Homomorphic Model Selection for Data Analysis in an Encrypted Domain †

Mi Yeon Hong 1,2, Joon Soo Yoo 1,2 and Ji Won Yoon 1,2,*

1 School of Cyber Security, Korea University, Seoul 02841, Korea; hachikohmy@korea.ac.kr (M.Y.H.); sandiegojs@gmail.com (J.S.Y.)
2 Institute of Cyber Security and Privacy (ICSP), Korea University, Seoul 02841, Korea
* Correspondence: jiwon_yoon@korea.ac.kr
† This paper is an extended version of our paper published in Hong, M.Y.; Yoon, J.W. Model Selection for Data Analysis in Encrypted Domain: Application to Simple Linear Regression. In Proceedings of the 20th International Conference, WISA 2019, Jeju Island, Korea, 21–24 August 2019.

Received: 20 July 2020; Accepted: 28 August 2020; Published: 4 September 2020

Abstract: Secure computation, a methodology of computing on encrypted data, has become a key factor in machine learning. Homomorphic encryption (HE) enables computation on encrypted data without leaking any information to untrusted servers. In machine learning, the model selection method is a crucial algorithm that determines the performance and reduces the fitting problem. Despite the importance of finding the optimal model, none of the previous studies have considered model selection when performing data analysis through the HE scheme. The HE-based model selection we proposed finds the optimal complexity that best describes given data that is encrypted and whose distribution is unknown. Since this process requires a matrix calculation, we constructed the matrix multiplication and inverse of the matrix based on the bitwise operation. Based on these, we designed the model selection of the HE cross-validation approach and the HE Bayesian approach for homomorphic machine learning. Our focus was on evidence approximation for linear models to find goodness-of-fit that maximizes the evidence. We conducted an experiment on a dataset of age and Body Mass Index (BMI) from Kaggle to compare the capabilities and our model showed that encrypted data can regress homomorphically without decrypting it.

Keywords: fully homomorphic encryption; bitwise operation; model selection; cross validation; evidence approximation; gauss-jordan elimination;

1. Introduction

Due to the continuous increase in computational power and the rapid development of processing and storage technologies, people use cloud computing to manage and analyze massive amounts of biomedical data generated from many sensors. Using cloud computing requires one to trust and share their data with the cloud server. Since all the data resides with the cloud server, a critical data issue arises if the cloud provider misuses the information. One of the best ways to guarantee data privacy is to encrypt data before sending it to the cloud server. However, there is still the problem of the cloud servers calculating the encrypted data to respond to the request from the client. The client needs to offer the secret key to the server to decrypt the data before executing the calculations, which could lead to breach of confidentiality or invasion of privacy. Homomorphic encryption (HE) is the appropriate solution for these issues as it allows the implementation of operations on encrypted data without decrypting it.

Machine learning provides systems with the ability to learn and improve automatically from experience without being programmed to do so. HE can be used to preserve privacy and can also be
applied to machine learning to find useful information from big data. For example, HE based machine learning prevents leakage of personal information, while predicting a patient’s disease in the healthcare industry or creating a model for assessing an individual’s creditworthiness in the banking industry. Various machine learning methods have been studied using various HE libraries [1–3]. Several studies have been conducted about linear regression, logistic regression, K-Nearest Neighbor, and K-means in References [4,5] based on the proposed HE scheme.

In our previous work, we treated HE based model selection from a validation technique perspective. In this paper, we extend the prior work by adding HE based evidence approach in a Bayesian framework by proposing various ways to construct the final algorithm, including the matrix inverse approach in different ways. To determine the best model for the HE, we conducted experiments with the proposed model using the same dataset and compared the results with the results from previously proposed model. The key contributions of this study were as follows:

- The proposed model implements basic dynamic operations with bit-by-bit operations that take accuracy and speed into account. We introduced isomorphic logarithms and HE logarithms used for Bayesian model selection.
- We also employed a matrix multiplication method based on our operations and expanded our computation to form matrix inverse operation using the Gauss-Jordan elimination method.
- Finally, we proposed two ways of the first HE based model selection: Homomorphic cross-validation (HCV) and Homomorphic Bayesian model selection (HBMS), to determine the complexity of models that can explain the data well when given the encrypted data.

2. Background

2.1. Homomorphic Encryption

Homomorphic encryption (HE) is a scheme that allows arbitrary computations on encrypted data. It is based on the property that the result of operations between ciphertext is equal to the result of operations between plaintexts, hence the server can operate on the client’s data without decrypting it. If the following equation holds, then the encryption scheme is called homomorphic over the operation $\ast$.

$$\text{Enc}(m_1 \circ m_2) = \text{Enc}(m_1 \ast m_2), \quad \forall m_1, m_2 \in M,$$

where Enc is an encryption algorithm, and $M$ is a set of plaintexts.

In 1978, the concept of privacy homomorphism was originally proposed as a modification to the RSA cryptosystem, which describes the concept of preserving the computations between ciphertexts as presented by Rivest et al. [6]. This concept has led to numerous attempts by researchers around the world to search for a homomorphic scheme with various sets of operations. However, the main hindrance that researchers faced for about 30 years was the limitation of the number of operations that could be evaluated. During arithmetic, the noise level is mounting, and at some point, it becomes too large that is impossible to proceed with the operation without losing its correctness.

In 2009, Craig Gentry presented the idea of a Fully homomorphic encryption (FHE) scheme that is based on NTRU (N-th degree Truncated polynomial Ring Units), a lattice-based cryptosystem that is considered Somewhat homomorphic encryption (SHE). It can hold up to a limited number of operations—that is, a few multiplications and many additions. A public FHE scheme is composed of 4-tuple of probabilistic polynomial time protocols and algorithms namely (KeyGen, Enc, Eval and Dec):

- KeyGen$(\lambda) \rightarrow (pk, evk, sk)$: KeyGen is key generation algorithm that takes a security parameter $\lambda$ and outputs public encryption key $pk$, public evaluation key $evk$ and secret decryption key $sk$.
- Enc$(pk, m) \rightarrow ct$: Encryption algorithm Enc takes $pk$ and a vector of plaintext message $m \in \{0, 1\}$, and outputs a ciphertext $ct$.
- Eval$(evk, f, <ct_1, \cdots, ct_k>) \rightarrow ct^\ast$: Evaluation algorithm Eval takes a result of $k$-input function $f: \{0, 1\}^k \rightarrow \{0, 1\}$ and a set of $k$ ciphertexts $ct_1, \cdots, ct_k$. Then it outputs a new ciphertext $ct^\ast$ that is an encryption of $f(m_1, \cdots, m_k)$ where $ct_i \leftarrow \text{Enc}(pk, m)$ for $i = 1, \cdots, k$. 

References

[1] Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st ACM Symposium on Theory of Computing (STOC ’09). ACM, New York, NY, USA, pp. 169–178.

[2] van Dijk, M. et al. (2012). Fully homomorphic encryption over the integers. In Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS ’12). ACM, New York, NY, USA, pp. 333–342.

[3] Brakerski, Z. et al. (2014). Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Proceedings of the 2014 ACM Conference on Computer and Communications Security (CCS ’14). ACM, New York, NY, USA, pp. 928–940.
• Dec(sk, ct) → m∗ or ⊥: Decryption algorithm Dec takes sk and a ciphertext ct, and outputs a message m∗ ∈ {0, 1} if ct is result of Enc(pk, m) and pk is matched to sk, otherwise, it outputs ⊥.

The key idea proposed in Gentry’s work is bootstrapping, which is the process of refreshing a ciphertext and maintaining a lower level of noise to produce a new ciphertext so that more homomorphic operations can be evaluated. However, the bootstrapping procedure required adding many ciphertexts to the public key, and encrypting the individual bits (or coefficients) of the secret key. This resulted in very large public keys, and plaintext has to be bit-by-bit encrypted, increasing the capacity of the ciphertext increases at each step, making it too expensive in terms of computations.

2.2. Our Novel Approach Based on THFE Library

Various HE libraries (e.g., References [7–11]) have been suggested based on Gentry’s scheme [12]. Chillotti et al. released TFHE (The Fast Fully Homomorphic Encryption over the Torus) [13] library, which was designed from FHEW(Fastest Homomorphic Encryption in the West). It is based on both learning with errors (LWE) [14] assumption and its ring variant [15]. It significantly improves the performance of the bootstrapping operation—less than 0.1 s and facilitating an unlimited number of operations based on a gate-by-gate bootstrapping procedure. Moreover, the bootstrapping key size was scaled down from 1 GB to 24 MB to maintain the security level while reducing the overhead caused by noise.

The library supports the homomorphic evaluation of the binary gates (AND, OR, XOR, NOR, NAND, etc.), as well as the negation and the MUX gate, which can be used for various operations. Additionally, there is no restriction on the number of gates or their compositions, making it possible to perform any computations over encrypted data.

2.2.1. Bitwise Representation of Number

We performed homomorphic encryption of plaintext bits, that is, each bit within the plaintext is encrypted with the same key. Therefore, using our notation, given the plaintext a, the result of the encryption stage yields a ciphertext ct.a, which is an array containing encrypted bits. This is different from the mainstream approach since it is lattice-based and initiated from the input’s integer values. Specifically, integer-based FHE requires the encoding process through the rounding operation of plaintexts for the conversion of real number input. Through the rounding process, the outcome contains an error.

Our approach is more accurately designed to solve the problem of conversion from real number to integer. In general, floating-point number system guarantees a broader range of input; however, it can also lead to more complex algorithms when using HE scheme. HE gate operations take a significant amount of time compared to plaintext gate operations, thus using floating-point is less efficient. On the other hand, the major advantage of using a fixed-point representation is that it is a simple integer arithmetic operation with much less logic than floating-point, which improves performance by reducing the bootstrapping procedure on every encrypted bit. Therefore, it is crucial in FHE to reduce, so we only adopt a fixed-point number system in this paper.

We assigned r2, r2 − 1 and 1 bit for integer, decimals and signed bit, respectively. The encryption of each bit is designated in the same position as in the plaintext. In consequence, the values between plaintext and its corresponding ciphertext are precisely equal. By assigning a lengthy input, we guarantee higher accuracy with a tradeoff of the execution time, increasing dramatically. Typically, HE gate operations take a significant amount of time compared to plaintext gate operations. This is due to heavy loads of noise accumulated through HE gates. Therefore, realistically, we assigned the length of the input as 8, 16, and 32 for experimentation. Figure 1 shows an example of an 8-bit fixed-point representation of number 6.75 using our approach.
2.2.2. Bitwise Operation

We illustrate some of the critical concepts, such as our basic scheme and functions, to help us understand. However, in order to grasp fully-understanding of our approach, interested readers should refer to the previous works of our approach [4,5,16,17]. For the sake of the flow, we introduce the very basics of our approach.

The goal of HE operations is to construct a HE function that yields the encrypted result that matches the plaintext operation. For instance, result of plaintext addition, \(a + b\) should match the result of the HE addition, \(ct.a +_{HE} ct.b\) where \(+_{HE}\) represents HE addition. These basic HE operations are built from the combination of HE gate operations. We provide a simple illustration of our method in constructing HE absolute value operation to show the difference between plaintext and ciphertext.

Suppose we have a plaintext value \(a\) which is then converted to fixed point number constituting 0 and 1. If the goal is to derive an absolute value of \(a\) in plaintext condition, it is not hard to see that by using 2’s complement method, one can easily obtain the absolute value depending on the sign of \(a\). However, in the encrypted state, since the sign bit of ciphertext, \(ct.a[r - 1]\) where \(r\) is the length of the input, is encrypted or not known, we have to consider both cases where \(a\) is positive or negative. Thus, we have to perform both negative and positive cases of \(ct.a\). Consider the case when \(ct.a\) is positive, then the sign bit is \(Enc[0]\). Using NOT gate on the sign bit, our encrypted value becomes \(Enc[1]\) and call it \(ct.s\). With the value \(ct.s\), if we perform AND gate operations to every bits of \(ct.a\), we will obtain \(ct.a\) when \(ct.a\) is positive. If \(ct.a\) is negative, then \(ct.s\) is \(Enc[0]\) which, in the same manner, yields \(Enc[0]\) after executing AND gate operations to every bits of \(ct.a\).

Also in the negative case, we have to perform 2’s complement operation on \(ct.a\), and denote the outcome of operation to be \(ct.n\). Likewise, AND gate operations on every bits of \(ct.a\) and the sign bit, which is \(Enc[0]\) provides the result of negative case. Therefore, by adding two results of positive and negative cases, one can obtain an encrypted result of absolute value operation starting from the encrypted number of \(a\). In this way, we constructed various HE operations from HE bootstrapping gates. Now, we provide basic notations of gates and operations that are used in our work. Table 1 demonstrates our homomorphic operations and homomorphic functions used in this study.

### Table 1. The basic notation of homomorphic operations and homomorphic functions.

| Gate Operation       | Homomorphic Operation | Operation   | Homomorphic Function |
|----------------------|-----------------------|-------------|----------------------|
| AND gate             | bootsAND              | Addition    | HomAdd               |
| NOT gate             | bootsNOT              | Subtraction | HomSub               |
| OR gate              | bootsOR               | Multiplication | HomMulti             |
| XOR gate             | bootsXOR              | Division    | HomDiv               |
| CONSTANT trivial gate| bootsCONSTANT         | Absolute value | HomAbs               |
|                      |                       | 2’s compliment | HomTwosComp          |
|                      |                       | Mux         | HomMux               |
|                      |                       | Power       | HomPow               |
|                      |                       | Logarithm   | HomLog               |
|                      |                       | Matrix inverse | HomGJ               |
|                      |                       | Matrix Multiplication | HomMatMulti         |
2.2.3. HE Bitwise Logarithm

We introduced absolute value function in the previous section to demonstrate that there should be other measures in the plaintext algorithm in order to obtain the same result in the encrypted domain. Now, we suggested another key HE function logarithm used in derivation of evidence function. The details of the HE function are well explained in Reference [16]. In this paper, we briefly explain its method and show an example to illustrate our approach.

To design logarithm function in plaintext, we derive \( y = \log_2 x \) in the first stage. Next, we obtain the general form of logarithm with different base. We assume \( x \) and \( y \) to be real numbers s.t. \( x \in (1, 2) \) and \( y \in [0, 1) \). Then, \( y \) can be written as

\[
y = b_0 \times 2^{-1} + b_1 \times 2^{-1} + \cdots + b_{n-2} \times 2^{\xi-2}
\]

where binary representation of \( y \in [0, 1) \) is \( [0, 0, \cdots , 0, b_2^{-2}, \cdots , b_1, b_0] \). Since \( y = \log_2 x \) can be rewritten as \( x = 2^y \), we obtain nested-form of the equation as the following.

\[
x = 2^y = 2^{2^{-1} \times (b_0 + 2^{-1} \times (b_1 + (b_2 + \cdots )))}.
\]

By squaring \( x \), the formula becomes \( x^2 = 2^{b_0 + 2^{-1} \times (b_1 + (b_2 + \cdots ))} \). Since \( b_1 \) is either 0 or 1 and if \( x^2 \geq 2 \), \( b_1 \) is equal to 1 otherwise 0. For the case of \( b_1 = 1 \), we divide \( x^2 \) by 2. Following the procedure recursively provides bits \( b_1 \) or fractional part of \( y \).

So far, the above procedure is to obtain decimals of \( y \) that are less than 1. If \( y \geq 1 \), we perform the above procedure for the fractional part of \( y \) whereas we only count the index of the position of most significant bit of \( x \) for deriving integer of \( y \). In conclusion, summation of the two outcomes is the result of \( y = \log_2 x \).

This plaintext mechanism is useful to our scheme on the encrypted domain since the process of obtaining decimals mainly involves two fast-operations: comparison and shift. However, the approach to solve the problem should be different. Given the problem of \( ct.y = \log_2(\text{ct.x}) \) \( \text{ct.x} \), our initial work is to gain \( \text{ct.x} \in [\text{Enc}(1), \text{Enc}(2)] \). This is not an easy task compared to plaintext situation where we can easily shift bits of \( x \) to normalized \( x \) s.t. \( x \) lies in between 1 and 2. Instead, we make a detour using our HE functions to process normalization of a ciphertext. This is the first step that we perform that is different from plaintext domain. Algorithm 1 is the process of normalization of \( \text{ct.x} \).

Algorithm 1 \( \text{HomNorm} (\text{ct.x}) \)

1: for \( i = 0 \) to \( r - 2 \) do
2: \( \text{ct.d}_i \leftarrow \) Ciphertext arrays where \( \text{ct.d}_i[i] = \text{Enc}(1) \), else \( \text{Enc}(0) \)
3: \( \text{ct.s}_i \leftarrow \) Ciphertext arrays where \( \text{ct.s}_i = \frac{i}{2} - i \)
4: \( \text{ct.o}_i \leftarrow \text{HomCompareLarge}(\text{ct.x}, \text{ct.d}_i) \)
5: \( \text{ct.a} \leftarrow \text{HomShift} (\text{ct.x}) \) by \( \frac{i}{2} - i \)
6: end for
7: \( \text{ct.p} \leftarrow \) Add all the elements of \( \text{ct.o}_i[i] \)
8: \( \text{ct.p} \leftarrow \) Subtract \( \text{ct.p} \) by \( \text{Enc}(\frac{i}{2}) \)
9: \( \text{ct.e}_i \leftarrow \text{HomEqualCompare} (\text{ct.s}_i, \text{ct.p}) \)
10: \( \text{ct.r}_i \leftarrow \) Bitwise bootsAND \( (\text{ct.e}, \text{ct.a}) \)
11: \( \text{ct.r} \leftarrow \) Add all the elements of \( \text{ct.r}_i \)
12: return \( \text{ct.r} \)

Next, we proceed to obtain fractional bits of \( \text{ct.x} \) from the Algorithm 1. The problem in the encrypted domain is to decide whether the square of \( \text{ct.x} \) is greater than or equal to \( \text{Enc}(2) \). We use
Algorithm 2 HomSquareShift(\text{ct}.x)

\begin{enumerate}
\item for \( i = 0 \) to \( \frac{x}{2} - 1 \) do
\item \( \text{ct}.x^2 \leftarrow \text{HomMulti}(\text{ct}.x, \text{ct}.x) \)
\item \( \text{ct}.r[i] \leftarrow \text{ct}.x^2[\frac{x}{2} - 1] \)
\item \( \text{ct}.n[i] \leftarrow \text{bootsNOTct}.r[i] \)
\item \( \text{ct}.a[i] \leftarrow \text{bootsAND}(\text{ct}.r[i], \text{ct}.x^2) \)
\item \( \text{ct}.b[i] \leftarrow \text{bootsAND}(\text{ct}.n[i], \text{ct}.x^2) \)
\item \( \text{ct}.v[i] \leftarrow \text{HomAdd}(\text{ct}.a[i], \text{ct}.b[i]) \)
\end{enumerate}
\text{return } \text{ct}.v

Through Algorithms 1 and 2, we obtain the decimals of \( \text{ct}.x \). Since, the integer value of \( \text{ct}.x \) is the value of the position of the most significant value, it is \( \text{ct}.p \) at line 8 of the Algorithm 1. Therefore, \( \text{HomAdd}(\text{ct}.v, \text{ct}.p) \) is the result of HE logarithm of \( y = \log_2 x \). For the general result of \( \text{ct}.(\log_2 x) \), we first derive \( \text{ct}.(\log_2 a) \) to perform \( \frac{\text{ct}.(\log_2 x)}{\text{ct}.(\log_2 a)} \) which is equal to \( \text{ct}.(\log_2 x) \).

2.2.4. Time Complexity of Designed Homomorphic Operation

The experimental environment configuration settings are as follows. All computations ran on a computer with 32 GB RAM, Intel Core i7-8700 CPU 3.2 GHz (Intel, Santa Clara, CA, USA), Ubuntu 18.04 and we used TFHE library version 1.0.1 was used. We measured computational speeds of 1-bit basic gate operations for 1000 times in TFHE. All bootstrapping HE gates except NOT and MUX gates took about 10.8 ms to evaluate. The MUX gate took 20.9 ms and NOT gate takes 0.000154 ms, which is significantly lower than other gates. Therefore, we ignored the speed of NOT gate in calculating execution. We denoted time complexity of all binary gates as \( T_B \) except Mux gate which was denoted as \( T_X \). Table 2 shows performance time for each operation in detail.

| Operation          | 8-bit (sec) | 16-bit (sec) | 32-bit (sec) | \( r \)-bit |
|--------------------|-------------|-------------|--------------|--------------|
| Addition           | 0.38        | 0.83        | 1.69         | \((5r - 3)T_B\) |
| Subtraction        | 0.36        | 0.82        | 1.67         | \((5r - 3)T_B\) |
| Multiplication     | 4.85        | 17.96       | 69.07        | \((6r^2 + 4)T_B + 4rT_X\) |
| Division           | 6.89        | 27.48       | 109.86       | \((8r^2 - 4r + 4)T_B + (r^2 + 2r)T_X\) |
| Absolute value     | 0.33        | 0.64        | 1.35         | \(2rT_B + rT_X\) |
| 2's compliment     | 0.14        | 0.31        | 0.66         | \((2r - 3)T_B\) |
| Power              | 8.78        | 34.09       | 135.53       | \((12r^2 + 5r + 5)T_B + 8rT_X\) |
| Logarithm          | 23.85       | 152.55      | 1123.13      | \(3r^3T_B + 55rT_B - 33T_B + 2rT_X\) |

As more bits are assigned to the input value, execution time increases dramatically because it involves many operations, including addition, subtraction and comparison that increase linearly with the length of data and their interactions.

2.3. Model Selection

Most statistical inference approaches for analyzing the data aim to make “good” predictions. However, “good” predictions we cannot be obtained if the proper models are not chosen in...
the first place. Worse, there exists no such perfect model that is generally suitable for any data. Therefore, a crucial step in data analysis is to consider a set of candidate models and then select the most appropriate one. This is called model selection, one of the most important and essential steps to obtain stably accurate results in data analysis. Also, the model selection can be divided into two main branches because the meaning of the ‘model’ is interpreted differently in various fields.

- **A Model regarded as an algorithm**: It selects the most appropriate process among different machine learning approaches—for example, support vector machine (SVM), KNN, logistic regression, and so forth.

- **A Model regarded as a complexity**: It selects among different hyperparameters in a set of features for the same machine learning approach—for example, determining an order between polynomial models in linear regression.

In this study, we focus on model selection with various complexities, by adapting numerical solutions for the regression analysis.

Polynomial regression is a type of linear regression that refers to the relationship between the independent variable \( x \) and dependent variable \( y \) modeled as \( M \)th degree polynomial:

\[
y(x, \theta) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_M x^M = \sum_{i=0}^{M} \theta_i x^i.
\]  

In Equation (3), \( \theta \) is a set of polynomial coefficients, and it is determined by fitting the polynomial to the training data by minimizing the errors. \( M \) is the order of the polynomial, so we need to estimate the optimal order \( M^* \) to obtain the best prediction results.

Generally, model selection criteria are based on an estimator of generalization performance evaluated over the data. The models following \( M \) can be evaluated as estimated errors that are decomposed by bias and variance. If the model is too simple to describe the data, there is a high probability of a high bias and low variance called under-fitting. On the contrary, over-fitting occurs when a complex model has low bias and high variance. In machine learning, an over-fitted model may fit perfectly into training data but may not be suitable for new data.

To avoid under-fitting and over-fitting, one must select an appropriate model with optimal complexity. Thus, we need to find a way to determine the right value between models with different complexity. A considerable number of selection procedures were proposed in the literature, for example, the AIC(Akaike Information Criterion) method [18], the Cp method [19], the BIC(Bayesian Information Criterion) method [20], the Cross-Validation (CV) method [21], Bayesian evidence methods [22], and so forth. In this study, we developed two model selection algorithms that can work in encrypted domain for the Cross-validation (CV) and Bayesian model selection.

### 2.3.1. Cross Validation

CV is one of the most widely used methods to evaluate predictive performances of a candidate model in model selection. CV estimates the expected error, and does not require the models to be parametric. It makes full use of data without leaking information into the training phase. Regarding data splitting, some of the data is used for fitting each model to be compared, and the rest of the data is used to measure the predictive performances of the models by the validation errors. Through these processes, the model with the best overall performance is selected.

### 2.3.2. Bayesian Model Selection

Bayesian model selection is also a well-known approach for choosing an appropriate model. The Bayesian paradigm offers a principle approach that addresses the model choice by considering the posterior probability given a model. The Bayesian view of model comparison includes the consistent application of the rules of sum and product of probabilities, as well as the use of probabilities that
represent uncertainty in model comparison [23]. More precisely, suppose that the comparing models can be enumerated and indexed by the set \( \{ M_i : i = 1, \cdots, N \} \). It represents the probability distribution for the observed data \( D \) generated by model \( M_i \). The posterior distribution for a set of model parameters is:

\[
p(\theta | D, M_i) = \frac{p(D|\theta, M_i)p(\theta|M_i)}{p(D|M_i)},
\]

where \( p(D|\theta, M_i) \) is the likelihood and \( p(\theta|M_i) \) represents the prior distribution of the parameters of \( M_i \). The model evidence for model \( M_i \), based on the product and sum rule as,

\[
p(D|M_i) = \int p(D|\theta, M_i)p(\theta|M_i)d\theta
\]

represents the preference shown by the data for different models, and it is also called marginal likelihood because it can be viewed as a likelihood function over the space of models in which the parameters have been marginalized out.

3. Problems

Our approach is different from the mainstream of the homomorphic encryption scheme. We addressed several critical underlying problems that the current research holds.

3.1. Model Selection in Homomorphic Encryption

First, a majority of the current literatures that links HE to machine learning are confined to solving a “modeled” problem. Based on their HE operations, they look for a suitable data distribution that can show high performance of their functions. Kim et al. [1] and Aono et al. [3] achieved high performance of HE logistic regression on encrypted data in terms of accuracy and speed, using iDASH dataset and Pima diabetes dataset. But, they achieved the high performance mainly because they choose the appropriate data distribution that their performance of HE logistic regression can significantly benefit from.

However, if it were to perform logistic regression on unknown(or encrypted) data distribution, it is unlikely that the same results would be achieved. Therefore, we proposed model selection as a first step to decide the appropriate model for fitting unknown data distribution. We claim that our approach should be more fundamental because it provides an adequate model for encrypted data distribution preceded in the first round.

Furthermore, implementing an inverse of the matrix multiplication(and division) in the model selection process is a particularly challenging task in HE. Reference [24] discusses inverse matrix operation through the Schulz algorithm under the HE domain, however, their scheme does not have a division operation. Therefore, they inevitably must adopt an approximation method, in order to perform the inverse matrix operation during the model selection process. In this process, the error increases linearly through iterations. However, our approach, Gauss-Jordan elimination, based on bitwise operations mentioned in Section 2, can accurately derive the inverse matrix results. The proposed approach is the Gauss-Jordan elimination. The most useful property of the method is that reduced row echelon form is unique, which means row-reduction on a matrix will produce the same answer regardless of how the row operation are performed, which is essential to get a definite answer in the encrypted state. In the following, we propose homomorphic matrix operations to discuss a method of obtaining an inverse matrix.

Through our previous works [5,16], we designed nonlinear functions such as logarithm and exponential functions that can be applied more fundamentally and used them in the Bayesian model selection process. However, the mainstream approach does not support nonlinear functions such as exponential function and logarithm function [4,25]. Instead, they approximate these types of functions in terms of Taylor series or least square approximation in the form of summation of polynomials.
This approach is only limited to solving logistic regression problem and cannot be used to design other types of algorithms.

3.2. Problems of Cross-Validation, an Existing Model Selection Method

In the previous study [17], we used CV as an evaluation method for model selection. When performing CV using two sets of factors—the training and the test sets, the number of folds \( k \) affects the performance of our approach in terms of speed. It is known that the larger the value of \( k \), the more accurate the results. However, in our research, the cost of HE operations’ execution time is significantly high, balancing accuracy and speed. Thus, we need other methods that offset the shortcoming of computation time for our HE functions.

To overcome these shortcomings, we proposed model selection from a Bayesian perspective. This method avoids the over-fitting problem by performing marginalization based on the parameter instead of the point estimation of the model’s parameter values. In this case, the models can be directly compared based on the training set, hence a verification set is not required, which avoids multiple training for each model needed to implement the CV method, reducing computational time in the long run.

4. Methods

4.1. Homomorphic Matrix Operations

We begin with matrix multiplication and the inverse of a square matrix before implementing the model selection algorithm.

4.1.1. Homomorphic Matrix Multiplication

As more complex algorithms require many matrix operations have grown, performing matrix operations on HE has become equally important. Let \( A = [a_{ik}] \) and \( B = [b_{kj}] \) be two square matrices of size \( n \times n \). Using the component-wise definition of matrix multiplication, \( C = [c_{ij}] \) can be expressed as

\[
C_{ij} = \sum_{k=1}^{n} A_{ik} \otimes B_{kj} \otimes : \text{HomMulti} \quad \forall i, j = 1, 2, \ldots n.
\]  

(6)

The matrix multiplication process requires multiplication and addition of its entries. First, homomorphic multiplication method is introduced in Algorithm 3.

When entries of encrypted matrices \( ct.A \) and \( ct.B \) enter the input value, multiplication runs mainly on addition. Specifically, assuming the multiplication is between \( r \) bits, the multiplication operation converted to \( r \) number of additions using only AND gate and Right-Shift operations. Therefore, when we execute multiplication of entries of \( r \) bit, the result is obtained by \( j \) times of right shift and \( i \in \{0, 1, \ldots, r - 2\} \) times of summation. To obtain results regardless of the sign of the given data, we calculate the positive product using absolute value operation and then perform 2’s complement operation on the result according to its sign. Finally, the integration of part-by-part calculations is carried out, and the calculated value shown as a result. The outcome is double the length of the input, therefore, we limit the boundary of result to \( r \) to obtain an adjusted result in which the lengths of the partial and integer parts are equal to our setting.

The Algorithm 4 shows the addition operation using encrypted input values, which is the last part of the dot product of the matrix multiplication operation. As with the method in the plaintext, is based on a full adder scheme. Addition can be designed simply due to basic gate operations with bootstrapping, so only 2 XOR, 2 AND, and 1 OR gates are used in this process.

Thus, Algorithm 5 outputs the multiplication of matrices \( ct.A \) and \( ct.B \) through homomorphic multiplication and homomorphic addition.
Algorithm 3 HomMulti(ct.a, ct.b, r): Homomorphic multiplication for matrix operation

1: ct.\dot{a} ← HomAbs(ct.a), ct.\dot{b} ← HomAbs(ct.b)
2: for i = 1 to r do
3: 
4: end for
5: for i = 1 to r − 2 do
6:  
7: end for
8: if i < 2^L − 1 then
9: 
10: end if
11: return
12: if i = 2^L − 1 then
13: 
14: end if
15: else
16: 
17: end if
18: return
19: for i = 1 to r do
20: 
21: end for
22: ct.f ← HomTwosComp(ct.d)
23: ct.g ← bootsXOR(ct.a, ct.b)
24: ct.h ← bootsNOT(ct.g)
25: for i = 1 to r do
26: 
27: end for
28: ct.o ← bootsAND(ct.d, ct.h)
29: ct.p ← bootsAND(ct.f, ct.g)
30: end for
31: ct.q ← HomAdd(ct.o, ct.p)
32: return ct.q

Algorithm 4 HomAdd(ct.a, ct.b, r): Homomorphic addition for matrix operation

1: ct.a[0] ← bootsCONSTANT[0]
2: for i = 0 to r − 1 do
3: 
4: end for
5: ct.m[0] ← bootsXOR(ct.a[i], ct.b[i])
6: ct.m[1] ← bootsAND(ct.a[i], ct.b[i])
7: ct.c[i] ← bootsXOR(ct.m[0], ct.c[i])
8: ct.m[0] ← bootsXOR(ct.m[0], ct.c[i])
9: ct.m[1] ← bootsXOR(ct.m[0], ct.m[1])
10: ct.c[r − 1] ← bootsXOR(ct.a[r − 1], ct.b[r − 1])
11: return ct.c
Algorithm 5 HomMatMulti(ct.A, ct.B)

1: ct.C ← Enc[0]
2: for i = 1 to n do
3:     for j = 1 to n do
4:         for k = 1 to n do
5:             ct.cij ← HomMulti(ct.Aik, ct.Bkj)
6:             ct.C ← HomAdd(ct.C, ct.cij)
7:         end for
8:     end for
9: end for
10: Return ct.C

The performance evaluation of this approach is as follows. The order of square matrix and the length of data are set as a factor of matrix inversion. The time complexity $T_{matmulti}$ for homomorphic matrix multiplication can be represented as follows:

$$T_{matmulti} = (8r^2 + r + 1)n^3T_M + (r^2 + 2r)n^3T_X.$$  \hfill (7)

4.1.2. Accurate Homomorphic Matrix Inversion

If matrix $W$ is invertible, then every equation $Wx = p$ has a unique solution. Gauss-Jordan algorithm computes $W^{-1}$ given $W$. There are many ways to find the inverse matrix. Examples include such as Cramer’s rule, Gaussian elimination, and Gauss-Jordan elimination. Cramer’s rule is computationally intensive compared to other methods, and roundoff error may become significant on large problems with non-integer coefficients. Gauss-Jordan elimination is a modification of the Gaussian elimination that reduces the computations involved in back substitution by performing additional row operations to transform the matrix from echelon form to reduced row echelon form (rref). In general, the Gauss-Jordan method may require a more significant number of arithmetic operations as the size of matrix increases, however, the rref of a matrix has the advantage of guaranteeing uniqueness. The elimination process consists of three possible steps called elementary row operations: swap two rows, scale a row, and subtract a multiple of a row from another. Before we construct our algorithm, we first figure out how it works. If $I$ is an identity matrix, the equation can be express as follows.

$$Wx = IP.$$  \hfill (8)

The equation is still same if we multiply both terms by a non-singlar matrix $V_0$:

$$V_0Wx = V_0IP.$$  \hfill (9)

The trick of the Gauss-Jordan elimination consists of finding a series of matrices $V_0, V_1, ..., V_{n-1}$ so that

$$V_{n-1}...V_1V_0Wx = V_{n-1}...V_1V_0IP = x.$$  \hfill (10)

The process expression must be true for $P$ so, $x$ is the solution of $Wx = P$, by definition, $V_{n-1}...V_1V_0I \equiv W^{-1}$. Thus, given $W$, the Gauss-Jordan algorithm works exactly this way, it computes $W^{-1}$.

Algorithm 6 demonstrates the process of Gauss-Jordan elimination in an encrypted state. First, we essentially chose the largest absolute value element on an unvisited column and row using HomAbs operation. Next, we used HomMux in order to switch the $k$th and $i^*$th rows. HomMux(ct.S, ct.a, ct.b) means Mux gate that homomorphically outputs either the message of ct.a or ct.b relying on the boolean value of ct.S, without decrypting any of the ciphertexts. If ct.S = Enc[1], it indicates ct.a otherwise ct.b.
Algorithm 6 ct.\(W^{-1}\)=HomGJ(ct.\(W\)): Homomorphic Matrix Inversion using Gauss-Jordan Elimination

Require: Encrypted Matrix ct.\(W_{n \times n}\)
Ensure: Encrypted Inverse Matrix ct.\(W_{n \times n}^{-1}\)

1: for each row \(k\) do
2: \(ct.i^* \leftarrow \text{argmax}_{k \leq i \leq n} \text{HomAbs}(ct.W_{ik})\)
3: if \(ct.W_{i^*k} = 0\) then
4: Matrix is not invertile
5: end if
6: Swap rows \(k\) and \(i^*\) \(\leftarrow \text{HomMux}(ct.k, ct.i^*, ct.temp)\)
7: for each row \(j\) below \(k\) (i.e., \(j = k + 1, \ldots, n\)) do
8: \(ct.\tau \leftarrow \text{HomDiv}(ct.W_{jk}, ct.W_{kk})\)
9: \(ct.W_{K} \leftarrow \text{HomMulti}(ct.\tau, ct.W_{k})\)
10: \(ct.W_{j} \leftarrow \text{HomSubt}(ct.W_{j}, ct.W_{K})\)
11: end for
12: end for
13: for each row \(k = n, \ldots, 1\) (i.e., in reverse) do
14: \(ct.W_{k} \leftarrow \text{HomDiv}(ct.W_{k}, ct.W_{kk})\)
15: for each row \(j\) above \(k\) (i.e., \(j = k - 1, \ldots, 1\)) do
16: \(ct.\tau \leftarrow \text{HomDiv}(ct.W_{jk}, ct.W_{kk})\)
17: \(ct.W_{K} \leftarrow \text{HomMulti}(ct.\tau, ct.W_{k})\)
18: \(ct.W_{j} \leftarrow \text{HomSubt}(ct.W_{j}, ct.W_{K})\)
19: end for
20: end for
21: Return ct.\(W^{-1}\)

Then Gauss elimination to eliminate all components below the diagonal, followed by two phases to eliminate all components above the diagonal. Homomorphic operations are responsible for these processes.

- **phase1**: Convert ct.\(W\) into the row echelon form by eliminating all ct.\(W_{jk}\) in equation below the diagonal. By this phase of elimination, all elements below the diagonal are eliminated, and ct.\(W\) becomes an upper triangular matrix.

- **phase2**: Convert the ref obtained into a diagonal matrix, called rref, by eliminating all ct.\(W_{jk}\) in equation ct.\(W_{j}\) above the diagonal. By this phase of elimination, all elements above the diagonal are eliminated, and ct.\(W\) becomes a diagonal matrix, which can be turned into ct.\(I\).

The performance evaluation of this approach is as follows. The order of square matrix and the length of data are set as a factor of matrix inversion. The time complexity \(T_{\text{inverse}}\) for homomorphic matrix inverse can be represented as follows:

\[
T_{\text{inverse}} = \left[\left(2r^2 + \frac{7}{3}r + \frac{10}{3}\right)n^3 + \left(3r^2 + 4r + 7\right)n^2 + \left(3r^2 - \frac{19}{3}r - \frac{10}{3}\right)n\right]T_B + \frac{1}{3}n(2n^2 + 3n + 7)T_X.
\]

(11)

4.2. Design of Homomorphic Model Selection

4.2.1. Homomorphic Cross Validation Approach

The method proposed using the Equation (3) to construct the model selection algorithm is polynomial regression. Given a set of \(N\) points \((x_t, y_t)_{t=0}^{N-1}\) for all \(n\), the goal is to fit the data with a
polynomial of a degree $M$ and the least-square method figure out the coefficients $\theta_i$ to minimize the squared error function

$$R(\theta_0, \ldots, \theta_M) = \sum_{i=0}^{N} \left( \sum_{i=0}^{M} \theta_i x_i^j - y_i \right)^2.$$  \hfill (12)

The coefficient gets its global minimum when the gradient of $R$ is zero, or the partial derivatives of $R$ must be zero for $0 \leq j \leq M$:

$$\frac{\partial R}{\partial \theta_j} = 2 \sum_{i=0}^{N-1} \left( \sum_{i=0}^{M} \theta_i x_i^j - y_i \right) x_i^j = 0. \hfill (13)$$

It simplifies to

$$\sum_{i=0}^{M} \left( \sum_{i=0}^{N-1} x_i^j \right) \theta_i = \sum_{i=0}^{N-1} x_i^j y_i. \hfill (14)$$

This equation is a linear system of $(M+1)$ equations in $(M+1)$ unknown coefficients and is of the form

$$X^T \theta = Y,$$  \hfill (15)

where $X=[a_{ij}]$ is an $N \times (M+1)$ matrix whose general entry is $a_{ij} = x_i^j$, and $\theta = [\theta_j]$ is a $(M+1) \times 1$ column vector and $y=[y_i]$ is an $N \times 1$ column vector. The coefficients of the polynomial are

$$X^T X = \begin{bmatrix} \sum_{i=0}^{N-1} 1 & \sum_{i=0}^{N-1} x_i & \cdots & \sum_{i=0}^{N-1} x_i^M \\ \sum_{i=0}^{N-1} x_i & \sum_{i=0}^{N-1} x_i^2 & \cdots & \sum_{i=0}^{N-1} x_i^{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{N-1} x_i^M & \sum_{i=0}^{N-1} x_i^{M+1} & \cdots & \sum_{i=0}^{N-1} x_i^{2M} \end{bmatrix}, \quad Y = \begin{bmatrix} \sum_{i=0}^{N-1} y_i \\ \vdots \\ \sum_{i=0}^{N-1} x_i y_i \end{bmatrix}. \hfill (16)$$

The assumption that $x_k$ are increasing guarantees that $X^T X$ is invertible, so the least square estimator of $\theta$ is given by

$$\theta = (X^T X)^{-1} X^T Y \hfill (17)$$

and Algorithm 7 describes the details of the equation. $K$-fold CV method is used to evaluate the model selection performance. The values of the coefficients $\theta$ is determined by fitting polynomial to the data. This can be obtained by minimizing the error function called mean – square – error (MSE). It is widely used for the sum of the squares of the errors between the prediction $y(x_i, \theta)$ for each data point $x_i$ and the corresponding target values $z_i$,

$$R(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left( y(x_i, \theta) - z_i \right)^2 \hfill (18)$$

For $K$ subsets, we compute the SSE of each subset, $R(\theta)_k$, on the training set and record the total error on the validation set then compute the average error over all folds. Finally, we choose the value of the parameter that minimizes the average error.

$$\hat{R}(\theta) = \frac{1}{K} \sum_{k=1}^{K} R(\theta)_k \hfill (19)$$

Based on TFHE, we implement the $K$ – fold homomorphic CV for the model selection. We first initialize part of training and test error matrix elements to zero and updated our elements using the model selection algorithm as a polynomial regression function. During this process, the value of
the elements are computed through homomorphic operations and the solution of matrix inverse 
\((ct.X^T ct.X)\) is multiplied by the vector \(ct.y\). After \(K\)-fold CV, final return value stands for average MSE 
at each polynomial degree.

**Algorithm 7 Homomorphic Regression Coefficients for each model order**

**Require:** Encrypted input features \(ct.w, ct.y\)

1. \(ct.X \leftarrow Enc[0]\)
2. for \(i = 1\) to \(\text{length}(x)\) do
3. \(\text{for } j = 1\) to \(\text{order} + 1\) do
4. \(ct.X \leftarrow \text{HomPow}(ct.x(i), (j - 1))\)
5. end for
6. end for
7. \(ct.A \leftarrow \text{HomGJ}(\text{HomMatMulti}(ct.X^T, ct.X))\)
8. \(ct.B \leftarrow \text{HomMatMulti}(ct.A, ct.X^T)\)
9. \(ct.\theta \leftarrow \text{HomMatMulti}(ct.B, ct.y)\)
10. Return \(ct.\theta\)

4.2.2. Homomorphic Bayesian Model Selection Approach

Another way to overcome over-fitting associated with maximum likelihood is marginalizing 
(summing or integrating) over the model parameters instead of making point estimates of 
parameters in the model and \(N\) is the number of data. \(R(\theta)\) is defined as

\[
R(\theta) = \beta R_D(\theta) + \alpha R_W(\theta) = R(\mu_N) + \frac{1}{2}(\theta - \mu_N)^T L (\theta - \mu_N) \tag{23}
\]

and \(R(\mu_N)\) is together with

\[
R(\mu_N) = \frac{\beta}{2} \| z - H \mu_N \|^2 + \frac{\alpha}{2} \mu_N^T \mu_N, \tag{24}
\]

\[p(z|\theta, \beta)\) is defined by \(\mathcal{N}(z|\theta, \beta^{-1})\) and \(p(\theta|z, \alpha, \beta)\) is defined by \(\mathcal{N}(\theta|\mu_N, V_N)\) with \(\mu_N = \beta V_N H^T z\) and \(V_N^{-1} = a I + \beta H^T H\). According to Bayes’ theorem in the previous section, the posterior distribution for hyperparameter is given by

\[
p(\theta|z, \alpha, \beta) \propto p(z|\theta, \beta)p(\theta|\alpha, \beta)\]

\[
(21)
\]

The evidence function \(p(z|\alpha, \beta)\) is obtained by integrating over the parameters, so

\[
p(z|\alpha, \beta) = \int \int p(z|\theta, \beta)p(\theta|\alpha) d\theta d\beta = \left( \frac{\beta}{2\pi} \right)^{N/2} \left( \frac{\alpha}{2\pi} \right)^{M/2} \int \exp\{ -R(\theta) \} d\theta, \tag{22}
\]
where $H$ is the design matrix. The right hand side of the equation shows the application of the square over parameter through recognition as being equal to a constant of proportionality to the regularized SSE function. Hessian matrix $L$ is same as $V_N^{-1}$ and it implies that a square matrix of second-order partial derivatives of a scalar-valued function, $L = \nabla \nabla R(\theta)$. $\mu_N$ is also given by $\mu_N = \beta L^{-1}H^Tz$.

Based on the equation above, the integral over parameter can evaluate by appealing to the standard result for the normalization coefficient of a multivariate Gaussian. Moreover, the log of the marginal likelihood created using Equation (22), which is the required expression for the evidence function.

The optimal model complexity determined by the maximum evidence is given through the equation above. Based on TFHE, the evidence approximation for the model selection is presented in Algorithm 8.

**Algorithm 8 Homomorphic Bayesian Model Selection for Linear Regression**

**Require:** Encrypted input features $ct.x$, $ct.z$ and observations $a$ and $\beta$

**Ensure:** ct.$m^\star = \arg\max_M [ct.Evi(m)]$

1. ct.$H \leftarrow \text{Enc}[0]$
2. **for** $M = 1$ to $M_{\text{max}}$ **do**
3.  
4.  
5.  
6.  
7.  
8.  
9.  
10.  
11.  
12.  
13.  

---

5. Implementation

In this section, we present the results of the model selection for encrypted data. Two types of datasets were used to implement the model selection algorithm. The input value is encrypted with 16-bits in length and used for calculation. In the first experiment, we confirmed that the HE Gauss-Jordan Elimination algorithm works correctly with various HE functions. To expand the scope, we tested the algorithm on a real-world dataset from Kaggle and used it in our experiment to see that the algorithm works well for many features.

5.1. Toy Dataset

We use artificially created dataset comprising 20 observations of $x$, together with corresponding observations of the values of $y$ to check performance and evaluation of our algorithm. Figure 2 shows a plot of a training set comprising $N = 10$ data points and test set comprising $N = 10$ data points respectively.

Then we evaluated through two different ways to achieve good generalization by making accurate predictions for new data. In this process, we use root mean square error (RMSE) as opposed to MSE because the square root ensures that RMSE is measured on the same scale as the variable $y$, so it is easy to interpret. Figure 3 shows the results of implementing two approaches of our HE model selection.
Figure 2. Plot of a training and test dataset. The black curve shows the function used to generate the data.

Figure 3. (a): Graphs of the root-mean-square error (RMSE) evaluated on the training set and independent test set for various values of M. (b): Plot of a model evidence for various values of M.

Toy dataset is too small to perform CV procedure, therefore, we focused on training and test error of each order of $M$. Values of $M$ in the range $3 \leq M \leq 7$, give small values for the test set error in the Figure 3a. For 9th polynomial, the regression model fits exactly to all data points, therefore, RMSE error converges to 0 in training dataset. In this case, the values of 6th provide a smaller error, making it is the best predictor of new data when $M$ is 6.

In the evidence approach, we fixed $\alpha = 5 \times 10^{-3}$, $\beta = 1.5$ and performed our algorithm and when $M = 3$, giving the best evidence of all polynomials. Additional increases in the value of $M$ make only small improvements in the fit to the data but increases the complexity penalty, which decreases the overall evidence value. Table 3 shows the evidence values for each order in the plaintext and ciphertext through the same experimental method. The difference in the evidence values obtained from the two results indicates that there are not many errors for the experiment in the encrypted state. The error occurs in the process of converting a real number plaintext into a ciphertext, which can be reduced by increasing the number of input $r – bits$. 
5.2. Body Mass Index (BMI) Dataset

BMI, consisting of various sensors including ultra-sonic, is a measurement of a person’s weight based on his/her height. BMI is calculated by dividing the weight of the body in kilograms by the square of height in meters, and the result of BMI indicates if that person is obese, overweight, regular, or underweight. BMI classification does not depend on age, therefore, we determined the tendency between them through regression. For experimental evaluation, we use “Medical Cost Personal Datasets” from Kaggle [26]. The dataset consists of age, sex, BMI, number of children, region, and so on, however, in this experiment, we used only two types of age and BMI to find out the correlation between them. The total number of data for each type is 1339, and we sampled 1000 data respectively to proceed model selection.

Before starting the homomorphic CV approach, we standardized our dataset because variables measured at different scales do not contribute equally to the analysis. Our dataset contains features highly varying in magnitudes, units, and ranges. Standardization (or Z-score normalization) is the process of rescaling the features to have the properties of a gaussian distribution with \( \mu = 0 \) and \( \sigma = 1 \), where \( \mu \) is the mean and \( \sigma \) is the standard deviation from the mean. Figure 4a shows 9th order of polynomial regression for standardized data and Figure 4b shows standardized data distribution of each dataset.

|          | M = 0  | M = 1  | M = 2  | M = 3  | M = 4  | M = 5  | M = 6  | M = 7  | M = 8  | M = 9  |
|----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Plaintext| –89.47 | –92.83 | –64.04 | –35.75 | –40.71 | –41.93 | –46.09 | –50.34 | –56.01 | –62.34 |
| 16-bit Ciphertext | –89.88 | –93.26 | –65.47 | –37.11 | –42.34 | –43.78 | –48.11 | –52.78 | –58.56 | –65.03 |

Table 3. Comparison of error between plaintext and ciphertext evidence.

Figure 4. (a): Graphs of the 9th polynomial regression with 100 data (b): Plot of 1000 data sets for use in training set, test set, and validation set divided by 6:2:2.

For training and test sets of randomly selected age and BMI values, we assumed that each set represents an entire dataset. We performed 5 – fold CV technique to estimate RMSE. A polynomial problem associated with implementing nth order regression on the computer is that the normal equations tend to be ill-conditioned, especially for higher-order versions. Thus, as with the toy dataset, we varied complexity by using our algorithm that range in model order from 0 (least complex) to 9 (most complex).

Figure 5a shows training, test, validation, and 5 – fold CV results. The validation dataset is a part of the training set that is sacrificed to evaluate the performance of the model when the CV method is not used. In order to use all the data through the 5 – fold method, we put the training, test, and validation set together and divide them into five equally partitions, and use sequentially four of them as a training set and the remaining one as a test set. The performance measure reported
by \(5-\)fold CV is then the average of the values computed in the loop. We repeated the procedure 50 times to obtain the total RMSE values of each dataset. The boxplot in Figure 5b reveals the tendency of distribution for the dataset. In the CV approach, the best estimator corresponds to a polynomial model of order 4.

Figure 5. (a): Graphs of RMSE of each order in 600 training, 200 test, 200 validation and cross validation dataset. Cross validation performs with 800 training(training + validation) dataset and 200 test dataset. (b): Boxplot of total sum of RMSE on the dataset.

To maximize model evidence over model order \(M\), we set up hyperparameter \(\alpha = 1 \times 10^{-6}\), \(\beta = 18.3\). As with the CV approach, we figure out the evidence approximation up to the 9th order and when the order of polynomial model was 2, it corresponded to the best estimator corresponds as with Figure 6. We measured the time performance of each HE model selection approach and the execution times were 1764 and 480 min, respectively.

Figure 6. Plot of a model evidence for various values of \(M\).

6. Limitation

Although our proposed approach improves computational resources, it still has a few limitations. In an encrypted domain, each operation speed is very slow as the memory space is large. However, this problem can be solved by accelerating calculations with many state-of-art technologies. Using FPGA(Field Programmable Gate Array), ASIC(Application Specific Integrated Circuit), GPU(Graphic Processing Unit) [27–29], the computations can be offloaded to the HE co-processor.
It improves the performance of certain HE tasks by performing directly in hardware. The CPU vector computation extension using SIMD (Single Instruction Multiple Data) [30] also improves efficiency. Therefore, we can combine these approaches into a hybrid system to create a fully optimized solution.

7. Conclusions

To date, privacy-preserving machine learning is one of the key research areas in computer security and machine learning. Homomorphic encryption is one of the most theoretically sound approaches for privacy-preserving machine learning and data mining. However, most algorithms made using homomorphic encryption do not consider the model’s complexity and stability. It is not straightforward to evaluate the stability of the model given data in the homomorphic encryption since all data are encrypted. Therefore, homomorphic model selection methods are an essential tool for data analysis in the encrypted domain, especially for big data obtained from biomedical sensors. If the model is not properly chosen, the results cannot be trusted, and it can be a disaster for the biomedical domain. Therefore, homomorphic model selection algorithms that provide both privacy and stability.

Therefore, we introduced two different methods for constructing the homomorphic model selection—homomorphic Cross-Validation (HCV) and homomorphic Bayesian model selection (HBMS). In addition, both model selection approaches require homomorphic matrix inversion. We designed and developed a homomorphic matrix inversion using Gauss-Jordan elimination in the homomorphic encryption scheme, specially, using a homomorphic mux operation. Finally, based on the results of the two different approaches, we confirmed that the HBMS approach is more efficient in encrypted domains.

Author Contributions: Conceptualization, M.Y.H. and J.W.Y.; methodology, M.Y.H.; software, M.Y.H. and J.S.Y.; validation, M.Y.H.; formal analysis, M.Y.H. and J.S.Y.; investigation, M.Y.H.; data curation, M.Y.H.; writing–original draft preparation, M.Y.H.; writing–review and editing, J.S.Y. and J.W.Y.; visualization, M.Y.H.; supervision, J.W.Y.; project administration, J.W.Y.; funding acquisition, J.W.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This study was supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. 2017-0-00545).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kim, A.; Song, Y.; Kim, M.; Lee, L.; Cheon, J.H. Logistic regression model training based on the approximate homomorphic encryption. BMC Med. Genom. 2018, 11, 83. [CrossRef] [PubMed]
2. Giacomelli, I.; Jha, S.; Joye, M.; Page, C.D.; Yoon, K. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In Proceedings of the International Conference on Applied Cryptography and Network Security, Leuven, Belgium, 2–4 July 2018; pp. 243-261.
3. Aono, Y.; Hayashi, T.; Trieu Phong, L.; Wang, L. Scalable and secure logistic regression via homomorphic encryption. In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, New Orleans, LA, USA, 9–11 March 2016; pp. 142-144.
4. Song, B.K.; Yoo, J.S.; Hong, M.Y.; Yoon, J.W. A Bitwise Design and Implementation for Privacy-Preserving Data Mining. Secur. Commun. Netw. 2019, 2019, 3648671. [CrossRef]
5. Yoo, J.S.; Hwang, J.H.; Song, B.K.; Yoon, J.W. A Bitwise Logistic Regression Using Binary Approximation and Real Number Division in Homomorphic Encryption Scheme. In Proceedings of the International Conference on Information Security Practice and Experience, Tokyo, Japan, 25–27 September 2018; pp. 20–40.
6. Rivest, R.L.; Shamir, A.; Leonard, A. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 1978, 21, 120–126. [CrossRef]
7. Halevi, S.; Shoup, V. Algorithms in HElib. In Proceedings of the Advances in Cryptology, Santa Barbara, CA, USA, 17–21 August 2014; Volume 8616, pp. 554–571.
8. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the Advances in Cryptology—ASIACRYPT, Hong Kong, China, 3–7 December 2017; Volume 10624, pp. 409–437.
9. Leo, D.; Daniele, M. FHEW: Bootstrapping homomorphic encryption in less than a second. In Proceedings of the Advances in Cryptology—EUROCRYPT, Sofia, Bulgaria, 26–30 April 2015; Volume 9056, pp. 617–640.
10. Chillotti, I.; Gamma, N.; Georgieva, M. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 s. In Proceedings of the Advances in Cryptology—ASIACRYPT, Hanoi, Vietnam, 4–8 December 2016; Volume 10031, pp. 3–33.
11. Chen, H.; Laine, K.; Player, R. Simple encrypted arithmetic library—SEAL v2. 1. Financ. Cryptogr. Data Secur. 2017, 10323, 3–18.
12. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
13. Chillotti, I.; Gamma, N.; Georgieva, M. TFHE: Fast fully homomorphic encryption over the torus. J. Cryptol. 2020, 33, 34–91. [CrossRef]
14. Regev, O. On lattices, learning with errors, random linear codes, and cryptography. In Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, 22–24 May 2005; pp. 84–93.
15. Lyubashevsky, V.; Peikert, C.; Regev, O. On ideal lattices and learning with errors over rings. In Proceedings of the Advances in Cryptology, Monaco and Nice, France, 30 May–3 June 2010; pp. 1–23.
16. Yoo, J.S.; Song, B.K.; Yoon, J.I. Logarithm design on encrypted data with bitwise operation. In Proceedings of the International Workshop on Information Security Applications, Jeju Island, Korea, 23–25 August 2018; pp. 105–116.
17. Hong, M.Y.; Yoon, J.W. Model Selection for Data Analysis in Encrypted Domain: Application to Simple Linear Regression. In Proceedings of the International Workshop on Information Security Applications, Jeju Island, Korea, 21–24 August 2019; pp. 155–166.
18. Akaike, H. A new look at the statistical model identification. IEEE Trans. Autom. Control 1974, 19, 716–723. [CrossRef]
19. Mallows, C.L. Some comments on $C_p$. Technometrics 1973, 15, 661–675.
20. Schwarz, G. Estimating the dimension of a model. Ann. Stat. 1978, 6, 461–464. [CrossRef]
21. Breiman, L.; Friedman, J.H.; Olshen, R.; Stone, C.J. Classification and Regression Trees; CRC Press: Boca Raton, FL, USA, 1984.
22. MacKay, D.J.C. Bayesian Methods for Adaptive Models. Ph.D. Thesis, California Institute of Technology, Pasadena, CA, USA, 1992.
23. Bishop, C.M. Pattern Recognition and Machine Learning; Springer: Berlin/Heidelberg, Germany, 2006; pp. 161–181.
24. Cheon, J.H.; Kim, A. Homomorphic Encryption for Approximate Matrix Arithmetic; Report 2018/565; 2018, Cryptology ePrint Archive.
25. Chen, H.; Gilad-Bachrach, R.; Han, K.; Huang, Z.; Jalali, A.; Lauter, K.L.; Lauter, K. Logistic regression over encrypted data from fully homomorphic encryption. BMC Med. Genom. 2018, 11, 4–81. [CrossRef] [PubMed]
26. Medical Cost Personal Datasets. Available online: https://www.kaggle.com/mirichoi0218/insurance/data (accessed on 21 December 2019).
27. Cao, X.; Moore, C.; O’Neill, M.; Hanley, N.; O’Sullivan, E. High-speed fully homomorphic encryption over the integers. In Proceedings of the Financial Cryptography and Data Security, Christ Church, Barbados, 7 March 2014; pp. 169–180.
28. Dorz, Y.; Ztrak, E.; Sunar, B. Accelerating fully homomorphic encryption in hardware. IEEE Trans. Comput. 2015, 64, 1509–1521. [CrossRef]
29. Wang, W.; Hu, Y.; Chen, L.; Huang, X.; Sunar, B. Accelerating fully homomorphic encryption using gpu. In Proceedings of the 2012 IEEE Conference on High Performance Extreme Computing, Waltham, MA, USA, 10–12 September 2012; pp. 1–5.

30. Migliore, V.; Seguin, C.; Real, M.M.; Lapotre, V.; Tisser, A.; Fontaine, C.; Gogniat, G.; Tessier, R. A high-speed accelerator for homomorphic encryption using the karatsuba algorithm. *ACM Trans. Embed. Comput. Syst.* 2017, 16, 1–17. [CrossRef]