protection scheme. The mechanism based on Differential Privacy (DP) was first proposed and provides theoretical privacy guarantees to protect data privacy [2]. Against this background, global differential privacy (GDP) and local differential privacy (LDP) are proposed as two commonly used data privacy protection mechanisms [3].

Global Differential Privacy (GDP) involves the introduction of noise to the data set query output for personal privacy protection. To achieve this approach, a trusted data center is required, which can be challenging to establish [4]. Unlike GDP, LDP eliminates the need for a trusted data center by directly performing differential privacy operations on the user's data set. The data set is then transmitted to the data center for aggregation calculations. This method ensures that the data center cannot infer the original data set, thereby ensuring the privacy of the user's data [5].

In previous research, the authors proposed Modify Optimized Unary Encoding ($MOUE$) and Utility Enhancing Randomization (UER) methods to implement LDP [6]. Both methods convert the original data into binary form, and then add perturbation based on probabilities. The $MOUE$ method enhances the flexibility of binary string randomization, and UER maintains the utility during the randomization of long binary strings with high sensitivity. Nevertheless, when the data is converted into binary, the different weights of each bit may lead to diverse impacts on data utility. Such impacts may result in unnecessary noise, which can diminish the effectiveness of the machine learning models.

To address the above-mentioned issue, this paper proposes a modification to the existing encoding method to achieve a better balance between data privacy and the accuracy of the training model while enhancing the utility of data. Theoretical studies have shown that the closer each weight is to 1, the better the effect will be. Thus, a digital segmentation method with equal weight is introduced, which divides the original data into corresponding branches of data intervals. Subsequently, the method encodes the data according to the interval segments and then converts the number into 0s and 1s. By comparing the relationship between the upper and lower limits of data values and the data coding bits under the use of $MOUE$ and UER probabilities, our data representation method has a lower error expectation or variance than that of the binary representation method, given specific conditions. The details will be discussed in Section IV.

The main contributions of this paper are as follows:

- A novel data coding method is proposed, and it is proven to satisfy LDP.
- The relationship between the error expectation and the variance of weights is analyzed. The proposed segmentation method theoretically and experimentally outperforms the traditional binary method in error analysis.
- Experiments are carried out on two real-world datasets $BANK$ and $MNIST$ with different combinations of weights. The machine learning results proved that the proposed method could better preserve data utility under the same privacy level.

The rest of the paper is structured as follows: Section II introduces relevant work. Section III presents the background and the methods used. Section IV introduces our method's implementation, theoretical analysis and error analysis. Section V is the experimental part. Section VI concludes the paper.

II. Related work

Throughout the evolution of artificial intelligence, the core technology of machine learning has continued to advance, with

This is the author's manuscript of the article published in final edited form as:

Zhang, N., Wang, S., Gao, T., Li, F., Sundar, A. P., & Zou, X. (2023). Local Differential Privacy Preservation via the Novel Encoding Method. 2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS), 478–486. https://doi.org/10.1109/MASS58611.2023.00066

new domains of machine learning being researched, including the dangers of privacy breach posed by the training process of that technology [7]. The emergence of privacy violations and data breaches has prompted extensive study and research into protecting user privacy and data security in machine learning. Differential privacy technology, a privacy protection methodology that is underpinned by mathematical theory, is well-suited to satisfy the demands of personal privacy defence in the modern era of big data.

Differential privacy was introduced by Dwork et al. [8] in 2006 as the most secure disturbance-based privacy protection method. Local differential privacy is achieved through encoding methods and random response. In their research paper [11], Dwork et al. introduced the Direct Encoding (DE) method that perturbs the probability values of the original decimal number to achieve LDP. However, DE has a significant drawback, which is that when the numerical range is extensive, the probability of an input value being transmitted correctly is severely diminished. To compensate for the shortcoming of the DE method, Guo et al. [12] proposed the Unary Encoding method. The unary encoding method encodes the decimal number into a corresponding bit string that has only one '1', and the remaining digits are '0'. This method perturbs the bit string, ensuring the probability of attaining LDP. Furthermore, T. Wang et al. [13] introduced the Pure LDP and Support function concept. Through these two concepts, they analyzed the current Local differential privacy protocols and proposed aggregation methods and estimated the frequency of deviations to identify the appropriate protocol.

Differential privacy technology has gained extensive usage in the fields of machine learning and deep learning. Abadi et al. [14] built a differential privacy random gradient descent algorithm that utilized noise distribution to compute the total privacy cost during network model training. They demonstrated that the appropriate limitation for noise amplitude and size could reduce privacy loss. A generative adversarial network model that is supported by differential privacy was proposed by Beaulieu-Jones [15] et al. to tackle the issue of data leakage in generative adversarial networks. Xie [16] et al. subsequently developed a GAN model based on differential privacy to address the gradient disappearance flaw in Beaulieu-Jones's scheme. Adesuyi et al. [17] proposed a security neural network training method based on differential privacy to deal with the issue of low accuracy resulting from training models on ciphertext. Wang et al. [18] presented a differential privacy algorithm suitable for protecting user privacy in a wide range of machine learning models.

III. PRELIMINARIES

A. Differential Privacy

Differential privacy (DP) [19] is a mechanism designed to address the problem of disclosing private information from databases. In the DP mechanism, a trusted data administrator collects personal user data and injects noise to the data before publishing it. Adding noise to the data ensures that the results of the query request on publicly-visible data will not divulge personal privacy information. The fundamental principle of its implementation involves manipulating the number of records in a statistical data query. The difference between the two data sets used in the query is one record, and the almost similar output obtained from the two sets helps avoid the disclosure of personal privacy data.

Definition 2.1 (ε-Differential privacy) Assuming that D1 and D2 are two adjacent data sets with at most one different datum, we can observe algorithm A outputting result S on D1 and D2. To satisfy the following formula, we need to confirm that the output S of algorithm A on D1 and D2 is as close as possible to each other.

$$Pr[A(D1) \in S] \leq e^{\varepsilon} Pr[A(D2) \in S]. \tag{1}$$

Algorithm A is considered to satisfy ε-Differential Privacy if the output probability distribution of function A on adjacent data sets D1 and D2 remains nearly the same. Here, Pr[.] specifies the probability of event occurrence, and ε is the budget allocated for privacy preservation. A decrease in the value of ε leads to an increase in privacy level, but it also results in a reduction of utility. This is because the more we increase the privacy level, the more we are likely to introduce noise in the dataset, leading to lower accuracy in the resulting output.

B. Local differential privacy

Global Differential Privacy (GDP) requires data providers to place their trust in data curators to safeguard against data leaks, but the task of finding trusted data curators can be challenging. Therefore, Local Differential Privacy (LDP) was proposed as an alternative solution [20]. Unlike GDP, LDP does not rely on trusted data curators. Instead, each data provider adds a layer of noise to their data individually, ensuring differential privacy. By doing so, the data curators cannot deduce the original data, effectively preserving data privacy.

Definition 2.2 (ε-LDP): A randomized algorithm M satisfies ε-Local Differential Privacy (ε-LDP) if and only if, for any two input values t1 and t2 $\in$ D, and for any output z produced by M, the resulting inequality always holds.

$$Pr[M(t1) = z] \leq e^{\varepsilon} Pr[M(t2) = z]. \tag{2}$$

The definitions of Local Differential Privacy (LDP) and Differential Privacy (DP) are similar. The primary distinction is that t in LDP comes from a single user, whereas t in DP comes from all users.

C. Global Sensitivity

Definition 2.3 (Global Sensitivity) Sensitivity represents the amount of noise added to a dataset; it quantifies how much the original data changes as a result of adding the noise. When given adjacent datasets D1 and D2 and a corresponding query function f, sensitivity Δf can be defined by Equation (3) [21].

$$\Delta f = \max_{D1,D2} \parallel f(D1) - f(D2) \parallel_1 \tag{3}$$

D. Improvement process for $MOUE$ and UER

- **Unary Encoding (UE)**

The UE [12] code converts individual data into a d-dimensional vector that consists of a single 1 and all other entries set to 0. The resulting vector is denoted B, with $B[v] = 1$ and $B[i] = 0$ for $i \neq v$. To represent the perturbed vector, we use B'. With a sensitivity of 2 and a privacy budget of ε, the probability of perturbing B to obtain B' can be computed as follows:

$$\Pr\left[B'[i]=1\right] = \begin{cases} p = \frac{e^{\frac{\varepsilon}{2}}}{1+e^{\frac{\varepsilon}{2}}}, & \text{if } B[i]=1 \\ q = \frac{1}{1+e^{\frac{\varepsilon}{2}}}, & \text{if } B[i]=0 \end{cases} \tag{4}$$

- **Optimized Unary Encoding (OUE)**

The UE code generates a vector with many 0s. To assign additional privacy budget to the 0s during the transmission process, a modified UE method was created and named Optimized Universal Encoding OUE [22]. It achieves this by altering the perturbation probability in UE. Specifically, the values of p and q are set as $p = \frac{1}{2}$ and $q = \frac{1}{1+e^{\varepsilon}}$, respectively.

- **Modify Optimized Unary Encoding ($MOUE$)**

$MOUE$ [6] introduces a privacy budget coefficient α to modify OUE. It can increase the probability of transmitting 0 bits and also apply to randomizing strings containing a considerable number of 1s. When the sensitivity is Δf, its randomization probability is:

$$\begin{aligned} p &= \frac{1}{1+\alpha} \\ q &= \frac{1}{1+\alpha e^{\frac{\varepsilon}{\Delta f}}} \end{aligned} \tag{5}$$

- **Utility Enhancing Randomization (UER)**

In $MOUE$, when the α is large, the randomization of 1s is also greatly increased. UER [6] extends the ideas in $MOUE$ to enhance the utility of randomizing binary strings. This is achieved by randomizing half of the bits in the bit string with different probabilities than the other half. As α increases, the UER will increase the probability of keeping 0 in its original state. While for half of the string, the probability of keeping 1s in their original state increases, the corresponding probability decreases for the other half of the string. Thus maintaining the privacy budget parameters unchanged. When the sensitivity is Δf , UER satisfies $\varepsilon -$ LDP in probabilities as follows:

$$p[B[i]v] = \begin{cases} Pr[B[v_1]=1 \mid v_1] = \frac{\alpha}{1+\alpha}, & \text{if } i \in 2N; N \in \mathbb{N} \\ Pr[B[v_2]=0 \mid v_1] = \frac{\alpha e^{\frac{\varepsilon}{\Delta f}}}{1+\alpha e^{\frac{\varepsilon}{\Delta f}}} & \\ Pr[B[v_1]=1 \mid v_1] = \frac{1}{1+\alpha^3}, & \text{if } i \in 2N+1 \\ Pr[B[v_2]=0 \mid v_1] = \frac{\alpha e^{\frac{\varepsilon}{\Delta f}}}{1+\alpha e^{\frac{\varepsilon}{\Delta f}}} & \end{cases} \tag{6}$$

IV. LDPseg

This chapter entails a detailed description of our proposed method in addition to an analysis of the findings. Perturbing the high bit of the binary string usually results in a significant difference between the final and original data. Herein, the impact of each perturbation on the result differs. Excessive error may jeopardize the processing and analysis of the data. We purpose an array of coding methods that aim to represent the data and alleviate the weight of the high position of the coded data. Initially, we studied the basis between 1 and 2 and developed an encoding system to convert decimal numbers into bit strings with different weights. Notably, we found that encoding method with weight 1 performs well in terms of error mathematical expectation or variance during the research process. Consequently, we designed a data representation method, LDPseg, with a weight of 1. The LDPseg method eliminates the high error risk caused by changes in the high-weight bits, with the error caused by perturbations at each position being identical for the final result. This method will not cause a large difference between the disturbed results and the original data and reduce the error. In this chapter, we describes the process of converting data using other coding methods, LDPseg method, differential privacy analysis, and error analysis.

A. The process of other coding methods

(1) Set the length of the encoded data as $length$ and the weight of the converted data as $weight$. Calculate the maximum value $length\_represent$ when the weight of $length$ bit is $weight$. Next, establish the upper bound (UB) and the lower bound (LB) of the input data. Finally, calculate the size of a piece of data in the data range $d = (UB - LB)/length\_represent$;

(2) Compute the decimal value that each bit string data represents under the given weight. Then, multiply the resulting value by the size of a piece of data d to ensure that the represented data range falls within [LB,UB].

(3) The decimal value closest to each original data in (2) should be determined, and its corresponding bit string is the resultant output after data conversion. This process ensures accurate and reliable results while converting data from one format to another.

B. The steps of LDPseg

(1) Data conversion. Convert the decimal system to a string containing only 0s and 1s, but not binary. The conversion process involves specifying the number of bits (n) of the resultant data and computing the upper (UB) and lower (LB) limits for all source data. Subsequently, each original decimal value (OD) can be converted in turn. Formula 7 can be employed to determine the data segment interval (i, j) that corresponds to each OD.

$$((UB - LB)/n) * i \leq OD - LB \leq ((UB - LB)/n) * j \tag{7}$$

Algorithm 1 The process of LDPseg generating differential private output.

Require:
The original decimal number data set $OD = \{OD_1, OD_2, \dots OD_r\}$;
The estimated upper bound (UB) ;
The estimated lower bound (LB) ;
The number of bits for the data string n ;
The privacy budget ε ;

Ensure:
The perturbed new decimal number data set $ND = \{ND_1, ND_2, \dots ND_r\}$;

1: Calculate the $\{i_1, i_2, \dots i_r\}$ corresponding to OD through the formula $i_j = \text{int}\left(\frac{(OD_j - LB) \times n}{UB - LB}\right)$, where $1 \leqslant j \leqslant r$;
2: Convert OD to the data string set $ST = \{ST_1, ST_2, \dots ST_r\}$ according to set the first i_j position to 1 and the last $n - i_j$ to 0;
3: Merge all data strings in ST into a data string LST of length rn in order ;
4: Randomize each element of LST according to the selected probability p to form a new data string $PLST$;
5: Re-split $PLST$ into r data strings of length n, which is $PST = \{PST_1, PST_2, \dots PST_r\}$;
6: Count the number c_{1j} of 1s in each data string in PST to form $ND = \{ND_1, ND_2, \dots ND_r\}$ according to the formula $ND_j = c_{1j} \times \frac{UB - LB}{n}$, where $1 \leqslant j \leqslant r$;
7: **return** ND;

After i and j are calculated, the data after OD conversion is that the first part contains i 1s and the last half contains $(n-i)$ 0s. For instance, if the upper bound (UB) in a dataset happens to be 100 and the lower bound (LB) is 0, and a 10-bit conversion is required, then the interval for 62 can be expressed as (6,7). Such an interval value for 62 would be converted into 1111110000, comprising of six 1s and four 0s.

(2) Data aggregation. Given a privacy budget ε, when there exist r data in step (1), if each string is perturbed individually without merging the data, the final privacy budget will become superimposed, causing privacy budget to become $r * \varepsilon$. This violates the given privacy budget. However, such issue can be resolved by merging all data into one data and then perturbing it. To achieve this, we propose a specific method. We can combine the converted data of r data to form a new string that contains only 0s and 1s; its length will be rn.

(3) Data randomization perturbation. In step (2), the length of the merged string is $r * n$. As each bit can differ by a maximum of $r * n$ bits due to the combined string perturbation, the sensitivity is $r * n$ bits. Based on the different probability methods presented in section III-D, we can obtain the necessary probability required by the perturbation in our approach by merely replacing the sensitivity value. Moreover, the specific probability expressions are presented in sections IV-B and IV-C.

(4) Data segmentation. Divide the perturbed data in step (3) into r data of length n according to the length n of each data in step (2).

(5) Data recovery. Count the number of 1s in each disturbing data point in (4), denoted by c, and restore it to the corresponding new decimal value (ND). The calculation formula of ND is $ND = c * (UB - LB)/n$. The calculated ND value will be a privacy-enhanced data that can be safely used.

Algorithm 1 presents the detailed procedure for generating differential private output using LDPseg.

C. Differential privacy analysis

The $MOUE$ and UER mentioned in III-D convert the data into binary and then perturb according to their probabilities. It has been shown that the resulting perturbed data satisfies ε-LDP [6]. The LDPseg converted data string only contains 1s and 0s and perturbs according to the corresponding probabilities, which also satisfies the LDP, as described below.

(1) Using the probabilities in $MOUE$. The probabilities p and q used in LDPseg are determined based on the sensitivity of the data, which is $r * n$. Specifically, p and q should be set to $p = \frac{\alpha}{1+\alpha}$ and $q = \frac{1}{1+\alpha e^{\frac{\varepsilon}{rn}}}$, respectively. With these probabilities, the LDPseg mechanism satisfies ε - LDP.

$$\begin{aligned}\frac{\Pr[\boldsymbol{B} \mid v_1]}{\Pr[\boldsymbol{B} \mid v_2]} &= \frac{\prod_{i\in[d]} \Pr[\boldsymbol{B}[i] \mid v_1]}{\prod_{i\in[d]} \Pr[\boldsymbol{B}[i] \mid v_2]} \\ &\leq \left(\frac{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_1] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_1]}{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_2] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_2]}\right)^{rn} \\ &= \left(\frac{\left(\frac{1}{1+\alpha}\right)}{\left(\frac{\alpha}{1+\alpha}\right)} \cdot \frac{\left(\frac{\alpha e^{\frac{\varepsilon}{rn}}}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}{\left(\frac{1}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}\right)^{rn} = e^{\varepsilon}.\end{aligned} \tag{8}$$

(2) Using the probabilities in UER. Because the sensitivity is $r*n$. We can set p and q as shown in Equation 6 by replacing Δf in the original formula with rn. With these probabilities, the LDPseg mechanism satisfies ε - LDP.

$$\begin{aligned}&\frac{\Pr[\boldsymbol{B} \mid v_1]}{\Pr[\boldsymbol{B} \mid v_2]} = \frac{\prod_{i\in[d]} \Pr[\boldsymbol{B}[i] \mid v_1]}{\prod_{i\in[d]} \Pr[\boldsymbol{B}[i] \mid v_2]} \\ &= \frac{\prod_{i\in 2N} \Pr[\boldsymbol{B}[i] \mid v_1]}{\prod_{i\in 2N} \Pr[\boldsymbol{B}[i] \mid v_2]} \times \frac{\prod_{i\in 2N+1} \Pr[\boldsymbol{B}[i] \mid v_1]}{\prod_{i\in 2N+1} \Pr[\boldsymbol{B}[i] \mid v_2]} \\ &\leq \left(\frac{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_1] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_1]}{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_2] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_2]}\right)^{\frac{rn}{2}}_{i\in 2N} \\ &\times \left(\frac{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_1] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_1]}{\Pr[\boldsymbol{B}[v_1] = 1 \mid v_2] \Pr[\boldsymbol{B}[v_2] = 0 \mid v_2]}\right)^{\frac{rn}{2}}_{i\in 2N+1} \\ &= \left(\frac{\left(\frac{\alpha}{1+\alpha}\right)}{\left(\frac{1}{1+\alpha}\right)} \cdot \frac{\left(\frac{\alpha e^{\frac{\varepsilon}{rn}}}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}{\left(\frac{1}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}\right)^{\frac{rn}{2}} \left(\frac{\left(\frac{1}{1+\alpha^3}\right)}{\left(\frac{\alpha^3}{1+\alpha^3}\right)} \cdot \frac{\left(\frac{\alpha e^{\frac{\varepsilon}{rn}}}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}{\left(\frac{1}{1+\alpha e^{\frac{\varepsilon}{rn}}}\right)}\right)^{\frac{rn}{2}} \\ &= e^{\varepsilon}.\end{aligned} \tag{9}$$

D. Error analysis

The previous binary representation of data has been limited to an integer and decimal division. However, the weight-based method with a range between 1.1 and 2, as well as the LDPseg algorithm we have studied, do not represent data with a decimal part. To address this, we divide the n-bit data into 2 parts; an a-bit integer section and a b-bit decimal section of binary. For comparison purposes, we have also divided the n-bit data using the weight 1.1 to 2 method and LDPseg, as depicted in Figure 1. Negative numbers are not considered (when $LB \geqslant 0$). In each perturbation probability method, we first analyze the weight-based methods between 1.1 to 2, discussed in our paper. Next, we will introduce a comparison of the weight-one method with the previous binary approach. The proposed binary method is equivalent to scaling data within a representation range. Though, we should note that the traditional binary representation data format has changed, the underlying principle remains the same.

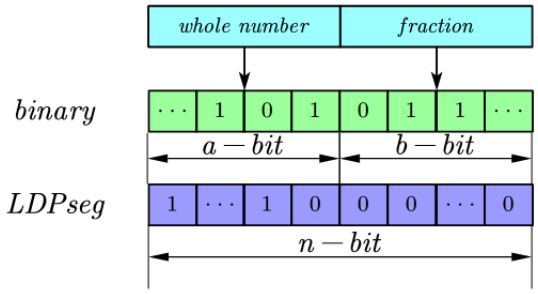

Fig. 1. The example of dividing n-bit data into a-bit and b-bit.

(1) Compared with $MOUE$. In $MOUE$, the perturbation probabilities between 0 to 1 and 1 to 0 differ, where in q_1 and q_2 are set. $e(i)_1^0$ and $e(i)_0^1$ represent the error caused by the i_{th} 1 converting to 0, and the i_{th} 0 turning to 1, respectively, where $e(i)_1^0$ and $e(i)_0^1$ are opposite numbers. We use X to denote the error, and thus, the mathematical expectation is expressed as follows.

$$\begin{aligned} EX &= \sum_{i=1}^{n}\left[\frac{1}{2}\times q_1 \times e(i)_1^0 + \frac{1}{2}\times q_2 \times e(i)_0^1\right] \\ &= \left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)\sum_{i=1}^{n} e(i)_0^1 \end{aligned} \tag{10}$$

When the weight range of this paper is between 1.1 to 2, we denote the selected weight as x. For data consisting of 0s and 1s with a length of n, the values data represent with 1 from the high position to the low position are as follows: $x^{n-1}*d, x^{n-2}*d, x^{n-3}*d, ...x^1*d, x^0*d$. Here, d represents the measurement of data represented by n-bit data with a weight of x, given the original range of data to be converted is $[LB, UB]$. We provide the calculation formula below.

$$d = \frac{UB-LB}{\frac{1-x^n}{1-x}} = \frac{x-1}{x^n-1}(UB-LB) \tag{11}$$

While

$$\begin{aligned} \sum_{i=1}^{n} e(i)_0^1 &= d(x^{n-1}+x^{n-2}+x^{n-3}+...+x^1+x^0) \\ &= \frac{(x-1)(UB-LB)}{x^n-1} * \frac{1-x^n}{1-x} \\ &= UB-LB \end{aligned} \tag{12}$$

As a result, the expected error when a weight of x is selected is calculated as follows

$$EX_1 = \left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)(UB-LB) \tag{13}$$

It is evident that the error expectation for the data set represented by this method is identical. As a result, the variance is calculated as follows:

$$\begin{aligned} DX_1 &= EX_1^2 - (EX_1)^2 \\ &= \frac{1}{2}(q_2-q_1)(UB-LB)\sum_{i=1}^{n}(e(i)_0^1)^2 - (EX_1)^2 \end{aligned} \tag{14}$$

Where

$$\begin{aligned} \sum_{i=1}^{n}(e(i)_0^1)^2 &= d^2(x^{2(n-1)}+x^{2(n-2)}+x^{2(n-3)}+...+x^{2*1}+x^{2*0}) \\ &= \frac{(1-x)(UB-LB)^2}{1-x^n} * \frac{1+x^n}{1+x} \end{aligned} \tag{15}$$

From this, we can conclude that when n is constant, a smaller x leads to a smaller variance, resulting in more stable errors for each number. As a result, we arrive at our next limit scenario, where x equals 1, representing the LDPseg and the previous binary discussion.

For binary, $e(i)_0^1$ is equivalent to the weight of the i_{th} bit, which we define as 2^{w_i}. Thus, the expected error is mathematically calculated as EX_2= $\left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)\sum_{i=1}^{n} 2^{w_i}$. In contrast, for LDPseg, we distribute the value of $UB-LB$ equally into n bits, making the weight of every bit the same. Therefore, $e(i)_0^1$ is equal to $d = \frac{(UB-LB)}{a+b}$, wherein the expected error is EX_3= $\left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)\sum_{i=1}^{n} d$. Since $\left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)$ is a constant term, only $\sum_{i=1}^{n} 2^{w_i}$ and $\sum_{i=1}^{n} d$ require comparison. The results for $\sum_{i=1}^{n} 2^{w_i}$ and $\sum_{i=1}^{n} d$ are as follows.

$$\begin{aligned} &\sum_{i=1}^{n} d = nd = UB-LB \\ &\sum_{i=1}^{n} 2^{w_i} = \frac{2^{a-1}\left[1-\left(\frac{1}{2}\right)^{a+b}\right]}{1-\frac{1}{2}} = 2^a - 2^{-b} \end{aligned} \tag{16}$$

Corollary 1. When the integer part requires a bits, $UB - LB < 2^a$.

Proof: In order to ensure that all numbers can be represented in binary form, the length of the integer part in the binary string should be determined by UB. When the integer part requires a bits, UB must fall within the following range: $2^{a-1} \leqslant UB < 2^a$. Therefore, $UB - LB < 2^a$, completing the proof.

Corollary 2. When the integer part and the fraction part require a bits and b bits, and $UB \leqslant 2^a - 1$ or $LB \geqslant 2^{-b}$, $UB - LB < 2^a - 2^{-b}$.

Proof: From Corollary 1 we can get: when $UB \leqslant 2^a - 1$,$UB - LB \leqslant 2^a - 1 - LB < 2^a - 2^{-b}$; or when $LB \geqslant 2^{-b}$, $UB - LB < 2^a - LB < 2^a - 2^{-b}$. The proof is over.

According to Corollary 2, it can be proven that $\sum_{i=1}^{n} d \leqslant \sum_{i=1}^{n} 2^{w_i}$ when $UB \leqslant 2^a - 1$ or $LB \geqslant 2^{-b}$. It can also be demonstrated that, regardless of the length of the data string, and subject to the aforementioned conditions, the mathematical expectation of error in our method approaches zero whether $\left(\frac{1}{2}q_2 - \frac{1}{2}q_1\right)$ is positive or negative.

(2) Compared with UER**.** As previously introduced in the background section, UER is an improved approach to $MOUE$. Unlike $MOUE$, it makes the number change with three probabilities, denoted as q_1, q_2, q_3, according to the following definitions.

$$\begin{cases} q_1 = \frac{1}{1+\alpha}; & 1 \to 0 \quad \text{(even digits)} \\ q_2 = \frac{\alpha^3}{1+\alpha^3}; & 1 \to 0 \quad \text{(odd digits)} \\ q_3 = \frac{1}{1+\alpha e^{\frac{\varepsilon}{rn}}}; & 0 \to 1 \quad \text{(all digits)} \end{cases} \tag{17}$$

And the mathematical expectation of X can be expressed as:

$$\begin{aligned} EX = & \sum_{i \in 2I}^{n} \left[\frac{1}{2} \times q_1 \times e(i)_1^0 + \frac{1}{2} \times q_3 \times e(i)_0^1 \right] \\ & + \sum_{i \in 2I+1}^{n} \left[\frac{1}{2} \times q_2 \times e(i)_1^0 + \frac{1}{2} \times q_3 \times e(i)_0^1 \right] \\ & = -\frac{1}{2}q_1 \sum_{i \in 2I}^{n} e(i)_0^1 - \frac{1}{2}q_2 \sum_{i \in 2I+1}^{n} e(i)_0^1 + \frac{1}{2}q_3 \sum_{i=1}^{n} e(i)_0^1 \end{aligned} \tag{18}$$

Therefore, the error expectation when selecting a weight of x is

$$\begin{aligned} EX_1 = & \frac{(x-1)(UB-LB)}{x^n - 1}[-\frac{1}{2}q_1 * \frac{x(1-x^n}{1-x^2} \\ & - \frac{1}{2}q_2 * \frac{1-x^n}{1-x^2}] + \frac{1}{2}q_3(UB-LB) \\ & = \frac{(x-1)(UB-LB)}{x^n - 1} * \frac{1-x^n}{1-x^2}(-\frac{1}{2}q_1 x - -\frac{1}{2}q_2) \\ & + \frac{1}{2}q_3(UB-LB) \\ & = (UB-LB)\frac{1}{1+x}(-\frac{1}{2}q_1 x - \frac{1}{2}q_2) + \frac{1}{2}q_3(UB-LB) \end{aligned} \tag{19}$$

In this study, we consider the case where n is an even number. If n is odd, there is only one missing number, and the analysis is similar to the even case; hence, we will not discuss the odd number situation anymore. By taking the derivative of the previous results, it can be concluded that the derivative function is always greater than 0, indicating that EX_1 is monotonically increasing. Consequently, smaller values of x lead to smaller errors. Next, the error discussion between $x = 1$, i.e. LDPseg, and previous binary method is drawn again. That is, the error expectation of traditional binary is

$$\begin{aligned} EX_2 = & \frac{1}{2}q_1 \sum_{i \in 2I}^{n} 2^{w_i} - \frac{1}{2}q_2 \sum_{i \in 2I+1}^{n} 2^{w_i} + \frac{1}{2}q_3 \sum_{i=1}^{n} 2^{w_i} \\ & = (-\frac{1}{3}q_1 - \frac{1}{6}q_2 + \frac{1}{2}q_3)(2^a - 2^{-b}) \end{aligned} \tag{20}$$

Similarly, the mathematical expectation EX_3 of LDPseg is:

$$\begin{aligned} EX_3 = & -\frac{1}{2}q_1 \sum_{i \in 2I}^{n} d - \frac{1}{2}q_2 \sum_{i \in 2I+1}^{n} d + \frac{1}{2}q_3 \sum_{i=1}^{n} d \\ & = (-\frac{1}{4}q_1 - \frac{1}{4}q_2 + \frac{1}{2}q_3)(UB - LB) \end{aligned} \tag{21}$$

Corollary 3. When the sensitivity is high and $(UB-LB) < \frac{2}{3}(2^a - 2^{-b})$, EX_3 is closer to 0 than EX_2 .

Proof: It is worth noting that in practice, since rn is too large, $\frac{\varepsilon}{rn}$ tends to 0, and q_3 tends to q_1. And EX_2 and EX_3 are approximately equal to the following

$$\begin{aligned} EX_2 &= \frac{1}{6}(q_1 - q_2)\left(2^a - 2^{-b}\right) \\ EX_3 &= \frac{1}{4}(q_1 - q_2)(UB - LB) \end{aligned} \tag{22}$$

Therefore, when $(UB-LB) < \frac{2}{3}(2^a - 2^{-b})$, EX_3 is closer to 0 than EX_2. The proof is over.

Therefore, from Corollary 3 we have: under the same probability conditions as UER, our method performs better on the error mathematical expectation when the sensitivity is high and $(UB - LB) < \frac{2}{3}(2^a - 2^{-b})$

V. EXPERIMENTAL RESULTS AND DISCUSSION

This chapter presents a comparative analysis of our proposed method and a previous binary method in the context of deep learning and machine learning experiments. We employed two datasets: $MINST$ [14] and $BANK\ MARKETING$ [23]. The experiments were conducted using a computer with the Windows 10 operating system, and implemented using the Python programming language.

A. Experimental Setup

The $MINST$ dataset comprises grayscale images of handwritten digits organized by the National Institute of Standards and Technology (NIST) in the United States. Among the 70,000 images, 250 individuals contributed pictures, with 50% high school students and 50% Census Bureau staff. Each digital image size was 28x28 pixels, including hand-drawn numerals. These data were divided into 60,000 training set and 10,000 test set, which allowed for proper analysis of the dataset's effectiveness in image classification.

The $BANK$ dataset originated from the UCI website, and was used to track the direct marketing activities carried out by a Portuguese banking institution. The primary objective of this dataset was to develop a classification model capable of predicting whether or not a customer would subscribe

to a term deposit. The dataset comprises a total of 45,211 observations, each of which contains 16 input variables and one outcome variable. No missing values were recorded in the dataset. The $BANK$ dataset contains text information with fixed terms, and different numbers can be used to represent different attributes in each column, thus converting the dataset original file into a numeric csv file.

We decided to employ the Convolutional Neural Network model for the $MINST$ dataset, and the k-Nearest Neighbors model for the $BANK$ dataset. The addition of differential privacy to the two models was implemented in two suitable locations. As discussed in Chapter III, we perturbed the $BANK$ dataset directly, followed by machine learning. Meanwhile, for the $MNIST$ dataset, we performed perturbation during the model's tensor conversion. Given the high sensitivity, the privacy budget's effect on the perturbation probability is negligible, except when set to relatively high values; Hence, we performed the experiment with only a privacy budget value of 1. Our objective is to compare coding methods' influence on accuracy, and therefore, we set hyperparameter α to 5, and there is not much to explore the influence of hyperparameter α. At the same time, we did not vary any additional deep learning or machine learning parameters. Finally, we evaluated the accuracy of differential private methods using distinct encodings on the $BANK$ and $MNIST$ datasets, respectively.

B. Experimental details

a)$MNIST$

The convolutional neural network takes the 28×28 image dataset as input, which is passed to the convolutional layer. This layer employs a convolution kernel for feature extraction and mapping. As convolution is a linear operation, non-linear mapping needs to be added, and the activation layer is utilized for non-linear mapping. Common activation functions include sigmoid, tanh, Relu, Maxout, etc. In our experiment, we select ReLU as the activation function. Following the activation layer, the pooling layer performs downsampling and sparse processing to minimize data computation. Finally, the fully connected layer is added at the end of the CNN to mitigate information loss of features.

We transform the data into a one-dimensional vector between the pooled layer and the fully connected layer. Differential privacy is added to the one-dimensional vector through perturbation. The perturbed data is then transferred to the fully connected layer to preserve the privacy of the original data. We adopt diverse encoding techniques for implementing the differential private data conversion process, while keeping other parameters constant. Specifically, when converting data using the previous binary method, we use 2 bits to represent integers and 8 bits to represent decimals. In the encoding method proposed in this paper, only 10 bits are needed to represent data, and there is no need to divide integer and decimal parts. In our experiment, the privacy budget is set to 1, and the data perturbation involves 320 pieces of data, leading to a total sensitivity of 320*10=3200. The learning rate is 0.01, the optimization algorithm used is SGD, and the number of training epochs is 100, which ensures a stable accuracy rate.

b)$BANK$

For numerical csv file for the $BANK$ data set, each column of the input variable can be perturbed through differential privacy, and then machine learning can be performed. In our experiment, we first count the number of attributes, then start numbering from 0 and clean the original data to convert all words into numbers. We normalize each column of numbers to facilitate perturbation. After the normalized csv data is obtained, the data are converted by taking different approaches to the encoding respectively, which are perturbed to produce new csv files. In the perturbation process, we turn each original number into a bit string of one integer and 9 decimals, convert the 45211 numbers in each column, splicing and perturbation, so the sensitivity is 452110. And our privacy budget ε is set to 1.

The k-nearest neighbor (k-NN) algorithm [24], a popular classification and regression method, was used in this study as the supervised learning approach. The KNeighborsClassifier function in the sklearn package was called to implement the KNN algorithm with the parameter n_neighbors set to 5, representing the number of neighbors to consider for classification. Specifically, when the KNN algorithm classifies a given training dataset, it assigns a new instance to the class most frequently occurring among its five nearest neighbors. The parameter p was set to 2, indicating that the Euclidean distance metric was used. To evaluate the performance of the trained model, the dataset was split into a training set (75% of the data) and a test set (25% of the data).

C. Experimental results

a)$MNIST$

The experimental results obtained using the MOUE probability perturbation approach are presented in Figures 2 and 3. In the figure, the preceding numbers in the usage method represent the weights used for encoding. Figure 2 illustrates the performance of the training and testing sets with different weight values assigned to each encoding. As indicated in the figure, the accuracy improves as the weight value approaches 1. Therefore, we compared the LDPseg method with a weight of 1 to previous binary methods, and the corresponding accuracy values are displayed in Figure 3. The results demonstrate that the LDPseg method significantly outperforms the binary method, achieving a training set accuracy of 92.3% and a testing set accuracy of 90.45%, while the binary method only achieves 84.91% and 82.65% on the training and test set, respectively. Lastly, we compared the performance of the LDPseg method to that of the differential privacy approach that uses Gaussian noise. The results presented in Figure 3 show that the LDPseg method outperforms the Gaussian noise-based method.

The experimental results using UER probability perturbation are presented in Figures 4 and 5. Because UER is obtained by optimizing MOUE, the results obtained with these methods are similar to each other, but the performance achieved with

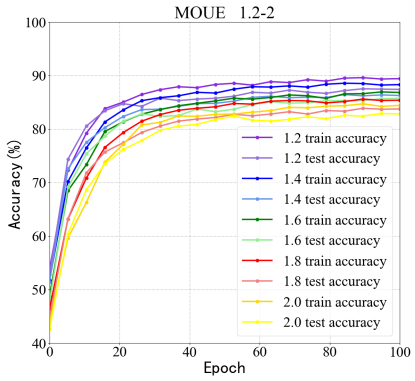

Fig. 2. When the weight is 1.2 to 2, the accuracy result of using MOUE probability perturbation

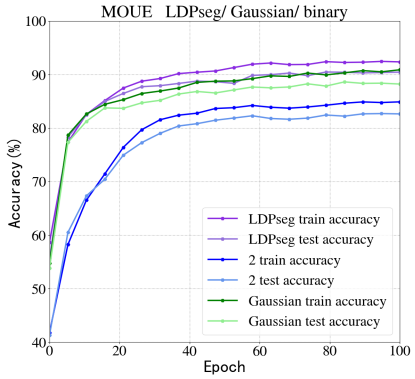

Fig. 3. When the method used is LDPseg, Gaussian noise and binary, the accuracy result of using MOUE probability perturbation

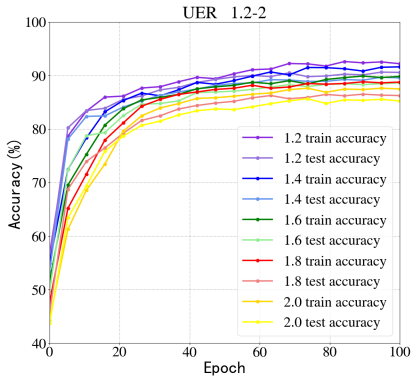

Fig. 4. When the weight is 1.2 to 2, the accuracy result of using UER probability perturbation

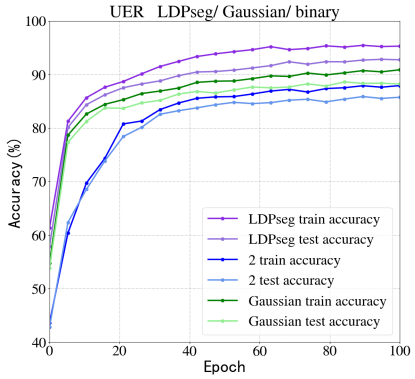

Fig. 5. When the method used is LDPseg, Gaussian noise and binary, the accuracy result of using UER probability perturbation

UER is generally superior to MOUE. The results displayed in Figure 4 depict that the accuracy of both the training and test set increases as the weight of each encoding utilized approaches 1. Figure 5 presents a comparison between the LDPseg and binary methods separately. Notably, the experimental results indicate superior performance of the LDPseg method in comparison to binary methods. The LDPseg method has an accuracy of 95.41% for the training set and an accuracy of 92.76% for the testing set, whereas the binary method has accuracy of only 87.84% for the training set and an accuracy of 85.76% for the testing set. Additionally, we compared the LDPseg method with differential privacy method with Gaussian noise, and the results indicate that LDPseg method outperforms Gaussian significantly.

b)$BANK$

We conducted experiments only using UER perturbation probabilities on the BANK dataset, since UER is an optimization of MOUE, and we had already experimented on the MNIST dataset. The encoding methods with weights ranging from 1.2 to 2 were utilized for the experiment. As illustrated in Figure 6, weights closer to 1 yield better results. Consequently, we introduced the LDPseg method with a weight of 1 and performed comparative experiments with previous binary methods, as shown in Figure 7. The results demonstrate a significant improvement in the LDPseg method compared to binary methods, with a training set accuracy of 91.49% and a test set accuracy of 90.23%, in contrast to a binary method training set accuracy of only 88.37% and a test set accuracy of only 87.42%. We also compared the LDPseg method with the differential privacy method utilizing Gaussian noise, and the results in Figure 7 reveal that even on the BANK dataset, LDPseg method outperforms Gaussian.

VI. CONCLUSION

We explored other data coding methods applicable to LDP, which only contain 0s and 1s, distinct from binary methods used previously. According to theoretical analysis and experiments, the utility of local differential privacy (LDP) increases as each weight occupies a smaller portion. In other words, the closer the weight is to 1, the higher the utility of LDP. Therefore, we introduced the LDPseg approach with a weight of 1. It eliminates the influence of weights assigned to different numbers, this approach is applied to the LDP mechanism where the perturbation probability of each number

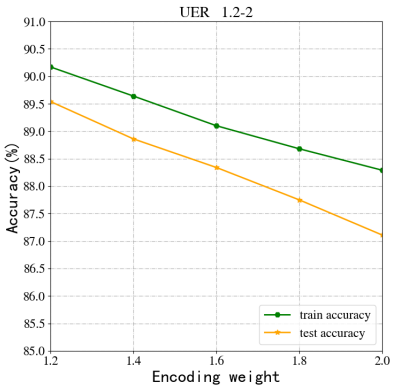

Fig. 6. When the method used is LDPseg, Gaussian noise and binary, the accuracy result of using UER probability perturbation

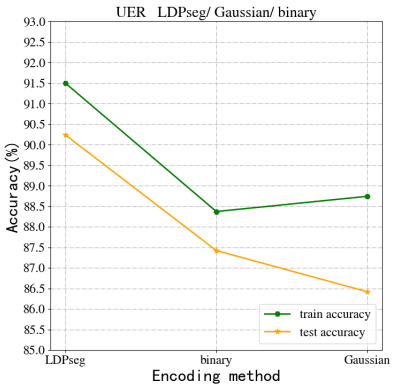

Fig. 7. When the method used is LDPseg, Gaussian noise and binary, the accuracy result of using UER probability perturbation

is the same, and the result of each perturbation has the same influence on the final outcome. Avoiding the situation where the binary perturbation causes a significant difference between the data before and after perturbation if the high level of the perturbation changes, which can impact the use of the data, LDPseg employs probabilistic perturbations in $MOUE$ and UER. It has been demonstrated that LDPseg satisfies ε-LDP and results in lower error expectation than the binary representation method. We used the $MNIST$ and $Bank$ dataset to experiment with this method. The results show that by using LDPseg to preserve data privacy while training the datasets using both deep learning and machine learning techniques, we can represent the data with higher accuracy and less error compared to binary. Our method offers a new data privacy protection approach with high accuracy and minimal error.

‘

REFERENCES

[1] Zheng, H. , H. Hu , and Z. Han . ”Preserving User Privacy for Machine Learning: Local Differential Privacy or Federated Machine Learning?.” Intelligent Systems, IEEE PP.99(2020):1-1.

[2] Begum, S. H. , and F. Nausheen . ”A comparative analysis of differential privacy vs other privacy mechanisms for Big Data.” 2018:512-516.

[3] Nguyên, Thng T, et al. ”Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy.” (2016).

[4] Dwork, C. , and A. Roth . ”The Algorithmic Foundations of Differential Privacy.” Foundations and trends in theoretical computer science (2013).

[5] [1] Bao, T. , et al. ”Privacy-Preserving Collaborative Filtering Algorithm Based on Local Differential Privacy.” China communications 18.11(2021):42-60.

[6] Arachchige, Pcm , et al. ”Local Differential Privacy for Deep Learning.” IEEE Internet of Things Journal 7.7(2020):5827-5842.

[7] Papernot, N. , et al. ”Towards the Science of Security and Privacy in Machine Learning.” arXiv (2016).

[8] Dwork, C. . ”Differential Privacy.” lecture notes in computer science (2006).

[9] Dwork, C. , F. Mcsherry, and K. Nissim. ”Calibrating Noise to Sensitivity in Private Data Analysis.” Proceedings of the Vldb Endowment 7.8(2006):637-648.

[10] Mcsherry, Frank, and K. Talwar. ”Mechanism Design via Differential Privacy.” Foundations of Computer Science, 2007. FOCS ’07. IEEE Symposium on IEEE, 2007:94-103.

[11] Wang, Shaowei , et al. ”Private Weighted Histogram Aggregation in Crowdsourcing.” International Conference on Wireless Algorithms, Systems, and Applications, 2016: 250-261.

[12] Erlingsson, Lfar , V. Pihur , and A. Korolova . ”RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response.” Proceedings of the 2014 ACM SIGSAC conference on computer and communications security, 2014: 1054–1067.

[13] T. Wang, J. Blocki, N. Li, and S. Jha, “Locally differentially private protocols for frequency estimation,” in Proc. 26th USENIX Security Symp.(USENIX Security), 2017: 729–745.

[14] Martí, et al. ”Deep Learning with Differential Privacy.” ACM (2016).

[15] Beaulieu-Jones, Brett K. , et al. ”Privacy-preserving generative deep neural networks support clinical data sharing.” Cold Spring Harbor Laboratory 7(2017).

[16] Xie, L. , et al. ”Differentially Private Generative Adversarial Network.”, 10.48550/arXiv.1802.06739. 2018.

[17] Adesuyi, T. A. , and B. M. Kim . ”A layer-wise Perturbation based Privacy Preserving Deep Neural Networks.” International Conference onArtificial Intelligence in Information and Communication Department of Software Engineering, Kumoh National Institute of Technology, Gumi, South Korea;Department of Software Engineering, Kumoh National Institute of Technology, Gumi, South Korea;, 2019.

[18] Wang, N. , et al. ”Collecting and Analyzing Multidimensional Data with Local Differential Privacy.” 2019.

[19] Chamikara, Map , et al. ”An Efficient and Scalable Privacy Preserving Algorithm for Big Data and Data Streams.”, 10.1016/j.cose.2019.101570.

[20] Joseph, M. , et al. ”Local Differential Privacy for Evolving Data.” Journal of Privacy and Confidentiality (2018).

[21] Yue, W. , X. Wu , and D. Hu . ”Using Randomized Response for Differential Privacy Preserving Data Collection.” EDBT/ICDT Workshops, 2016: 1–8.

[22] Wang, Tianhao , J. Blocki , and S. K. Jha . ”Locally differentially private protocols for frequency estimation.” USENIX Security Symposium, 2017: 729–745.

[23] Shashidhara, B. M. , et al. ”Evaluation of Machine Learning Frameworks on Bank Marketing and Higgs Datasets.” 2015 Second International Conference on Advances in Computing and Communication Engineering (ICACCE) IEEE, 2015.

[24] Okfalisa, et al. ”Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification.” 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), 2017: 294-298.