Privacy Enhanced Decision Tree Inference

Kanthi Sarpatwar
IBM Research, Yorktown Heights, NY
sarpatwa@us.ibm.com

Nalini Ratha
IBM Research, Yorktown Heights, NY
ratha@us.ibm.com

Karthik Nandakumar
IBM Singapore Lab, Singapore
nkarthik@sg.ibm.com

Karthikeyan Shanmugam
IBM Research, Yorktown Heights, NY
karthikeyan.shanmugam2@ibm.com

James T Rayfield
IBM Research, Yorktown Heights, NY
jtray@us.ibm.com

Sharath Pankanti
IBM Research, Yorktown Heights, NY
sharat@us.ibm.com

Roman Vaculin
IBM Research, Yorktown Heights, NY
vaculin@us.ibm.com

Abstract

In many areas in machine learning, decision trees play a crucial role in classification and regression. When a decision tree based classifier is hosted as a service in a critical application with the need for privacy protection of the service as well as the user data, fully homomorphic encrypted can be employed. However, a decision node in a decision tree can't be directly implemented in FHE. In this paper, we describe an end-to-end approach to support privacy-enhanced decision tree classification using IBM supported open-source library HELib. Using several options for building a decision node and employing oblivious computations coupled with an argmax function in FHE we show that a highly secure and trusted decision tree service can be enabled.

1. Introduction

Decision Trees are very popular machine learning models used in both classification and regression settings [14]. Typically, it consists of a binary tree where each node makes a decision by comparing a feature to a threshold and splitting the decision path at that node. Leaf nodes contain decisions, real values or class labels depending on whether the task is classification or regression. Several greedy algorithms (popular examples are CART and C4.5 [6, 24]) have been proposed to learn a decision tree from training data. A common feature of these algorithms is to greedily search for a leaf node to split to grow the current tree that minimizes cross entropy or other popular metrics like Gini index.

Decision trees are very effective when the number of training samples is limited since evaluating a candidate split is less expensive during training and other complex models (like neural networks) cannot be trained due to large number of parameters. One of the appealing properties of decision trees is that they are invariant to feature scaling thereby reducing any explicit need for standardization during training. There are many popular variants of this algorithm (e.g. CHAID and FACT) [22]. Combined with ensemble methods like Gradient boosted trees [8], decision trees are very important primitives for modern day machine learning. In fact, in the case of classification and regression tasks over tabular datasets, Gradient Boosted Trees are quite the standard choice [23]. In a popular survey in the data mining community before the deep learning era, popular decision tree learning algorithms like CART and C4.5 were voted as one of the top ten algorithms used in the field of data mining [29]. There is a recent surge and renewed interest in learning decision trees due to its attractive property of being interpretable [18]. There has been a lot of work recently trying to transfer information from a complex model (neural networks and boosted trees) into a single decision tree to leverage the interpretable nature of decision trees [15, 2]. When machine learning models are used to aid decisions made by human end users in important areas like healthcare [7] and judiciary [20], interpretable models that are easy to train in a small data regime like decision trees would be a

preferable choice.

In this work, we turn our attention to the issue of deploying these ubiquitously used decision trees, hosted as a service (in a cloud environment), for critical applications where user data privacy needs to be preserved. One way to achieve this is to enable decision tree inference on homomorphically encrypted input sample at test time.

1.1. Homomorphic Encryption

Homomorphic encryption (HE) refers to a cryptosystem that allows some computation on ciphertexts without decryption [1]. A cryptosystem is *homomorphic* with respect to operation $\diamond$, if there exists an operation $\odot$ such that $x_1 \diamond x_2 = \mathcal{D}\left(\mathcal{E}(x_1, pk) \odot \mathcal{E}(x_2, pk), sk\right)$, where $\mathcal{E}$ and $\mathcal{D}$ represent the encryption and decryption mechanisms, respectively, pk and sk represent the public and private/secret keys, respectively, and x_1 and x_2 are the two plaintext operands. An HE scheme is considered to be *fully* homomorphic (FHE) if it allows arbitrary computations on the ciphertext [16]. Specifically, given $c_i = \mathcal{E}(x_i, pk), i = 1, 2, \cdots, K$, an FHE scheme allows the computation of $c = g(c_1, c_2, \cdots, c_K)$ such that $\mathcal{D}(c, sk) = f(x_1, x_2, \cdots, x_K)$ for any arbitrary function f. FHE is often achieved by employing a *somewhat homomorphic* (SWHE) or *leveled* HE scheme in combination with a *bootstrapping* or *recryption* technique. The SWHE scheme is capable of supporting computations only up to a preset level of complexity determined by its parameters. This is because the ciphertexts are "noisy", the noise keeps growing with each HE computation, and once the noise grows beyond some (parameter-dependent) threshold the ciphertext can no longer be decrypted. This problem is solved using Gentry's bootstrapping technique [16], which "refreshes" the ciphertext and reduces its noise level (at the cost of relying on circular security). However, bootstrapping is a computationally expensive and time-consuming operation. Therefore, for practical feasibility, the number of bootstrapping operations should be kept at a minimum and possibly avoided.

Several FHE cryptosystems have been proposed in the recent past based on hardness of the Ring Learning With Errors (RWLE) problem. Prominent examples of such cryptosystems include the BFV [4, 12], BGV [5], and CKKS [9] schemes. All these three schemes are both additively and multiplicatively homomorphic. While the BFV and BGV schemes are efficient for vector operations over integers, the CKKS scheme is more appropriate for "approximate" (limited precision) floating-point operations. Furthermore, all the above three schemes support Single Instruction Multiple Data (SIMD) operations, by packing different plaintext values into different *slots* in the ciphertext. While this ciphertext packing enables parallelization of addition and multiplication operations, it is important to emphasize that it is not possible to randomly access values in the individual slots of the ciphertext after packing. Only limited operations such rotation of the slots within the ciphertext is possible. Consequently, the benefits of SIMD operations can be fully leveraged only if ciphertext packing is performed smartly, so as to minimize any interaction between the slots.

In this work, we employ the CKKS scheme implemented in the HELib library [17] for our HE operations. This allows us to directly use real-valued features as input to the decision tree, without worrying about appropriately transforming the inputs as integers. Moreover, one of the critical operations in a decision tree is the conditional branching based on the result of a comparison. To implement this in practice, we rely on a soft/fuzzy comparison operator, which requires floating point operations.

2. Existing/Related Work

Machine learning algorithms in general and decision tree algorithms in particular have been studied through the lens of privacy preservation for a long time. In this section, we provide an overview of some the related existing work in this area. Three types of privacy risks from inference attacks in the life cycle of a decision tree classifier have been identified in [25], namely: (1) leakage of the private training data, (2) public release of the decision tree model, (3) privacy risk involved in using the decision tree classifier for evaluating classification queries in the inferencing phase. While all of these are important from the perspective of secure decision tree classification, in our work, we are focusing on the step 3 since in our model, the trained decision tree is hosted using a secure server. In [27], the authors develop methods to securely construct random decision trees (RDT)for both horizontally and vertically partitioned data sets. Based on [13] RDTs are constructed as follows: a non-tested feature is "randomly" chosen without using any training data at each level of the tree. The tree construction stops as soon as the depth of the tree exceeds a predefined limit. After the structure of the tree is built, the training data is used to update the statistics of each node. Each node records the number of examples of different classes that are "classified" through that node. The process is similar to classifying with a decision tree. The structure of the tree is only dependent on the order that the features are randomly chosen and is completely independent from the training data. To classify an example x, the probability outputs from multiple random trees are averaged as an estimate. Based on the protocol followed in the paper for honest adversary, they compare the performance of the proposed protocols with the existing ID3-based protocols and conclude that RDTs can provide good security with very high efficiency and security. Additionally in [19], it has been shown that RDTs can be designed to support differential privacy easily.

A cloud based privacy enhance decision tree training and inferencing system has been proposed in [21]. In this paper,

an additive homographic computation scheme is used for privacy-preserving decision tree training (PPDT) and propose three methods for different privacy levels. Additionally, to fully protect the query privacy, the classification result and the decision tree model simultaneously, they propose privacy-preserving decision tree evaluation (PPDE). Both these schemes, fully exploit their functions for counting and comparison in encrypted data under the additive homography.

Based on decision tree randomization and comparison protocol based on oblivious transfer, wu et al. [28] propose a secure decision tree evaluation. It is based on additive homomorphic computation scheme and supports semi-honest and malicious adversary attacks. Extending this concept further, De Cock et al. [11] propose privacy-preserving decision tree classifiers by decomposing the problem into few secure components such as (i) secure distributed multiplication, (ii) distributed comparison, (iii) bit-decomposition of shares, (iv) distributed inner product, (v) argmax computation, and (vi) oblivious input selection. They also support secure evaluation of other machine learning classifiers such as SVM and logistic regression using these building blocks. In a foundational work, Bost et al. [3] focus on techniques to identify a set of core operations over encrypted data: comparison, argmax, and dot product. These operators can be combined to build many classification protocols. These operators are also useful in secure decision tree evaluation. In [26], the authors propose to use BGV style FHE scheme to address the secure decision tree evaluation. In BGV scheme with secure access to sign bit in arithmetic computation, the comparison function can be easily built. In a more recent article [30] use a novel method to compute securely evaluation of decision trees using Fully Homomorphic Encryption (FHE). They replace the comparison function by a sigmoid function. The sigmoid function is approximated by a Chebyshev polynomial. They show that a 24-degree Chebyshev polynomial performs quite well for their experiments.

We propose to use FHE based approach using CKKS scheme. Additionally, we exploit the SIMD operations supported in HElib to get efficient decision tree inference. The details are described in next section.

3. The Setting

In a typical *Machine Learning as a Service* (MLaaS) scenario, trained models are hosted on a *Cloud* server. As a service, the hosting Cloud server, allows users to run inference queries on the model. Our work falls in a fundamental regime of "privacy preserving inference problems", where the goal is

- the users (*Client*) must maintain the privacy of the scoring data points

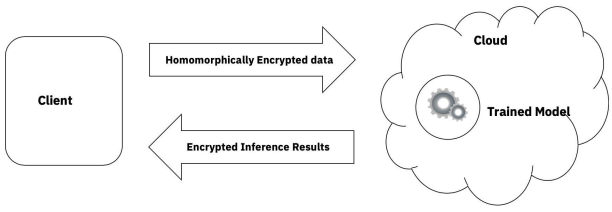

Figure 1. Privacy Preserving MLaaS Setting using Homomorphic Encryption

- The *Cloud* wishes to reveal only the model predictions in some form and keep model private to itself.

In this work, we assume that the *Cloud* owns a decision tree based model $\mathfrak{T}$ and the *Client* wishes to evaluate its data points on $\mathfrak{T}$.

4. Background: Decision Trees

Definition 1 (Decision Tree) *A Decision Tree $\mathfrak{T}$ is a binary tree structure where each node $N \in \mathfrak{T}$ has an associated triplet (f_N, t_N, v_N), where f_N is the* splitting *feature, t_N is the* threshold *and v_N is the value associated with the node.*

Definition 2 (Decision Tree Evaluation) *Given a data point $\mathbf{d}$ and a decision tree $\mathfrak{T}$, we define evaluation of $\mathbf{d}$ on $\mathfrak{T}$,* eval$(\mathbf{d}, \mathfrak{T})$ *recursively, starting from the root node, as follows:*

- *If the current node N is a leaf node, then we return* eval$(\mathbf{d}, \mathfrak{T}) := v_N$. *Otherwise,*
- *Let N_l and N_r be the left and right child of N. If $t_N > \mathbf{d}[f_N]$, we set N_l as the current node. Otherwise, we set N_r as the current node.*

Note that for regression trees the values are real values, where as, for classifier trees the values are class labels. Figure 2 shows an example of a decision tree.

5. Private Inference on Decision Trees

We use homomorphic encryption to enable privacy preserving inference on decision trees. Figure 1 depicts the high-level approach. *Client* homomorphically encrypts its data points and shares them with the *Cloud*. *Cloud* then uses the public key received from the *Client* to encrypt its model and homomorphically evaluates the decision tree on the encrypted data points.

Homomorphic encryption is excellent at evaluating polynomials of low degree. However, it is highly expensive to perform certain non-polynomial operations such as *division*, *comparison* and *argmax*. As is evident from the definition in Section 4, comparison is a basic building block of

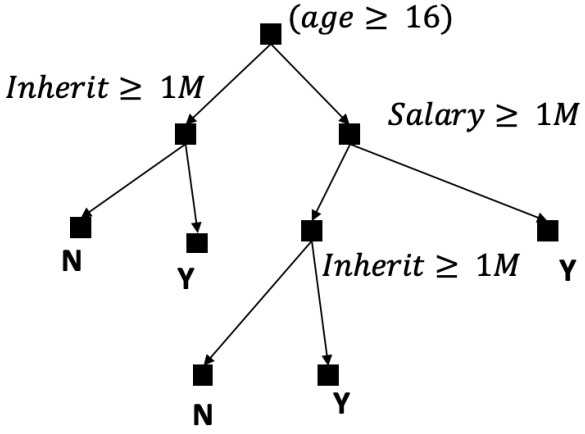

Figure 2. Decision Tree Example: Will I be a millionaire by next year?

the decision tree evaluation process. We use the approximate comparison function from [10]. The comparison function can be described as follows: Given two positive real numbers a and b, and a parameter $k \in \mathbb{N}$, we define the operation:

$$\text{EVAL}(a \geq b; k) = \frac{a^k}{a^k + b^k}$$

The intuition behind this function is the observation that if $a > b$ the above quantity tends to 1 as the parameter k increases. Since, these computations are performed in an encrypted space, their results are also obtained in an encrypted form. Hence, the tree evaluation needs to be performed in the following oblivious fashion:

1. The soft-membership (SM) of a data point $\mathbf{d}$ at any node N of the tree is computed recursively using the following equations:

   - If N is the right child of a node P, then: $SM(K, t_K, v_K) = SM(P, t_P, v_P) \times Eval(v_P \geq t_P)$

   - If N is the left child of a node P, then $SM(K, t_K, v_K) = SM(P, t_P, v_P) \times Eval(v_P < t_P)$

2. If the decision tree is a classifier over the class set $\mathcal{C}$, then for each class C:

   – Let $\mathcal{L} = \{L_1, L_2, \ldots L_k\}$ be the set of leaves in $\mathfrak{T}$ whose value is label C, then the aggregated class confidence is given by

   $$ACC(C) = \sum_{L \in \mathcal{L}} SM(L, \mathbf{d})$$

   We return the class label with maximum $ACC(.)$ value, i.e., $\mathbf{argmax}_C\, ACC(C)$

3. If the decision tree is a regressor, then assuming the leaves of the tree are $\mathfrak{L}$, we return the final value as:

   $$\sum_{L \in \mathfrak{L}} SM(L, \mathbf{d}) \cdot v_L$$

6. Implementation and Experimental Results

6.1. Implementation Details

As mentioned before, the CKKS scheme supports SIMD type operations. In plain language, this means one can pack several floating-point numbers into a single ciphertext and thereby achieve significant parallelism in operations. We exploit this SIMD capability of the CKKS scheme and evaluate all the comparisons operations involved in the decision tree evaluation using a single ciphertext based computation. A key point to note here is once packed into a ciphertext, it is not possible to retrieve individual entries in a *random-access* fashion. However, we can perform certain types of operations such as *rotation* of a ciphertext which rotates the underlying plaintext entries as well.

Ciphertext Packing: We now present our ciphertext packing strategy. Suppose the maximum depth of the decision tree is ℓ. We assume, without loss of generality, that the tree is balanced. Indeed, if the depth of a certain leaf is $\ell' < \ell$, then we can add replicas of the leaf as its left and right child, thus increasing its depth until it becomes ℓ.

Now, for a given quantity q, we denote $[q, 2^t] = [q, q, \ldots 2^t \text{ times } \ldots q]$

Ciphertext Packing at Cloud's end: Suppose $N_1 N_2, \ldots N_{2^{\ell+1}-1}$ is the order in which nodes of the decision tree are visited in a bread-first search traversal. *Cloud* constructs two ciphertext, whose packing is given by

$$T^+ = [t_{N_1}, 2^\ell][0, 2^\ell][t_{N_2}, 2^{\ell-1}][0, 2^{\ell-1}][t_{N_3}, 2^{\ell-1}][0, 2^{\ell-1}] \ldots$$

and

$$T^- = [0, 2^\ell][t_{N_1}, 2^\ell][0, 2^{\ell-1}][t_{N_2}, 2^{\ell-1}][0, 2^{\ell-1}][t_{N_3}, 2^{\ell-1}] \ldots$$

Client similarly constructs the following ciphertexts:

$$V^+ = [\mathbf{d}[f_{N_1}], 2^\ell][0, 2^\ell][\mathbf{d}[f_{N_2}], 2^{\ell-1}][0, 2^{\ell-1}] \ldots$$

and

$$V^- = [0, 2^\ell][\mathbf{d}[f_{N_1}], 2^\ell][0, 2^{\ell-1}][\mathbf{d}[f_{N_2}], 2^{\ell-1}][0, 2^{\ell-1}] \ldots$$

The homomorphic evaluation of the decision tree can then be implemented as follows:

1. Let $M^+ = T^+ + V^+$ and $M^- = T^- + V^-$.
2. Compute $\mathfrak{C} \leftarrow$EVAL$(M^+ \geq M^-)$ in an SIMD fashion.
3. For $i \in [0, \log_2 \ell]$, we rotate the ciphertext to obtain **rot**($\mathfrak{C}$, -$2^{\ell+i}$) and compute their product as follows: $\mathfrak{C} = \mathfrak{C} \times$**rot**($\mathfrak{C}$, -$2^{\ell+i}$).

It follows that the first 2^ℓ entries of the resulting ciphertext contains the SM(.) value for each leaf node.

6.2. Preliminary Experiments:

We have implemented the above ciphertext packing based approach in C++ using IBM's homomorphic encryption library (HElib). HElib supports a basic version of CKKS encryption scheme but provides no bootstrapping support. Since the comparison operation we use needs bootstrapping, we *simulate* it by a simple decryption/re-encryption routine. For the experimental setup, we used the standard datasets from the scikit-learn machine learning library. We train a decision tree regression model on the standard Boston housing dataset. We obtain the prediction score on a test dataset using our private decision tree approach and compare it with the non-private counterpart. The results show that the R^2-score of these two predictions is 0.98, which implies that the private approach was able to match the non-private version almost perfectly. As a future work, we aim to conduct experiments on a wider range of datasets.

We used the iris dataset (150 samples, 4 features and 3 classes) to demonstrate the classification use case for decision trees. In the first stage, since the comparator and other related processing require the feature values to be in a specified range, we normalized the data between 0.5-1.5. The normalization involves min-max normalization on each feature. To handle outliers in data, on each feature, 5-95 percentile is scaled to 0.5-1.5. Data 0-5 percentile is set to 0 and >95% is set to 1. We built the decision tree using 90% of the data leaving 10% of the data for validation. This training is done in cleartext using regular scikit-learn in python. The inference testing is done using the 10% of the samples. The FHE simulated comparator has been used currently along with an argmax function. The tree has 12 decision nodes. The class confidence from each node is summed up and the argmax of the class confidence is returned. We expect the results to be similar when HElib is used. We found no difference in class labels in the confusion matrix between cleartext inference and (simulated) FHE encrypted version.

7. Conclusion

In this paper, we showed an efficient and secure decision tree inferencing for regression and classification. The security is derived from the underlying provable security associated with FHE scheme. The implementation used IBM open source version of the HElib. Our future plan includes using larger datasets to demonstrate the efficacy of our approach.

References

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys*, 51(4), September 2018.

[2] Osbert Bastani, Carolyn Kim, and Hamsa Bastani. Interpreting blackbox models via model extraction. *arXiv preprint arXiv:1705.08504*, 2017.

[3] Raphael Bost, Raluca Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. 01 2015.

[4] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Annual Cryptology Conference*, pages 868–886. Springer, 2012.

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[7] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[9] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[10] Jung Hee Cheon, Dongwoo Kim, Duhyeong Kim, Hun Hee Lee, and Keewoo Lee. Numerical method for comparison on homomorphically encrypted numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 415–445. Springer, 2019.

[11] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, W. Poon, and S. Truex. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Transactions on Dependable and Secure Computing*, 16(2):217–230, 2019.

[12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

[13] W. Fan, H. Wang, P. S. Yu, and S. Ma. Is random model better? on its accuracy and efficiency. In *Third IEEE International Conference on Data Mining*, pages 51–58, 2003.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[15] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[16] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[17] Shai Halevi and Victor Shoup. HElib - An Implementation of homomorphic encryption. `https://homenc.github.io/HElib/`, Accessed April 2020.

[18] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, pages 7265–7273, 2019.

[19] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*, pages 114–121, 2009.

[20] Jon Kleinberg, Himabindu Lakkaraju, Jure Leskovec, Jens Ludwig, and Sendhil Mullainathan. Human decisions and machine predictions. *The quarterly journal of economics*, 133(1):237–293, 2018.

[21] L. Liu, R. Chen, X. Liu, J. Su, and L. Qiao. Towards practical privacy-preserving decision tree training and evaluation in the cloud. *IEEE Transactions on Information Forensics and Security*, 15:2914–2929, 2020.

[22] Wei-Yin Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, pages 1710–1737, 2009.

[23] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

[24] J Ross Quinlan et al. Bagging, boosting, and c4. 5. In *AAAI/IAAI, Vol. 1*, pages 725–730, 1996.

[25] Stacey Truex, Ling Liu, Mehmet Gursoy, and Lei Yu. Privacy-preserving inductive learning with decision trees. pages 57–64, 06 2017.

[26] Anselme Tueno, Yordan Boev, and Florian Kerschbaum. Non-interactive private decision tree evaluation, 2019.

[27] J. Vaidya, B. Shafiq, W. Fan, D. Mehmood, and D. Lorenzi. A random decision tree framework for privacy-preserving data mining. *IEEE Transactions on Dependable and Secure Computing*, 11(5):399–411, 2014.

[28] David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *IACR Cryptology ePrint Archive*, 2015:386, 2015.

[29] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.

[30] Xiaodong Xiao, Ting Wu, Yuanfang Chen, and Xing yue Fan. Privacy-preserved approximate classification based on homomorphic encryption. *Journal of Mathematical and Computer Applications*, (4:92), 2019.