Permissioned Blockchain-Based XGBoost for Multi Banks Fraud Detection

Asrori, Savitri Septiviana

Wang, Lihua

Ozawa, Seiichi

(Citation)
Neural Information Processing:683-692

(Issue Date)
2023-04-13

(Resource Type)
conference paper

(Version)
Accepted Manuscript

(URL)
https://hdl.handle.net/20.500.14094/0100489646

Permissioned Blockchain-based XGBoost for Multi Banks Fraud Detection*

Septiviana Savitri Asrori[1], Lihua Wang[2], and Seiichi Ozawa[1]

[1] Graduate School of Engineering, Kobe University, Kobe, Japan
213t265t@stu.kobe-u.ac.jp, ozawasei@kobe-u.ac.jp
[2] National Institute of Information and Communications Technology, Japan
lh-wang@nict.go.jp

Abstract. Fraud detection is one of the financial institution problems which can utilize Machine Learning (ML). However, the fraud activity is hard to detect since the occurrence is relatively low compared to the actual transaction. Several banks can collaborate to gather more fraudulent transactions from their data. However, the collaboration can cause data leakage from each bank, where the customer data should be confidential. Decentralized ML is one of the approaches to tackle the privacy-preserving aspect. This work proposed a fully decentralized environment using a permissioned blockchain to detect multiple banks' fraud. The training process utilizes a continual eXtreme Gradient Boosting (XGBoost) model. We provided the architecture of blockchain implementation for multiple banks, where it is conducted as batch and streaming data processing. As we compared our approach with the centralized, individual, and federated GBDT models, it maintains a good prediction performance and fulfills the environment of a fully distributed system.

Keywords: Blockchain · XGBoost · Decentralized model.

1 Introduction

Fraud activity usually happens among the bank transaction data. However, fraud detection usually comes with imbalanced data, where the proportion of fraud data is low compared to non-fraud data. This characteristic makes it difficult to find the pattern due to the lack of data. Therefore, some banks collaborate for ML training to gather more data and generate an aggregated models which can give better prediction results. In [1], the Secure Aggregation method is implemented for practical Privacy-Preserving Machine Learning (PPML).

The ML training process can generally be divided into centralized and decentralized. Centralized learning will collect raw data from the data owner and perform the training on a central server. In comparison, decentralized training achieves the data owner's local training process and learns a shared model from

* This work was supported in part by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP20K11826, and in part by Japan Science and Technology Agency (JST) AIP Accelerated Program under Grant JPMJCR22U5.

a locally trained model. Decentralized training has various approaches, such as Federated Learning (FL) [4] and Swarm Learning (SL) [8].

FL [4] uses a local server for training and a central server for model aggregation. In [10], FL setting is implemented using the XGBoost algorithm. The efficient FL for Gradient-Based Decision Tree (GBDT) is introduced in [9]. However, FL needs the use of the central server to update the aggregated models. On the other hand, SL provides a fully decentralized model, where raw data and aggregated models are generated at the edge using the blockchain system. For the current implementation, SL only supports parametric ML algorithms, such as Neural Networks (NN).

Extending the use of blockchain in SL, our proposed method also utilized blockchain-based machine learning. Blockchain aims to achieve and maintain integrity in distributed systems [3]. That allows the blockchain to keep its privacy-preserving and put all the peers into equal positions. Thus, the utilization of blockchain in federated learning has its opportunities and challenges, as described in [6]. There are two types of blockchain: permissionless (also called public) and permissioned (also called private). Permissioned blockchains limit the access to the authorized nodes, and each node already trusts each other [11]. In [5], permissioned blockchain is also used for financial institution data by generating a Quorum blockchain network.

This paper's main contribution is to develop a fully decentralized XGBoost model in a permissioned blockchain network. Each data owner will train the local model and update the global model stored in the blockchain. The global model consists of tree representation, hyperparameter, and configuration for the XGBoost model. Since it's written in the blockchain network, each data owner can have an equal right to access the global model, and we can eliminate the use of the central server. To the best of our knowledge, this is the first implementation of XGBoost on a permissioned blockchain network that only utilized the shared model without involving the raw data in the blockchain network.

The organization of this paper explains as follows. Section 2 covers the problem statement. Then, section 3 describes the related works on XGBoost, Blockchain-based Machine Learning, and the FL GBDT application. Section 4 represents the architecture of permissioned blockchain used for multiple banks. It also explains the overall proposed method. Experiments and Conclusions are described Section 5 and Section 6 respectively.

2 Problem Statement

In this section, we adopt the concern from real-life applications. We focus on fraud detection for transaction data of bank customers with several problems and limitations during the training process. First, the number of fraud data is small in the overall transaction data, which leads to an imbalanced data problem. Due to the increasing number of fraud data, several banks can collaborate and generate aggregated models, which expects to give a better model. However, it leads to the second problem: each data owner must secure their dataset and

keep it confidential. Although each bank already trusts the other, sharing the raw dataset among banks is prohibited since it violates the customer's privacy. Thus, we need to provide a system where data owners can still conduct their training process locally. Still, the collaborated banks can share the resulting aggregated models among the data owners to increase the prediction performance.

3 Related Works

3.1 XGBoost

XGBoost [2] is an implementation of the Gradient Boosted Decision Tree (GBDT) algorithm that is claimed as an efficient and scalable method. It is based on the function approximation of a loss function and utilizes regularization. For a dataset with n samples and m features $D = \{(\mathbf{x}_i, y_i)\}(|D| = n, \mathbf{x}_i \in \mathbb{R}^m), y_i \in \mathbb{R}$ with $\hat{y_i}^{(t-1)}$ as the prediction calculated from previous trees, suppose that the t-th decision tree f_t is constructed to minimize the objective function $\mathcal{L}^t$ as

$$\mathcal{L}^t = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega f_t. \tag{1}$$

The objective function consists of the training loss $\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i))$, which measures how well the model fits into the training data, and regularization Ωf_t, which determines the complexity of the trees. In this work, we utilized XGBoost[3] due to the ability to control over-fitting and capability for continual learning. XGBoost allows continual learning by providing the current model as the parameter for the following training process.

Table 1. Information of Datasets

Notation	Definition
n	Number of DataOwners
BN	Blockchain Network
BN.$transact(addr, data)$	Add transaction of *data* in BN from *addr*
D	Overall Training Set
$\{D^{(1)}, \ldots, D^{(n)}\}$	Training Set for $\{DataOwner_{1,\ldots,n}\}$
$\{D^{j}_{chunk_i}\}$	*j-th* Data Chunk from $DataOwner_i$
$\{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_k^{(i)}, y_k^{(i)})\}$	Data training on each $DataOwner_i$
M^{i}_{global}	*i-th* Global Model
M^{i}_{local}	*i-th* Local Model from $DataOwner_i$
$M^{j}_{local_i}$	*j-th* Local Model from $DataOwner_i$
$\|Dataset\|$	Number of element in $Dataset$

[3] https://github.com/dmlc/xgboost

3.2 Federated Learning GBDT

Maintaining the privacy between data owners is also done in FL-XGBoost [10] and eFL-Boost[9], it has S as the central server and n number of $DataOwner$. The $DataOwners$ are denoted as $U = \{u_1, u_2, u_3, \ldots, u_n\}$. $u_{tr} \in U$ has a dataset that refers to the $DataOwner$ that consecutively trains the model. This work introduces FL-XGBoost-G (FL-XGBoost with the absolute average gradient of the loss function applied). Here the selection of u_{tr} depends on the absolute average gradient of loss function g.

As the improvement of FL-XGBoost, eFL-Boost allows the federated GBDT, which minimizes accuracy loss, communication costs, and information leakage. In FL GBDT approaches, the central server is still utilized to control the aggregation mechanism among data owners. Later in Section 5, since we have a similar setting with these approaches, we compare our proposed method with FL-XGBoost-G and eFL-Boost.

3.3 Evaluation Metric: F1 Score

F1 score is an evaluation metric used to measure the performance of a model, especially for the binary classification model. It's a harmonic mean from a combination of precision and recall. Let True Positive (TP) denote the actual and predicted value is True, False Positive (FP) denote the actual value is False, but the predicted value is True, False Negative (FN) denote the actual value is True, but the predicted value is False. Given the equation as

$$Precision = \frac{TP}{(TP + FP)}, \tag{2}$$

$$Recall = \frac{TP}{(TP + FN)}, \tag{3}$$

$$F1score = 2 \times \frac{Precision \times Recall}{(Precision + Recall)}. \tag{4}$$

Equation (4) is used for our proposed method, and this evaluation will explain further in the next section.

4 Proposed Method

4.1 Development Architecture

We need to develop the blockchain network and all related technology to enable the blockchain system. Fig. 1a explains the details of blockchain architecture. We use the Ethereum[4], an open-source blockchain that enables both permissionless

[4] https://ethereum.org/en/

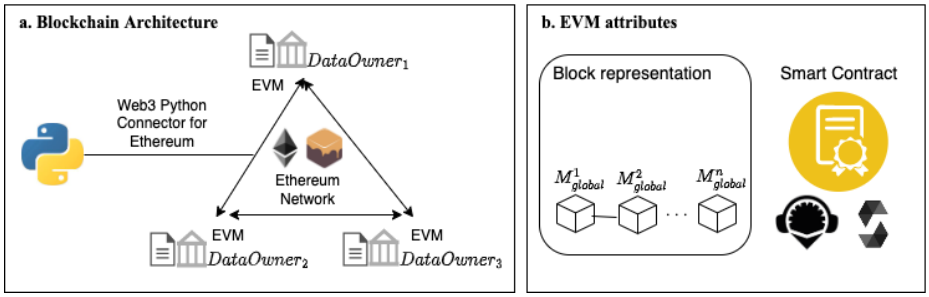

Fig. 1. Development Architecture for the Blockchain System

and permissioned blockchain for developers. For the experiment, the permissioned blockchain is generated using software called Ganache[5], which allows the quickstart of a personal Ethereum blockchain.

In the Ethereum network, the network member is called an Ethereum Virtual Machine (EVM). In our case, EVM represents each *DataOwner* who participates in the collaborative training process. Each *DataOwner* has its dataset in its local machine. To communicate with each EVM in the Ethereum network, we can utilize Web3[6], a Python3 [7] library connector to the Ethereum network. This architecture allows us to connect with the Ethereum network and do some operations inside the blockchain, such as creating a transaction and deploying the smart contract.

A smart contract is a fundamental rule that governs all activities inside the blockchain network. Solidity[7] is a high-level language for implementing smart contracts. Solidity can be compiled on Remix[8]. As illustrated in Fig. 1b, each EVM has a copy of the block representation and smart contract. When any EVM tries to make a new transaction or update the smart contract, the changes are recorded among other EVMs connected to the blockchain system. This paradigm of blockchain allows global model sharing in a fully distributed system.

4.2 Blockchain-based XGBoost Training

To achieve both the privacy aspect and the sufficient training data, we proposed the permissioned blockchain-based application on the continual XGBoost algorithm. Table 1 explains all the notations used in the following proposed method. As we already defined the problem statement, our proposed method provides two different treatments depending on the arrival time of the data. We divided the scenario into *batch processing* and *stream processing*. The batch data is collected as bulk data $D^{(i)}$ for $DataOwner_i$ in a specific time frame. This approach can

[5] https://trufflesuite.com/ganache/
[6] https://web3py.readthedocs.io/en/stable/
[7] https://docs.soliditylang.org/en/v0.8.11/
[8] https://remix.ethereum.org/

Algorithm 1 Batch Data Training

Input: Training set D from n $DataOwners$
Output: M^n_{global} aggregated models
1: BN for n $DataOwners$
2: Training set $D = \{D^{(1)}, \ldots, D^{(n)}\}$
3: where $D^{(i)} = \{(\mathbf{x}^{(i)}_1, y^{(i)}_1), \ldots, (\mathbf{x}^{(i)}_k, y^{(i)}_k)\}$
4: $M^0_{global} = 0$
5: **for** $i = 1, 2, \ldots, n$ **do**
6: $M^i_{local} \leftarrow \mathsf{train}(D^{(i)})$ from $DataOwner_i$ as continuation of M^{i-1}_{global}
7: $M^i_{global} \leftarrow M^i_{local}$
8: BN.$transact(DataOwner_i, M^i_{global})$
9: **end for**

simplify the training process since we have all the data at the beginning of the training process. While the stream data $D^{(i)}$ consists of a small chunk of data $D^j_{chunk_i}$ that arrives from time to time.

Since we proposed a collaborative training process, note that M_{local} refers to the local model trained at the edge, and M_{global} refers to the global model shared among data owners in the blockchain network. The information shared in a blockchain network is M_{global} and evaluation properties. There is no raw data sharing among the $DataOwners$.

Batch Data Training As described in Algorithm 1, all the $DataOwners$ join the BN. Then, for the first training process, the M^1_{global} refers to the M^1_{local} and $DataOwner_1$ transacts M^1_{global} to BN. Next, the M^2_{global} is the continuation from the M^1_{global} and M^2_{local}, make transaction to BN and so on. Using this method, each data owner has an equal position to affect the global model.

Stream Data Training Describe in Algorithm 2, after joining the BN, sequentially $DataOwner_i$ transact the $M^j_{local_i}$ into the BN. The shared $M^j_{local_i}$ is saved locally by each $DataOwner_k$ to calculate prediction result for $M^j_{local_i}$ using $D^j_{chunk-test_k}$. The $EvalSet_k$ consists of $F1score$ as calculated in Equation (4) and $|D^j_{chunk-test_k}|$.

Furthermore, we define Equation (5):

$$SharedF1(M) = \frac{\sum_{k=1}^{n} w_k F1score_k(M)}{\sum_{k=1}^{n} |D^j_{chunk-test_k}|}, \tag{5}$$

where w_k refers to the proportion of the chunk data in overall data, i.e. $|D^j_{chunk-test_k}| / \sum_{k=1}^{n} |D^j_{chunk-test_k}|$. The equation gives the weighted F1 score from $M^j_{local_i}$ called as $SharedF1(M^j_{local_i})$. This aggregation calculation is due to the model evaluation's fairness yet keeps the privacy of the raw data from each data owner. Once $SharedF1$ is calculated from all $M^j_{local_i}$, the $DataOwner$ with the highest $SharedF1$ is selected as $DataOwner_{best}$. The selection of $SharedF1$

Algorithm 2 Stream Data Training for the j-th chunk

Input: Training set D from n $DataOwners$
Output: $M_{global}^{j\_last}$ aggregated models

BN for n $DataOwners$
Training set $D = \{D^{(1)}, \ldots, D^{(n)}\}$
where $D^{(i)} = \{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_k^{(i)}, y_k^{(i)})\}$, and k for each $DataOwner_i$ may differ
$M_{global}^{0} = 0$
while ∞ **do**
    $D_{chunk_i}^{j} = \{(\mathbf{x}_1^{(i)}, y_1^{(i)}), \ldots, (\mathbf{x}_l^{(i)}, y_l^{(i)})\}$ split into $D_{chunk-train_i}^{j}$, $D_{chunk-test_i}^{j}$
    for $i = 1, 2, \ldots, n$ **do**
        $M_{local_i}^{j} \leftarrow \mathsf{train}(D_{chunk-train_i}^{j})$ from $DataOwner_i$ as continuation of M_{global}^{j-1}
        BN.$transact(DataOwner_i, M_{local_i}^{j})$
        for $k = 1, 2, \ldots, n$ **do**
            $EvalSet_k \leftarrow \mathsf{test}(M_{local_i}^{j}, D_{chunk-test_k}^{j})$
            BN.$transact(DataOwner_k, EvalSet_k)$
        end for
        $SharedF1(M_{local_i}^{j}) \leftarrow$ Calculate Equation (5)
    end for
    $DataOwner_{best} \leftarrow \underset{i \in \{1,\ldots,n\}}{\arg\max} (SharedF1[M_{local_i}^{j}])$
    $M_{global}^{j} \leftarrow \mathsf{train}(D_{chunk_{best}}^{j})$ from $DataOwner_{best}$ as continuation of M_{global}^{j-1}
    BN.$transact(DataOwner_{best}, M_{global}^{j})$
end while

in streaming training will eliminate the training set that does not increase the aggregated models performance. The $DataOwner_{best}$ then makes the transaction in BN by updating the M_{global}^{j}. Although it can be an infinite process in a real-life setting, in this experiment, we assume the final global model as $M_{global}^{j\_last}$, where j refers to the number of iterations until we proceed with all the training sets D. The calculation of $SharedF1$ is only proposed in Stream Data Training.

5 Experiment

5.1 Experiment Setting

As illustrated in Fig. 2, the experiment is divided into four mechanisms. Fig. 2a represents centralized training (**Central**), where all the datasets and training processes are centralized in a central server. Fig. 2b illustrates when each data owner performs an individual training (**Idv**) process relying on their dataset. Then, Fig. 2c is conducted in a federated setting, **FLXGB-G** [10] and **eFL-B** [9]. Finally, Fig. 2d is our proposed method where the training process is fully decentralized under a blockchain setting. We proposed **Batch** for Algorithm 1 Batch Data Training, and **Stream** for Algorithm 2 Stream Data Training.

For the experiment, we use a public dataset for fraud detection: **Credit** [9]. It consists of 284805 rows of data with 30 features. The proportion of fraud data

[9] https://www.kaggle.com/mlg-ulb/creditcardfraud

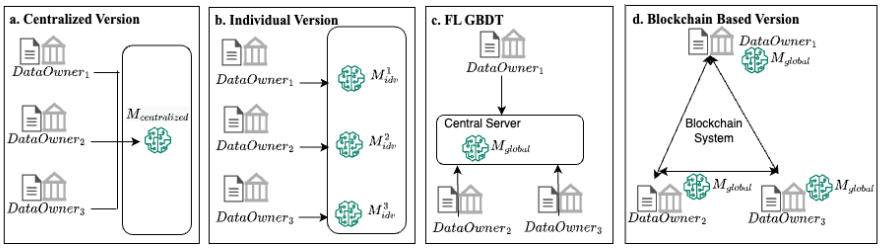

Fig. 2. Experiment setting from different perspective

Table 2. Experiment I Result

F1 score					
Central	**Idv**	**FLXGB-G [10]**	**eFL-B [9]**	**Batch**	**Stream**
0.853	0.818±0.016	0.844± 0.00	0.850 ± 0.00	**0.861 ± 0.007**	0.850

is 0.0017 (492 out of 284805). In addition, we utilized the hyperparameters of XGBoost, such as max_depth, learning_rate, subsample, colsample_bytree, colsample_bylevel, min_child_weight, gamma, reg_lambda, and n_estimators.

5.2 Experiment Result

We provide three experiments to investigate the proposed method's performance. First, Experiment I compares the result for all mechanisms described in Fig. 2. We split the dataset into a 70:30 training and testing set. Although the testing data for each implementation are the same, the **Central** mechanism trains the data all at once, while others divide the data into n *DataOwners*. For the proposed method, we assume the number of data owners $n = 3$. The experiment result is shown in Table 2.

Experiment II investigates the prediction performance when the *DataOwners* increase. Here, we provide the experiment result for $n = 3$, $n = 5$ and $n = 10$. In this experimental setting, we assume each data owner has the same number of data. Table 3 shows the Experiment II result.

In addition, Experiment III is conducted to investigate the performance when the amount of data is imbalanced among data owners. This setting represents the real-life implementation where each *DataOwner* can provide a different number of data. We follow the proportion conducted in [9], which are 1:1:1, 8:1:1, and 6:2:2. The 8:1:1 shows that one data owner has 80% of the dataset, while the other only has 10% each. Table 4 shows the Experiment III result.

5.3 Experiment Analysis

We can see from the experiment result that both **Batch** and **Stream** methods can maintain a good performance compared to other mechanisms. Furthermore, experiments I show that the proposed method can outperform the **Idv** since

Table 3. Experiment II Result

DataOwner	**FLXGB-G [10]**	**eFL-B [9]**	**Batch**	**Stream**
$n = 3$	0.799±0.00	0.805±0.00	**0.810±0.01**	0.801
$n = 5$	0.797±0.00	0.832±0.00	**0.836±0.02**	0.833
$n = 10$	0.825±0.00	**0.843±0.00**	0.842 ±0.00	0.838

Table 4. Experiment III Result

Data Proportion	**Idv**	**FLXGB-G [10]**	**eFL-B [9]**	**Batch**	**Stream**
1:1:1	0.818±0.016	0.844±0.00	0.850±0.00	**0.861±0.07**	0.850±0.00
8:1:1	0.849±0.02	0.850±0.00	**0.856±0.00**	0.854 ±0.02	0.849±0.00
6:2:2	0.830±0.00	0.843±0.00	0.848±0.00	**0.850 ±0.00**	0.844±0.00

Idv only relies on each *DataOwner* dataset. In contrast, our proposed method continually utilizes all the datasets during the training process.

Experiment II shows how adding data owners can improve prediction performance. For all algorithms compared, increasing the number of *DataOwners* can improve the prediction performance. **Batch** method shows slightly better performance than other methods. For Experiment III, the performance remains stable between proportions. Still, the blockchain-based method can maintain the prediction performance in a fully distributed manner. We need to mention that what we aim for in this proposed method is a secured system between *DataOwners*, so the prediction performance is not our primary goal.

From security analysis, the proposed blockchain method is secured from the attacker. As permissioned blockchain needs authentication of each member inside the network, it's hard for the attacker to enter the network. In the worst scenario, when any of the blockchain nodes is hacked by the attacker, the information stored in the blockchain network only consists of a shared global model (tree representation and parameter), evaluation result for shared F1 score calculation. Therefore, the attacker doesn't have access to the raw data of each data owner and is only left with the abstract tree representation.

6 Conclusion

This work introduces a fully distributed continual XGBoost approach utilizing the permissioned Ethereum blockchain network. The proposed method is designated for a secured training process among data owners, where each member is trustable and would like to generate the shared global model. However, each party still needs to secure the raw data due to privacy-preserving. To maintain data privacy, our blockchain system manages to store only the tree representation and evaluation attributes so that all the raw datasets are secured in the data owner's local machines. The proposed method provided the training process for both batch and stream depending on the characteristic of the dataset.

Our proposed method is still limited to XGBoost implementation in a blockchain setting. Investigating another algorithm, such as neural network-based or tree-

based algorithms in a blockchain network, can be considered future work. In addition, exploring another aggregation mechanism to improve the global model is also an exciting topic. While we propose the Shared F1-score, we can extend it to another evaluation method later. Then, the additional dataset for the experiment can help measure the performance of the proposed method.

References

1. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)
2. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794 (2016)
3. Drescher, D.: Blockchain Basics: A Non-Technical Introduction in 25 Steps. Apress, USA, 1st edn. (2017)
4. Konecnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
5. Mazzoni, M., Corradi, A., Di Nicola, V.: Performance evaluation of permissioned blockchains for financial applications: The consensys quorum case study. Blockchain: Research and applications **3**(1), 100026 (2022)
6. Nguyen, D.C., Ding, M., Pham, Q.V., Pathirana, P.N., Le, L.B., Seneviratne, A., Li, J., Niyato, D., Poor, H.V.: Federated learning meets blockchain in edge computing: Opportunities and challenges. IEEE Internet of Things Journal (2021)
7. Van Rossum, G., Drake Jr, F.L.: Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam (1995)
8. Warnat-Herresthal, S., Schultze, H., Shastry, K.L., Manamohan, S., Mukherjee, S., Garg, V., Sarveswara, R., Händler, K., Pickkers, P., Aziz, N.A., et al.: Swarm learning for decentralized and confidential clinical machine learning. Nature **594**(7862), 265–270 (2021)
9. Yamamoto, F., Ozawa, S., Wang, L.: efl-boost: Efficient federated learning for gradient boosting decision trees. IEEE Access **10**, 43954–43963 (2022)
10. Yamamoto, F., Wang, L., Ozawa, S.: New approaches to federated xgboost learning for privacy-preserving data analysis. In: International Conference on Neural Information Processing. pp. 558–569. Springer (2020)
11. Yang, R., Wakefield, R., Lyu, S., Jayasuriya, S., Han, F., Yi, X., Yang, X., Amarasinghe, G., Chen, S.: Public and private blockchain in construction business process and information integration. Automation in Construction **118** (2020). https://doi.org/https://doi.org/10.1016/j.autcon.2020.103276, https://www.sciencedirect.com/science/article/pii/S0926580520301886