electronics

MDPI

Article

FBLearn: Decentralized Platform for Federated Learning on Blockchain

Daniel Djolev, Milena Lazarova * and Ognyan Nakov

Faculty of Computer Systems and Technologies, Technical University of Sofia, 1000 Sofia, Bulgaria; daniel.djolev93@gmail.com (D.D.); nakov@tu-sofia.bg (O.N.)
* Correspondence: milaz@tu-sofia.bg

Abstract: In recent years, rapid technological advancements have propelled blockchain and artificial intelligence (AI) into prominent roles within the digital industry, each having unique applications. Blockchain, recognized for its secure and transparent data storage, and AI, a powerful tool for data analysis and decision making, exhibit common features that render them complementary. At the same time, machine learning has become a robust and influential technology, adopted by many companies to address non-trivial technical problems. This adoption is fueled by the vast amounts of data generated and utilized in daily operations. An intriguing intersection of blockchain and AI occurs in the realm of federated learning, a distributed approach allowing multiple parties to collaboratively train a shared model without centralizing data. This paper presents a decentralized platform FBLearn for the implementation of federated learning in blockchain, which enables us to harness the benefits of federated learning without the necessity of exchanging sensitive customer or product data, thereby fostering trustless collaboration. As the decentralized blockchain network is introduced in the distributed model training to replace the centralized server, global model aggregation approaches have to be utilized. This paper investigates several techniques for model aggregation based on the local model average and ensemble using either local or globally distributed validation data for model evaluation. The suggested aggregation approaches are experimentally evaluated based on two use cases of the FBLearn platform: credit risk scoring using a random forest classifier and credit card fraud detection using a logistic regression. The experimental results confirm that the suggested adaptive weight calculation and ensemble techniques based on the quality of local training data enhance the robustness of the global model. The performance evaluation metrics and ROC curves prove that the aggregation strategies successfully isolate the influence of the low-quality models on the final model. The proposed system's ability to outperform models created with separate datasets underscores its potential to enhance collaborative efforts and to improve the accuracy of the final global model compared to each of the local models. Integrating blockchain and federated learning presents a forward-looking approach to data collaboration while addressing privacy concerns.

Keywords: artificial intelligence; machine learning; blockchain; smart contract; federated learning; global model aggregation; credit score; credit card fraud

Citation: Djolev, D.; Lazarova, M.; Nakov, O. FBLearn: Decentralized Platform for Federated Learning on Blockchain. *Electronics* **2024**, *13*, 3672. https://doi.org/10.3390/electronics13183672

Academic Editor: Jian Sun

Received: 10 July 2024
Revised: 5 September 2024
Accepted: 12 September 2024
Published: 16 September 2024

1. Introduction

Blockchain has emerged as a transformative technology, revolutionizing data storage, privacy, and security. In response to the vulnerabilities of traditional centralized systems prone to hacks and data breaches, blockchain harnesses the power of distributed computing and cryptography to present a novel paradigm for secure, transparent, and efficient data storage and management. The inception of blockchain dates back to the early 1990s with the concept of tamper-proof ledgers capable of recording transactions in a distributed manner [1]. However, it reached a pivotal point in 2008 when Satoshi Nakamoto introduced the Bitcoin project [2], thus marking the beginning of a decentralized digital currency era, laying the foundation for blockchain's widespread adoption and the ongoing

evolution of crypto currencies. Fast forward to 2015: Ethereum took the stage, building upon Bitcoin's principles and introducing the ability to develop custom applications on blockchain using the programming language Solidity [3]. This innovation, driven by smart contracts, unlocked a myriad of possibilities beyond simple mining, enabling the creation of Turing-complete programs that could maintain specific states.

Ethereum's impact went beyond cryptocurrency development, paving the way for a new era in technology by addressing customer problems and reshaping data storage methodologies. From smart contracts to decentralized applications (dApps), blockchain's influence extends across various technological and business domains. DApps built on the decentralized foundation of blockchain offer enhanced security and transparency. Ethereum as a leading platform for dApp development provides tools for developers to handle basic to complex logics, manage digital assets, and conduct transactions. This specific application of blockchain has led to a paradigm shift in applications development known as Web 3.0 [4,5], also referred to as the "decentralized web." In contrast to the centralized nature of Web 2.0, Web 3.0 focuses on user ownership and control over data and digital identity within a secure and transparent environment. The distributed backend of blockchain ensures tamper-proof and immutable ledgers, aligning with the ethos of decentralized control. Web 3.0 revolutionizes internet dynamics by empowering users with unprecedented control over their digital identity and data, reducing reliance on centralized authorities. Smart contracts and dAapps are enabled by Web 3.0, providing a transparent, immutable, and trustless framework for executing agreements and transforming services in different domains as finance, gaming, social networking. The tokenization of assets in Web 3.0 allows for the representation of ownership or access rights on blockchain, opening new avenues for creating, transferring, and managing digital assets. Web 3.0 challenges censorship by design and leverages a decentralized architecture that resists control by any single entity, thereby promoting the freedom of information.

The realm of artificial intelligence (AI) has experienced consistent advancements and has evolved to empower machines to predict events, derive meaningful insights, and tackle intricate problems. Initially concentrated in statistical methods for data analysis, AI has progressively embraced sophisticated techniques, with neural networks [6] paving the way for the transformative era of deep learning. Machine learning (ML) models that are widely used in practice are related to prediction and classification tasks. Random forest is an ensemble learning algorithm in machine learning that operates by constructing a multitude of decision trees during training and outputting the mode (classification) or mean prediction (regression) of the individual trees. It is a versatile and powerful algorithm known for its robustness and accuracy. The core idea involves creating a "forest" of decision trees, each trained on a different subset of the data and with random feature subsets. This randomness and diversity help mitigate overfitting and enhance generalizations. It is widely used in various applications, including classification, regression, and feature selection tasks. Logistic regression is another example for a statistical method used in machine learning for binary classification problems in which the outcome variable is categorical and has two classes. It models the probability of an event occurring by fitting the data to the logistic function, also known as the sigmoid function.

ML can be advantageous in many fields. In the realm of finance [7], for example, in credit scoring, machine learning plays an important role in enhancing the quality of results. While traditional credit scores rely on basic income and debt data, modern credit models leverage machine learning techniques to incorporate additional layers of information. This includes behavioral data, financial data, transactional data, product data, and channel usage, significantly improving the accuracy of credit scoring. With credit models becoming increasingly complex and dealing with large sets of data, traditional weight-based matrices become inadequate. Machine learning-based data modelling becomes imperative in such scenarios. Credit scoring models [8] typically rely on three main data groups: customer-declared data, data from public national bureaus, and internal data held by the organization. The latter is the set that has the power to increase the accuracy of the resulted models

because modern organizations hold a vast amount of data for their customers. This practice not only reduces risk levels but also contributes to more precise credit scoring outcomes that can influence business appetites directly and positively. Another example is credit card fraud prediction, which is a very big part of fraud that is still not fully managed by the financial sector. The battle between fraudsters and anti-financial crime departments in financial institutions has not been won, because as digitalization evolves, the behavior of customers and the approaches of fraudsters change. Leveraging machine learning models, based on historical transaction data and registered fraud cases, financial organizations could enrich their transaction anti-fraud systems.

In order to create better or more accurate ML models, a single organization might struggle, but the cooperation of many organizations and their existing data assets could significantly increase the capability of their models. For example, in the financial industry, more powerful machine learning models [9] could be created based on data from each organization, better than what any of them could produce alone. Problems arise when organizations have to share their data, which, in most cases, is not an easy task, considering the privacy of the data being processed. In this case, an applicable approach is the concept of federated learning (FL), which relies on collaborative distributed training and is aimed to create a centralized model across decentralized devices or servers on decentralized data [10]. The advantage of federated learning is secured data privacy with the exchange of only training results, which also reduces network traffic. The main disadvantages of this approach are the single point of failure of the central server that maintains the global model, the lack of an incentive mechanism for participants, the lack of trust between participants in sharing training results, and the possibility of privacy attacks [11]. To overcome the disadvantages of federated learning, blockchain-based federated learning approaches have been suggested and used for various applications, such as intrusion detection for the Internet of Things [12], intelligent transportation systems, Internet of Vehicles and UAV-MEC networks [13–15], an e-health recommendation system [16], healthcare IoT systems [17,18], and a smart grid [19].

The paper presents a decentralized platform for federated learning using blockchain that is based on horizontal data distribution and allows for several global model aggregation approaches.

The main contributions of the work presented in this paper are as follows:

- The architecture of the federated learning platform using blockchain is presented. It allows for the communication of participants in FL through dApp as the front end, stores model data in decentralized storage (IPFS), and uses blockchain smart contracts to facilitate the data training interactions and final model creation;
- Global federated learning model aggregation approaches are suggested based on local model averaging or model ensembles using either local or globally distributed validation data for model evaluation;
- The presented aggregation approaches are evaluated for random forest and logistic regression classifiers;
- An experimental evaluation of two use cases of credit risk scoring using a random forest classifier and credit card fraud detection using a logistic regression is presented.

The rest of the paper is organized as follows: in Section 2, related works in the field are discussed. Section 3 presents the architecture of the FBLearn platform for federated learning on blockchain. Section 4 describes the suggested approaches for the aggregation of the global federated learning model using blockchain. Section 5 presents the experimental testing process and the results obtained for the two use cases of the FBLearn platform and the model aggregation. Section 6 contains a summary with the main conclusions as well as suggestions for future work.

2. Related Work

Federated learning, as described in Figure 1 in the classical scenario, represents a decentralized machine learning paradigm designed to train models across multiple edge

device/servers in different premises (organizations) while keeping data localized [10]. The conventional approach involves a central server orchestrating the learning process without directly accessing individual user data. Each participating device, often a smartphone or IoT device, performs local model updates based on its data and transmits only the model parameters to the central server. This collaborative learning process enables the creation of a global model that aggregates insights from diverse sources. The central server aggregates the model updates from all the devices, updating the global model iteratively. This decentralized training methodology addresses privacy concerns by lessening the need to share raw data centrally.

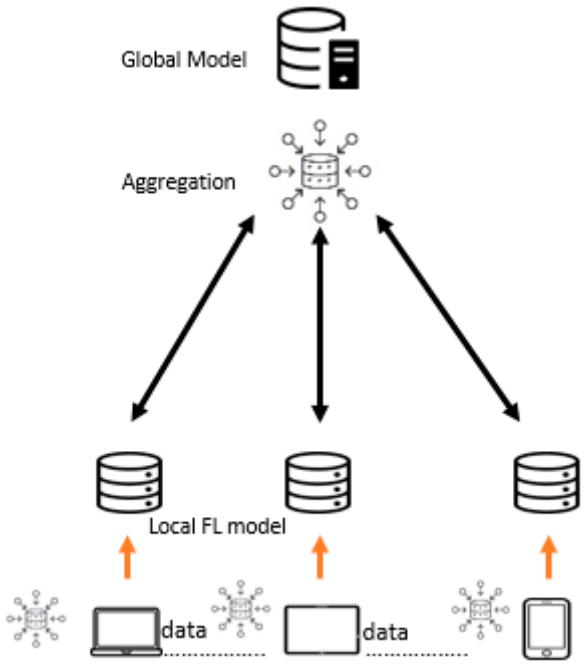


Figure 1. Classical federated learning architecture.

Federated learning is particularly advantageous in scenarios in which data privacy is a concern, as it allows for model improvement without compromising individual user information. The iterative nature of this process enhances model performance over time, fostering a collaborative learning environment without compromising the security and privacy of the contributing devices. One of the disadvantages of conventional federated learning comes from the existence of a centralized server, because it is managed by some party, which could be a problem when it comes to security and trust.

In order to use the federated learning paradigm in a way that data continue to be restricted to use for training only on the premises of each organization and, at the same time, not to have any centralization servers or dominant control by any of the participants, then further decentralization needs to be implemented in the model training. Thus, blockchain-based federated learning (BFL) was recently suggested to overcome the challenges of the general federated learning concept. As summarized in [20,21], blockchain-based federated learning either replaces the centralized server with a decentralized blockchain network or can only be used for data sharing, trustworthy storage, and the verification of client data. In the first scenario, the blockchain serves as an orchestrator of the model updates, eliminating the need for a central server. Smart contracts on the blockchain govern the aggregation of model parameters from participating organizations, ensuring transparency and integrity. Blockchain's inherent characteristics, such as immutability and transparency, enhance the security and trustworthiness of the federated learning process. Each participant's model updates are recorded on the blockchain, providing an immutable history of their contributions. This decentralized approach not only preserves data privacy but also

establishes a tamper-proof record of the collaborative model training process, fostering a secure and transparent learning environment across distributed devices. Moreover, if a token is created on a particular blockchain, it could be used to reward participants for their participation [22,23]. Using the native functions of cryptography for blockchain (the participants' public keys of their accounts) allows the participants to be kept anonymous whenever necessary.

One of the challenges of federated learning is the model aggregation approach. The federated averaging (FedAvg) algorithm based on the weighted average aggregation of the global FL model [10] is one of the most widely adopted aggregation techniques in federated learning. However, the drawback of this approach is non-IID client data as well as different types of adversarial attacks, which reflect the quality of the global FL model. To minimize the dependency of the model aggregation performed in a centralized server, thus removing the single point of failure and the data privacy issues, the global FL model in BFL platforms is usually based on smart contracts. Several studies have focused on different approaches for model aggregation in BFL. An asynchronous federated learning blockchain architecture DFL using a global consensus system to generate the proof of a participant's contribution to the ML model for model aggregation is described in [24] to achieve low levels of latency and resource consumption. The TrustDFL framework presented in [25] uses a zero-knowledge proof to establish the proof of the computation correctness of the local training process and the model aggregation process implemented by smart contracts. A blockchain-based decentralized federated learning (BCD-FL) model [26] uses a smart contract incentive mechanism based on a reverse auction and a reputation mechanism based on a model quality evaluation. The FedKD approach [27] utilizes adaptive mutual knowledge distillation and dynamic gradient compression techniques to reduce the communication cost of federated learning while preserving data privacy. In [28], the impact of jamming attacks on BFL in wireless networks is investigated. A knowledge distillation technique to defend against adversarial sample attacks is suggested in [29]. A proof-of-trust collaboration (PoTC) blockchain-based consensus protocol is presented in [19] as a decentralized trust management system in federated learning using direct and indirect trust evaluation methods and a weight difference game. A modified FedAvg algorithm with a Kullback–Leibler divergence estimation and adaptive weight calculation is used in a federated-learning-driven intrusion detection system (BFLIDS) [30] to boost model accuracy and robustness against adversarial attacks of IoMT networks. The federated learning algorithm based on update bias (FedUB) presented in [31] uses the difference between each round's local model updates and the global model updates as an update bias in the loss function of the local models to deal with partial client participation and non-IID data distributions. Bioinspired algorithms such as particle swarm optimization and genetic algorithms could be integrated with federated learning to improve the optimization of global model updates as well as local model optimization [32]. Evolutionary multi-model FL utilizing gradient-based particle swarm optimization and multiple lightweight models for handling non-IID data is presented in [33]. Based on analyses of various BFL architectures for crypto fraud detection, the authors in [34] recommend a geographically distributed cloud computing model that utilizes secure multi-party computation and lightweight consensus algorithms and protocols as the most beneficial approach to meet the scalability and performance challenges of BFL. A credit card fraud detection system presented in [35] integrates FL and blockchain techniques by maintaining a global learning model (cloud server) and local learning models in fog nodes and centralized model aggregation. The paper focuses on dataset balancing before model training and evaluates several ML and deep learning algorithms but does not use global model aggregation techniques. The authors in [36] present a modified EIFFeL architecture replacing the dedicated centralized server and the public bulletin board with a blockchain for sharing clients' credentials and intermediary results. The suggested model aggregation is based on a novel verification procedure to trace a malicious client using a shared group key for storing the encrypted shares by the clients on the blockchain platform and local aggregation recovery. A hier-

archical blockchain-enabled distributed FL system is proposed in [37] that uses FedAvg aggregation based on a distributed reputation-based verification mechanism (DRTM) and accuracy-dependent throughput management (ADTM) mechanism.

The presented research focuses on global FL aggregation approaches in a decentralized platform for federated learning on blockchain that are based on local model averaging and ensembles using either local or globally distributed validation data for model evaluation. The aim is to preserve local data privacy while reducing the risk of malicious training participants or non-IID client data.

3. FBLearn Platform

Federated learning using blockchain is not a trivial task because the two technologies (blockchain and AI) each have their own specifics. However, combining them, the resulting system could become resilient to attacks and fraud, while keeping the data of the participants undisclosed and achieving the goal of reaching a better-trained global model.

The FBLearn platform is based on the concept and the architecture of a BFL platform utilized for credit score modeling [38,39]. The platform is an Ethereum-based blockchain platform embracing Web 3.0, which orchestrates secure collaborations among organizations to contribute to the machine learning training of a global model using their local models, without any data leaving any premises. Comprising requestors (the parties that can request the training of a model) and trainers (all the parties that train local models and submit the models to the blockchain to generate the final global model), participants earn tokens for model training without exchanging data directly. Requestors initiate training via smart contracts, creating data projects and respective training plans (uploading the initial data model in IPFS cloud storage and storing only its CID in the blockchain) as transactions on the blockchain. Trainers, possessing data, contribute to an aggregated model by downloading the global modem from IPFS and training their local models, after which they upload them in IPFS and submit a transaction on the blockchain. The randomly selected aggregator finalizes the model after multiple rounds, uploading it to IPFS (cloud storage) and submitting a transaction in the blockchain again. All participants receive tokens for the respective jobs they perform in training. Requestors retrieve the model, ensuring decentralized and efficient data utilization. The system implementation consists of the following technology stack:

- Python is used for the workers' (trainers') code;
- Solidity is used for smart contracts (data plans and training plans) which run on Ethereum or another blockchain;
- The Brownie library is used to build, test, and deploy smart contracts on the blockchain;
- Genache for local blockchain instances;
- Web 3.0 for communication between workers and the blockchain;
- IPFS for web cloud storage;
- ReactJS for the web application (dApp) for the platform front end.

The system architecture of the platform and the communication process between each of the components is presented in Figure 2. The platform is based on the FELT project [40] with several modifications and new added functions that provide a federated learning network for communication between a requestor and trainers. The requestor generates a request for model training by creating a data project and a training plan in the blockchain and uploads the initial data model to distributed public storage (IPFS). The trainers use an FBLearn distributed application to join the network and to select a data plan to participate in by providing local model training. The local model training of each trainer based on private training data follows the processing steps shown in Algorithm 1. Each trainer uses its private dataset that is split into a training and test dataset (Step 1). The local model is trained using the training dataset (Step 2) and is submitted to the blockchain (Step 3).

Algorithm 1: Local Training

Input: private dataset
Output: local model

1 Each Trainer loads its local dataset using
x_train, x_test, y_train, y_test = train_test_split
2 Train local model using local training data x_train, y_train
3 Submit locally trained model on the blockchain

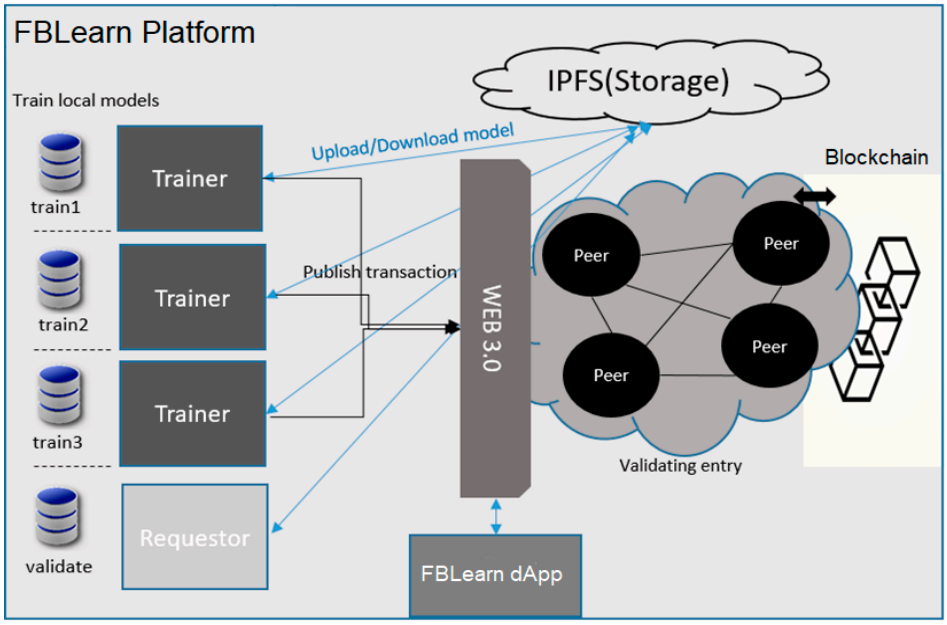


Figure 2. FBLearn platform architecture.

The federated learning orchestrator is a randomly selected trainer node that performs model aggregation following a certain aggregation approach. Several global FL model aggregation algorithms are developed in the platform as described in Section 4.

4. Global FL Model Aggregation Using Blockchain

Applying different machine learning algorithms in a federated environment is a challenge because each ML algorithm has its own specific requirements for the aggregation of the global model as a result of the local models of the participants. The aggregation algorithm has to prevent local models with lower levels of quality or accuracy from decreasing the quality of the global FL model.

The primary challenge in the research presented is aggregating local models into a global FL model. Unlike a linear regression in which averaging models' weights is straightforward, complex ML algorithms require different aggregation techniques. A simplified approach of directly averaging model parameters is either not applicable or leads to a poor global model accuracy. This aggregation technique proves to be ineffective due to the fact that the model structures of some ML algorithms are incompatible with it and the model parameters cannot be directly averaged. Based on the research and experimentation performed with the designed and developed FBLearn platform, the following two strategies for the aggregation of the local models are suggested:

- Combining models: This strategy is based on the predictions of each of the local models, which are averaged, and then the averaged prediction result is used to create a new global model;
- Ensemble models: This strategy is based on the creation of a new model that is the same type as the local models, in which the prediction functions are predefined to average the predictions of the local models that are embedded in the same model.

Based on the above strategies, several approaches for global FL model aggregation using the FBLearn platform are presented and evaluated.

4.1. Model Aggregation Using Average Predictions Based on Validation Dataset

When the local models are trained using a private dataset, the global FL model averaging is applied to combine and average the local models by using predictions on a validation dataset (Algorithm 2). In this case, each of the trainers in the FBLearn system should have a shared validation dataset. Sharing a validation dataset for the trainers before the local training is a problem because they can benefit from it by training the local models using their private data and fit them very strictly to the shared validation dataset. In order to avoid sharing a validation dataset before the training of the local models, an additional step is introduced in the general aggregation process of the global FL model. After all the trainers train their local models and write the transactions on the blockchain, the requestor initiates another transaction in order to share the validation dataset with the trainers, so each of them can generate a combined FL model using this validation dataset.

Algorithm 2: Combine and Average Local Models

Input: local models
Output: global aggregated FL model

1 Load validation data from source into validationData
2 Separate features and target variable from validationData
3 Initialize an empty list predictions
4 For each model in models:
  4.1 Predict credit score using the model on validationData
  4.2 Add the prediction to the predictions list
5 Average the predictions to obtain an averagedPrediction
6 Create a new model called FinalModel
7 Train FinalModel using the validationData and averagedPrediction
8 Return FinalModel

4.2. Model Aggregation Using Weighted Average Predictions Based on Validation Dataset

In order to avoid the additional step of sharing a validation dataset with the trainers by initializing another transaction in the blockchain by the requestor, an aggregation of the local models based on the custom weights of each of the local models is suggested. A custom property called "customWeight" is implemented which is calculated during the training of each of the local models by the trainers. The custom weight is calculated using the mean squared errors of the predictions of the local models versus the local test dataset:

$$\text{customWeight} = \frac{1}{1 + \text{MSE}}, \quad (1)$$

Thus, each calculated custom weight is a number in the range of [0, 1]. The modified algorithm for local training that involves the calculation of a custom weight value is given as Algorithm 3.

The calculated custom weight of each local model is used for the weighted average aggregation of the predictions obtained using a validation dataset. The suggested aggregation of the global FL model using the weighted average avoids the possibility of trainers with bad intentions decreasing the quality of the final model. The greater inaccuracy of the local model results in a lower weight when aggregating the global FL model. The final global model is returned after being fitted with the data of the predictions. Algorithm 4 presents the usage of the weighted average aggregation customWeight parameter.

Algorithm 3: Local Training and Calculation of Custom Weight

Input: private datasets
Output: local model with custom parameter

```
Each Trainer loads its local dataset using
    x_train, x_test, y_train, y_test = train_test_split
Train local model using local training data X_train, y_train
Calculate predictions y_pred using x_test
Calculate MSE on the model using y_pred, y_test
Calculate customWeight using MSE
Set the calculated customWeight as custom parameter of the model
Submit locally trained model on the blockchain
```

Algorithm 4: Combine and Weight Average Local Models

Input: local models
Output: global aggregated FL model

```
Load validation data from source into validationData
Separate features (x_val) and target variable (y_val) from validationData
Initialize an empty list predictions and an empty list weights
For each model in models:
    4.1 Calculate predictions using the local model on x_val
    4.2 Multiply the predictions by model.CustomWeight and add to predictions
    4.3 Add model.CustomWeight to the weights list
Calculate the weighted average prediction using the predictions and the weights
Round the weighted average prediction to obtain the final predictions
Create a new model FinalCombinedModel
Train FinalCombinedModel using the validation data (x_val) and the rounded weighted
average predictions
Return FinalCombinedModel
```

4.3. Model Aggregation Using Private Validation Dataset

In order to avoid the necessity of using shared validation data for calculating the predictions for the averaging of local models, requestor-based global model aggregation is adopted. Each trainer uses a validation dataset that is a portion of its private datasets. In this case, each trainer creates a different final model since the aggregation is based on local datasets and submits it to the blockchain. The requestor evaluates the submitted final models by each trainer using a validation dataset.

In order to fit with the blockchain rules, additional steps are introduced in the global FL process. Each of the trainers train its local models, downloads the other local models, aggregates them using its private validation dataset, and submits a final global model with a transaction on the blockchain. Algorithm 5 shows the processing steps for the local aggregation of the models using a private validation dataset.

In addition, the requestor can also share a validation dataset with a transaction in the blockchain, so that the trainers undergo a final round to choose the best global model out of the final candidate models making predictions using the validation dataset. In this case, the trainers use only private local data during the training and the requestor shares the validation data only after the training is complete.

The global model aggregation and the final model selection among the submitted final model by the trainers can also be made by the requestor without sharing any validation data. In this case, the aggregation of the global FL model is also based on the custom weights of each of the local models.

Algorithm 5: Combine and Weight Average Local Models with Private Validation Data

Input: local models, x_test
Output: global aggregated FL model

1 Set x_val to be equal to x_test
2 Initialize an empty list predictions and an empty list weights
3 For each model in models:
  3.1 Calculate predictions using the model on x_val
  3.2 Multiply by model.CustomWeight and add to predictions
  3.3 Add model.CustomWeight to the weights list
4 Calculate the weighted average prediction using the predictions and the weights
5 Round the weighted average prediction to obtain the final predictions
6 Create a new model CombinedFinalModel
7 Train CombinedFinalModel using x_val and the rounded weighted average predictions
8 Return CombinedFinalModel

4.4. Ensemble-Based Model Aggregation

In order to avoid the utilization of any dataset for making predictions at the global model aggregation stage, an ensemble-based model aggregation is applied. In order to create an ensemble of the final model, two new custom Python classes are implemented which inherit the RandomForestClassifier by overriding some of its functions and properties: FederatedEnsembleClassifier and WeightedFederatedEnsembleClassifier. The only difference between the two classes is the utilization of the weighted average of the local models based on custom weight parameters calculated using the MSE, as shown in Algorithm 3. Algorithms 6 and 7 present the processing stages of the global FL model aggregation using the FederatedEnsembleClassifier and WeightedFederatedEnsembleClassifier approaches.

The FederatedEnsembleClassifier creates a classification model and returns it as a global model that each of the trainers can write on the blockchain. The model validation can either be made by sharing a validation set by the requestor within a new transaction or by the requester only. The FederatedEnsembleClassifier holds the array of the local models, and predictions and the averaging of the local models can easily be performed on whatever dataset is needed. The advantage of this aggregation approach is that there is no data requirement for the global FL model aggregation during training; neither local nor validation data are required.

Algorithm 6: Federated Ensemble Classifier

Input: local models
Output: Ensemble global FL model

1 Initialize the FederatedEnsembleClassifier with a list of models
2 Define a method __init__ to set the initial models
3 Make predictions for each model on input data X using method predict_proba(X)
  3.1 Initialize an empty list predictions
  3.2 For each model in self.models:
    3.2.1 Make probability predictions using model.predict_proba(X)
    3.2.2 Add the predictions to the list
  3.3 Calculate the average predictions using np.average along axis=0
  3.4 Return the average predictions
4 Define a method predict(X) to predict the class labels for input data X:
  4.1 Call predict_proba(X) to get the probability predictions
  4.2 Find the index of the maximum probability along axis=1 using np.argmax
  4.3 Return the indices as the predicted class labels

Using WeightedFederatedEnsembleClassifier, the weights of the custom weight parameters are calculated during the local training of the models using the MSE calculated based on the predictions of private trainers' datasets.

Algorithm 7: Weighted Federated Ensemble Classifier

Input: local models
Output: Ensemble global FL model

1 Initialize the WeightedFederatedEnsembleClassifier with a list of models
2 Define a method __init__ to set the initial models.
3 Make predictions for each model on input data X using method predict_proba(X)
  3.1 Initialize an empty list predictions and an empty list weights
  3.2 For each model in self.models:
    3.2.1 Calculate probability predictions using model.predict_proba(X)
    3.2.2 Multiply the predictions by model.CustomWeight and add to predictions
    3.2.3 Add model.CustomWeight to the weights list
  3.3 Calculate the weighted average predictions using weights
  3.4 Return the weighted average predictions
4 Define a method predict(X) to predict the class labels for input data X:
  4.1 Call predict_proba(X) to get the probability predictions
  4.2 Find the index of the maximum probability along axis=1 using np.argmax
  4.3 Return the indices as the predicted class labels

4.5. Ensemble-Based Model Aggregation Using Confusion Matrix

Instead of using local MSEs for the calculation of the custom weight parameters of the local models, a confusion matrix can be utilized as a quality metric for the local models. The authors of [41] propose the use of Matthews correlation coefficient (MCC) [42] or the phi coefficient as a statistical measure used to discover the association between two binary variables. The MCC is calculated as follows:

$$\mathrm{MCC} = \frac{(\mathrm{TP} \times \mathrm{TN}) - (\mathrm{FP} \times \mathrm{FN})}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TP} + \mathrm{FN}) \times (\mathrm{TN} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FN})}}, \quad (2)$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negative predictions.

The MCC considers all elements within the confusion matrix, making it resilient to the challenges posed by imbalanced or skewed datasets. The MCC outperforms other commonly used performance metrics demonstrating superior levels of robustness and reliability in assessing classifier performance across both balanced and imbalanced scenarios.

MCC values are in the range of $[-1, +1]$, where +1 signifies a flawless classifier, 0 denotes a classifier making random guesses, and -1 indicates a classifier predicting all data incorrectly. Hence, the focus is solely on classifiers exhibiting a positive MCC value. In cases in which all instances in the dataset belong to a single label (either positive or negative), or if the classifier predicts all data as either positive or negative, both the numerator and denominator in the MCC become zero, rendering the calculation undefined. In the current study, the calculation of the MCC follows the proposal in [26]. The authors observed that removing weak classifiers with an MCC value below a certain threshold τ increases the performance of the final ensemble classifier. The reported experiments show that optimal results are obtained for a threshold value of $\tau = 0.2$; thus, the values below 0.2 are considered 0. Algorithm 8 presents the local training with the confusion matrix calculated as the mcc_weights parameter for each model.

In order to aggregate the global FL model, the MCC weights of the local models are calculated, the parameter customWeight is set to the MCC weight values and is stored in the local models, and the aggregation follows Algorithm 7 for WeightedFederatedEnsem-

bleClassifier. The steps for ensemble-based model aggregation using the MCC weights are given as Algorithm 9.

Algorithm 8: Local Training with Calculation of Confusion Matrix

Input: local models
Output: Locally trained model with custom parameter

1 Each Trainer loads its local dataset using
X_train, X_test, y_train, y_test = train_test_split
2 Train local model using local training data X_train, y_train
3 Calculate predictions using X_test and write them in y_pred
4 Calculate conf_matrix custom parameter of the model using y_test, y_pred
5 Submit locally trained model on the blockchain

Algorithm 9: Weighted Federated Ensemble Classifier Using Confusion Matrix

Input: local models
Output: Locally trained model with custom parameter

1 Check if all models have a 'customWeight' property:
1.1 If any model does not have 'customWeight',
raise a ValueError indicating that all models must have this property
2 Calculate MCC weights for each model:
2.1 Initialize an empty list mcc_weights
2.2 For each model in models:
2.2.1 Calculate MCC for the model's 'conf_matrixCustParam'
2.2.2 Add the MCC value to mcc_weights list
3 Normalize the MCC weights to ensure they sum to 1:
3.1 Divide each value in mcc_weights by the sum of mcc_weights
4 Set the 'customWeight' property for each model using
the corresponding normalized MCC weight
5 Create the weighted federated ensemble model:
5.1 Initialize ensemble_model as a WeightedFederatedEnsembleClassifier with models
6 Return the ensemble_model

5. Experimental Testing and Results

The experimental testing of the FBLearn platform aims to explore and evaluate the suggested approaches for global FL model aggregation. The designed experiments estimate the aggregated global FL model in comparison to the local models to prove if the aggregation approaches are beneficial for the requestor.

The results of two experimental use cases of the FBLearn platform are presented and discussed: Use Case 1 based on Credit Risk Scoring dataset [43] using the random forest classifier and Use Case 2 based on Credit Card Fraud dataset [44] using the logistic regression.

5.1. Experimental Setup

The experimental environment is based on the following setup:

- Hardware: Operating system: MS Windows 10; RAM: 16 GB; CPU: Intel$^{(R)}$ Core™ i5-7300HQ; GPU: Intel$^{(R)}$ HD Graphics 630, AMD Radeon™ RX 560;
- Software: Python–3.10.2; Brownie–v1.19.3; Solidity–0.8.0; ReactJS–React 18;
- Data sources: Use Case l: public data for Credit Risk Score model; Use Case 2: public data for Credit Card Fraud model;
- Number of parallel Local Trainers: Three trainers;
- ML algorithms: Random forest classifier; Logistic regression.

5.2. Evaluation Metrics

In order to access the data quality of the datasets, common ML evaluation metrics are used, which allow the datasets and the models to be evaluated:

- Accuracy: Accuracy is a performance metric that measures how well the model predicts or classifies data. It is the ratio of correctly predicted instances to the total instances in the dataset, expressed as a percentage. Thus, accuracy provides an overall assessment of a model's correctness by considering both true positive and true negative predictions. The accuracy of a model is calculated as follows:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{3}$$

- Precision: Precision is a performance metric that measures the accuracy of positive predictions made by a model. It is the ratio of true positive predictions to the sum of true positive and false positive predictions. Precision focuses on the precision of the model's positive predictions, indicating how well the model performs when it predicts a positive outcome. The precision is calculated as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{4}$$

- Recall: Also known as sensitivity or the true positive rate, recall is a performance metric in machine learning that measures the ability of a model to capture correctly all relevant instances of a particular class. It is the ratio of true positive predictions to the sum of true positives and false negatives. Recall provides insight into how well a model identifies all actual positive instances. The recall of a model is calculated as follows:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{5}$$

- F1 Score: F1 Score is a metric in machine learning that combines both precision and recall into a single value, providing a balanced measure of a model's performance especially in scenarios with an imbalanced class distribution. It is particularly useful when there is a need to strike a balance between false positives and false negatives. F1 Score is calculated as follows:

$$\text{F1} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}, \tag{6}$$

- Mean Squared Error (MSE): MSE is a commonly used metric to evaluate the performance of a regression model in machine learning. It quantifies the average squared difference between the predicted values and the actual values in a dataset. The MSE is calculated as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2, \tag{7}$$

  where n is the number of data points, Y_i are the observed values, and $\hat{Y}_i$ are the predicted values.

- R-Squared (R^2): R^2, also known as the coefficient of determination, is a statistical metric used to assess the goodness of fit of a regression model. R^2 measures the proportion of the variance in the dependent variable that is predictable from the independent variables in the model. Its values range from 0 to 1; a value of 1 indicates that the model perfectly predicts the dependent variable, while a value of 0 means the model does not provide any predictive information. Negative values are possible and indicate that the model performs worse than a simple mean-based model. R^2 is calculated as follows:

$$R^2 = 1 - \frac{\text{SSR}}{\text{SST}}, \tag{8}$$

where the SSR is the sum of the squared differences between the predicted and actual values (residuals), and the SST is the total sum of the squared differences between the actual values and the mean of the dependent variable.

- ROC/AUC (receiver operating characteristic curve/area under the curve): The ROC curve is a graphical representation commonly used in binary classification models to assess their performance. In the ROC curve, the true positive rate (sensitivity) is plotted against the false positive rate (1—specificity) at various threshold settings for the model. The diagonal line in the curve represents the ROC curve for a random classifier. The AUC is the measure of the area under this curve. A higher AUC value indicates the model has a better ability to distinguish between the positive and negative classes. The AUC ranges from 0 to 1, where 0.5 represents a model that performs no better than random chance, and 1 indicates a perfect model.

5.3. Experimental Results for Credit Risk Scoring Using FBLearn Platform

Use Case 1 is based on a dataset for Credit Risk Scoring [43]. The original dataset is split to training, test, and validation datasets that are preprocessed using the processing steps described in [45]. The exploratory data analyses applied are aimed at data cleaning, feature and target data transformation, encoding, and proper feature scaling. The obtained preprocessed training dataset is then split into three training files for each of the three trainers (nodes): score1.csv, score2.csv, and score3.csv. In addition, one fake training file (fake.csv), comprising randomly generated data, is generated and used in some of the test cases of the third node. A preprocessed validation dataset (validate.csv) is also used. A summary of the number of data samples and the number of data features of the datasets is given in Table 1.

Table 1. Training and validation datasets for Use Case 1.

Dataset File Name	Data Rows Count	Number of Features
score1	16,001	29
score2	16,000	29
score3	17,991	29
fake	17,990	29
validate	24,996	29

All data files comprise the same features: numeric columns for age, annual balance, number of credit cards, number of loans, outstanding balance, credit history, money spent, etc., as well as a "TARGET" column with values of {0, 1, 2}, in which a value of "0" represents a "Bad Credit Score", a value of "1" is a "Risky Credit Score", and a value of "2" denotes a "Good Credit Score". In this use case, the target variable is a numeric value from a limited set and the model is trained to predict one of the target values, thus classification ML is used for model training. For this use case of the FBLearn platform, a random forest classifier is utilized. Random forest classifiers are well suited for tasks with a large number of input features because they are capable of handling high-dimensional data without the risk of overfitting, as is the discussed use case for credit risk scoring.

The parameters of the random forest classifier used for the experimental evaluation in Use Case 1 are based on the default settings of the scikit-learn implemented using the Python class RandomForestClassifier: the number of trees in the forest is 100, the function to measure the quality of a split is based on Gini impurity, the minimum number of samples required to split an internal node is two, the minimum number of samples required for a leaf node is one, the number of features to consider for the best split is the square root of the number of features, and bootstrap samples are used when building the trees. No hyperparameter tuning is applied in the presented experimental evaluation as its focus is on distributed learning and global model aggregation. The same parameters are used in model training, testing, and validation.

Two approaches are applied in order to assess the quality of the data in all the training datasets:

- Approach 1: The same datasets are used for training and validation

Each dataset is split into training and testing data using a ratio of 80:20. A model is obtained using random forest classification based on the training data. Predictions are calculated using the test data and are compared with the target feature. The results for the selected evaluation metrics are given in Table 2. Figure 3 shows the ROC curves for the used data sources.

Table 2. Results for dataset assessment: Approach 1, Use Case 1.

File/Test	Accuracy	F1 Score	Precision	Recall	MSE	R^2
score1	0.8304	0.8202	0.8104	0.8320	0.1865	0.5767
score2	0.8481	0.8396	0.8318	0.8487	0.1847	0.5840
score3	0.8477	0.8393	0.8352	0.8437	0.1656	0.6198
fake	0.4947	0.2894	0.3036	0.3220	0.6203	−0.4591

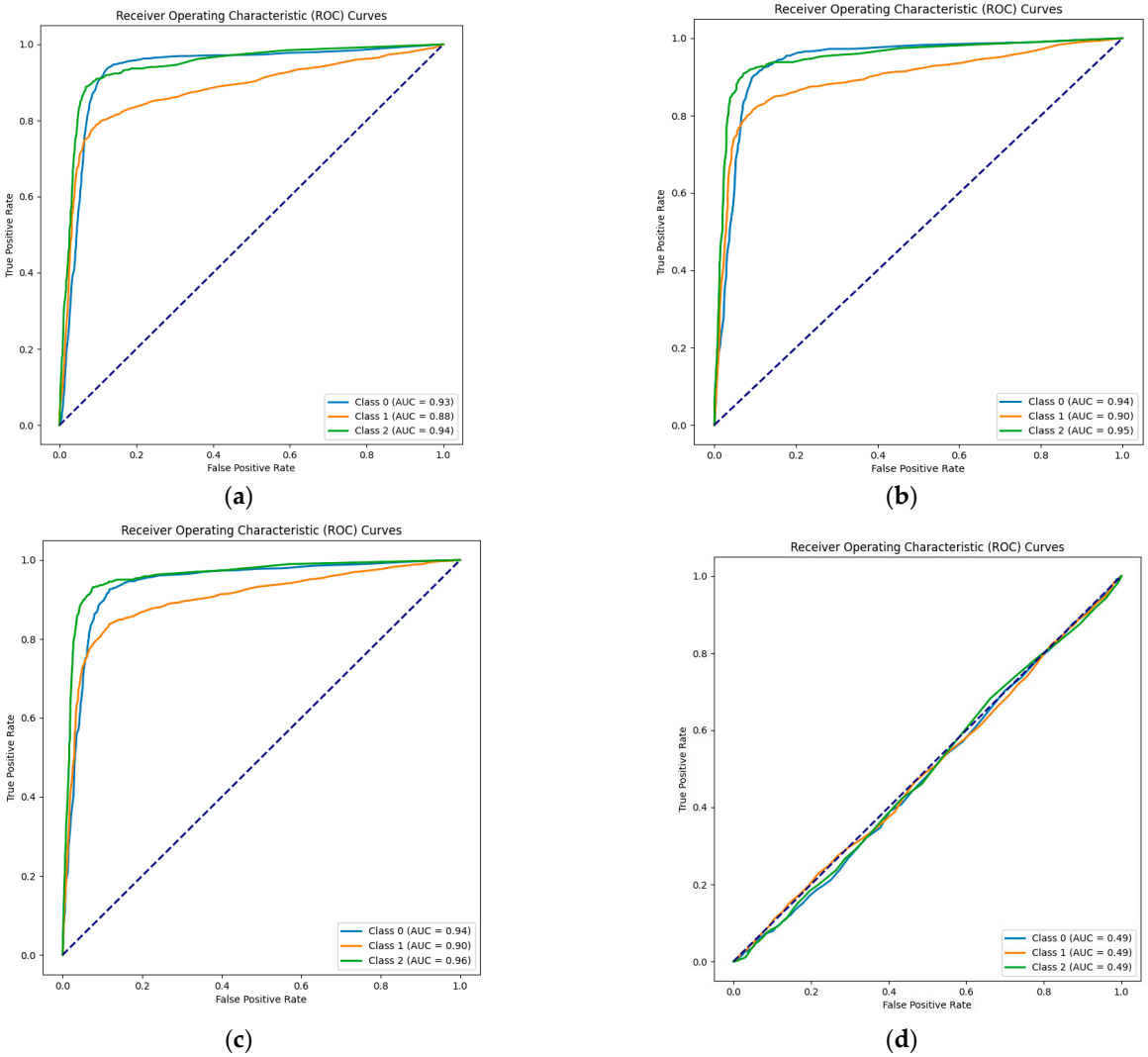


Figure 3. ROC curves for dataset assessment Approach 1, Use Case 1: Datasets: (**a**) score1; (**b**) score2; (**c**) score3; (**d**) fake.

- Approach 2: Validation using validation file

For each dataset, a model is obtained using random forest classification based on all the samples from the dataset. Predictions are calculated for the validation samples and are compared with the target feature in the validation dataset. The results for the selected evaluation metrics are given in Table 3. Figure 4 shows the ROC curves for the used data sources where Class 0 represents the "Bad Credit Score" category, Class 1 denotes the "Risky Credit Score" category, and Class 2 is the "Good Credit Score" category.

Table 3. Results for dataset assessment: Approach 2, Use Case 1.

File/Test	Accuracy	F1 Score	Precision	Recall	MSE	R^2
score1	0.6951	0.6803	0.6760	0.6853	0.3697	0.2172
score2	0.6963	0.6826	0.6778	0.6893	0.3727	0.2107
score3	0.7018	0.6854	0.6841	0.6867	0.3596	0.2386
fake	0.5016	0.2675	0.3408	0.3368	0.5486	−0.1616

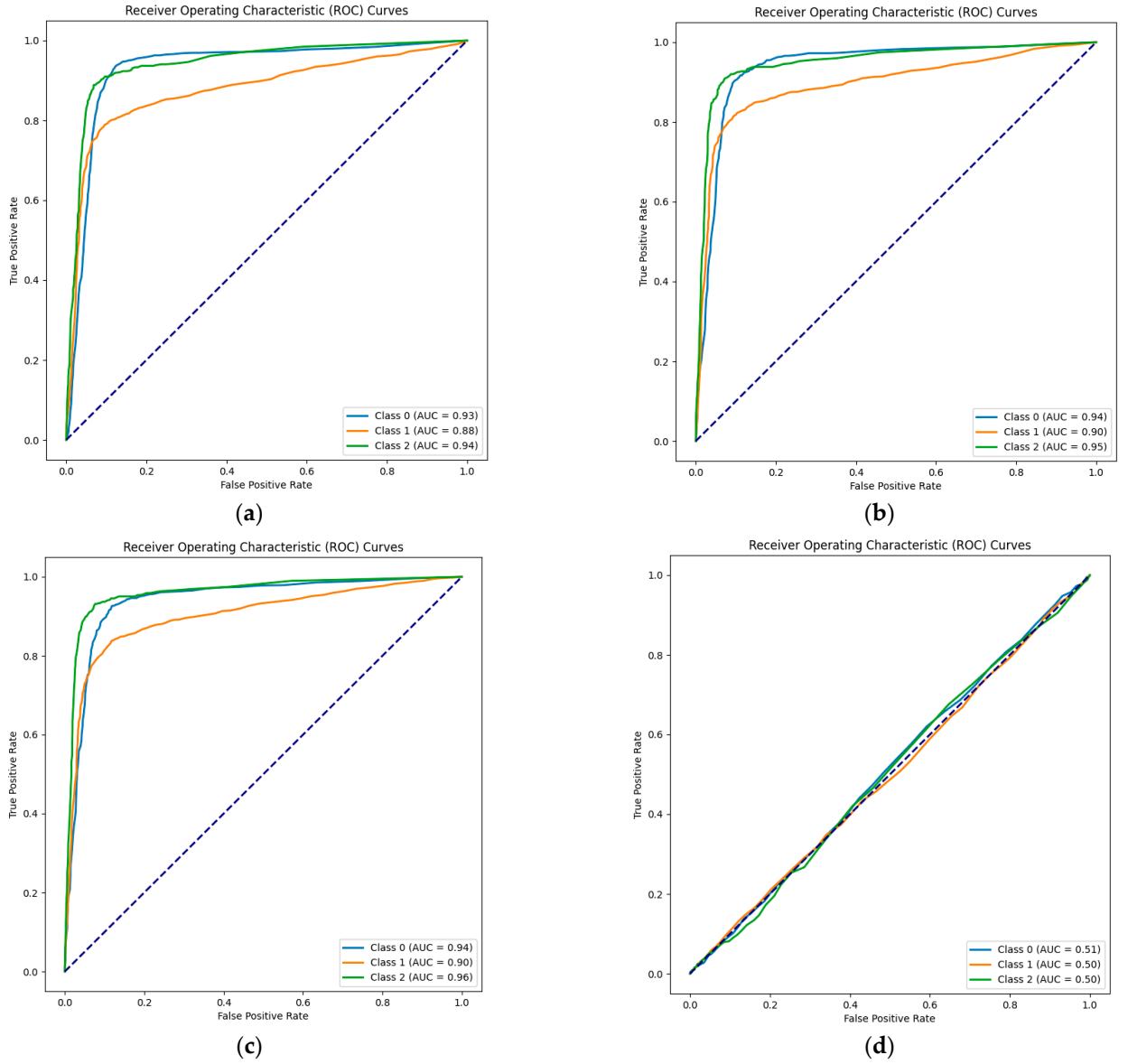


Figure 4. ROC curves for dataset assessment: Approach 2, Use Case 1. Datasets: (**a**) score1; (**b**) score2; (**c**) score3; (**d**) fake.

Both dataset assessment approaches show comparable results for the evaluation metrics. The first three data sources have almost equal evaluation metrics and ROC curves while the fourth (fake.csv), which has a different distribution, shows lower evaluation metrics. That is why the experimental results for Use Case 1 are divided in two parts, with two different hypotheses:

- Part 1: Experiments are executed using only the first three data sources (score1, score2, and score3). The hypothesis for the evaluation is that the trainers are participating in the distributed learning with high-quality datasets and are not aiming to disturb the system or to decrease the quality of the final global FL model. The goal is to evaluate the suggested approaches for global FL model aggregation;
- Part 2: Experiments are executed using the first two data sources (score1 and score2) for two trainers, and the third trainer uses the fourth data source (fake.cvs). The hypothesis is that the first two trainers participate in the distributed training with data sources of similar quality and the third one uses a lower-quality dataset for local model training, thus either it aims to decrease the quality of the final global model or it just does not have good-quality data. The goal is to evaluate the resiliency of the distributed training system against trainers with datasets of differing quality used during local training.

For part 1 of Use Case 1 (described in Table 4), the training datasets score1, score2, and score3 have a similar quality and distribution; thus, the goal is to explore different scenarios of distributed training and validation using the suggested approaches for global FL model aggregation. The test cases correspond to nine scenarios for global model aggregation according to the above presented algorithms.

Table 4. Test case scenarios for Part 1 of Use Case 1.

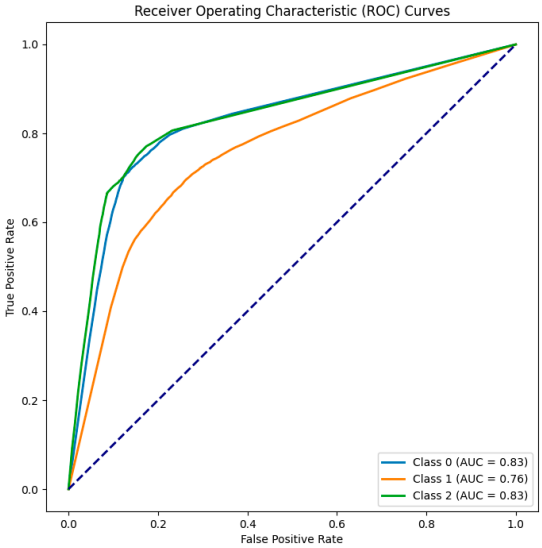
Scenario	Global Model Aggregation	Validation	Weights	Weight Type
1	Combine (Algorithms 1 and 2)	Validate dataset	No	n/a
2	Combine (Algorithms 3 and 4)	Validate dataset	Yes	MSE
3	Combine (Algorithm 5)	Local test datasets	Yes	MSE
4	Combine (Algorithm 5)	Local test datasets	Yes	MSE
5	Combine (Algorithm 5)	Local test datasets	Yes	MSE
6	Choose the best (Algorithm 5)	Each trainer	Yes	MSE
7	Ensemble (Algorithm 6)	Validate	No	n/a
8	Ensemble (Algorithms 6 and 7)	Validate	Yes	MSE
9	Ensemble (Algorithms 6, 8 and 9)	Validate	Yes	MCC

The results for the test cases are given in Table 5. The results show relatively similar values for all the evaluation metrics. The ROC curves shown in Figure 5 also confirm the quality of the global FL models obtained. Generally, the models have small MSE and R-squared parameter values and high values for accuracy, F1 score, precision, and recall. The aggregation of the global model in all the cases using different algorithms shows equal results when the data of both trainers are of similar quality. In all the cases, the achieved quality of the global FL model is good with quality metrics very similar to those of the original datasets, thus confirming that the model aggregation can be based on all of the suggested approaches. Test Cases 3, 4, and 5 show good performance and quality versus the original quality of the models trained on the separate datasets. The quality metrics for the resulting final global models of Test Cases 7, 8, and 9 are significantly good. This confirms that the aggregation techniques used in Test Cases 7, 8, and 9 are the best.

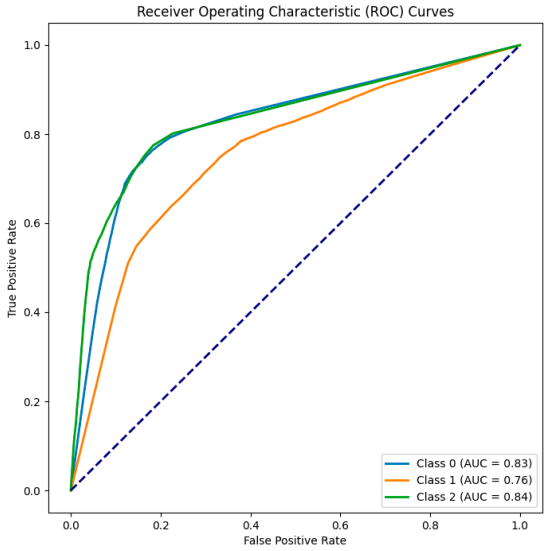
For Part 2 of Use Case 1 (described in Table 6), the same algorithms for aggregation of the global FL models are used, but the third local trainer node uses a dataset with a significantly lower quality (fake.csv).

Table 5. Experimental results for test case scenarios for Part 1 of Use Case 1.

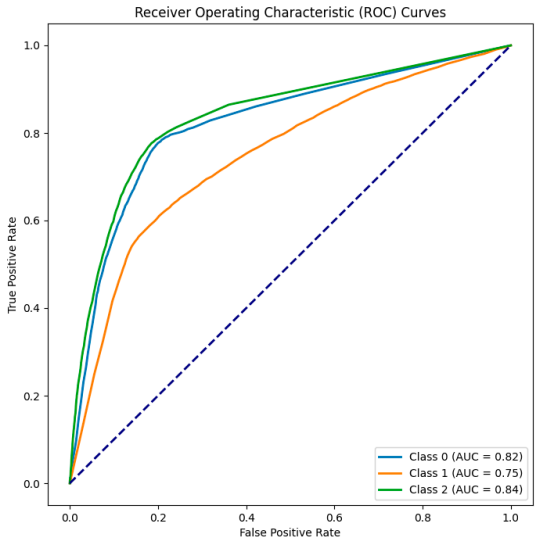
Test Case	Accuracy	F1 Score	Precision	Recall	MSE	R^2
1	0.7024	0.6921	0.6906	0.6939	0.3289	0.3035
2	0.7009	0.6736	0.7127	0.6543	0.3139	0.3354
3	0.6542	0.5959	0.6674	0.5799	0.3669	0.2231
4	0.6545	0.6000	0.6637	0.5854	0.3707	0.2151
5	0.6511	0.5897	0.6651	0.5771	0.3727	0.2109
6	0.6511	0.5897	0.6651	0.5771	0.3727	0.2109
7	0.7386	0.7256	0.7201	0.7327	0.3241	0.3137
8	0.7374	0.7247	0.7186	0.7327	0.3268	0.3080
9	0.7380	0.7254	0.7195	0.7330	0.3249	0.3121



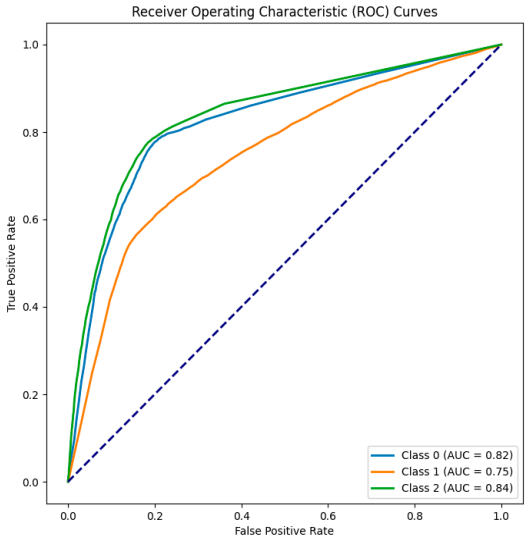
(**a**)



(**b**)



(**c**)



(**d**)

Figure 5. *Cont.*

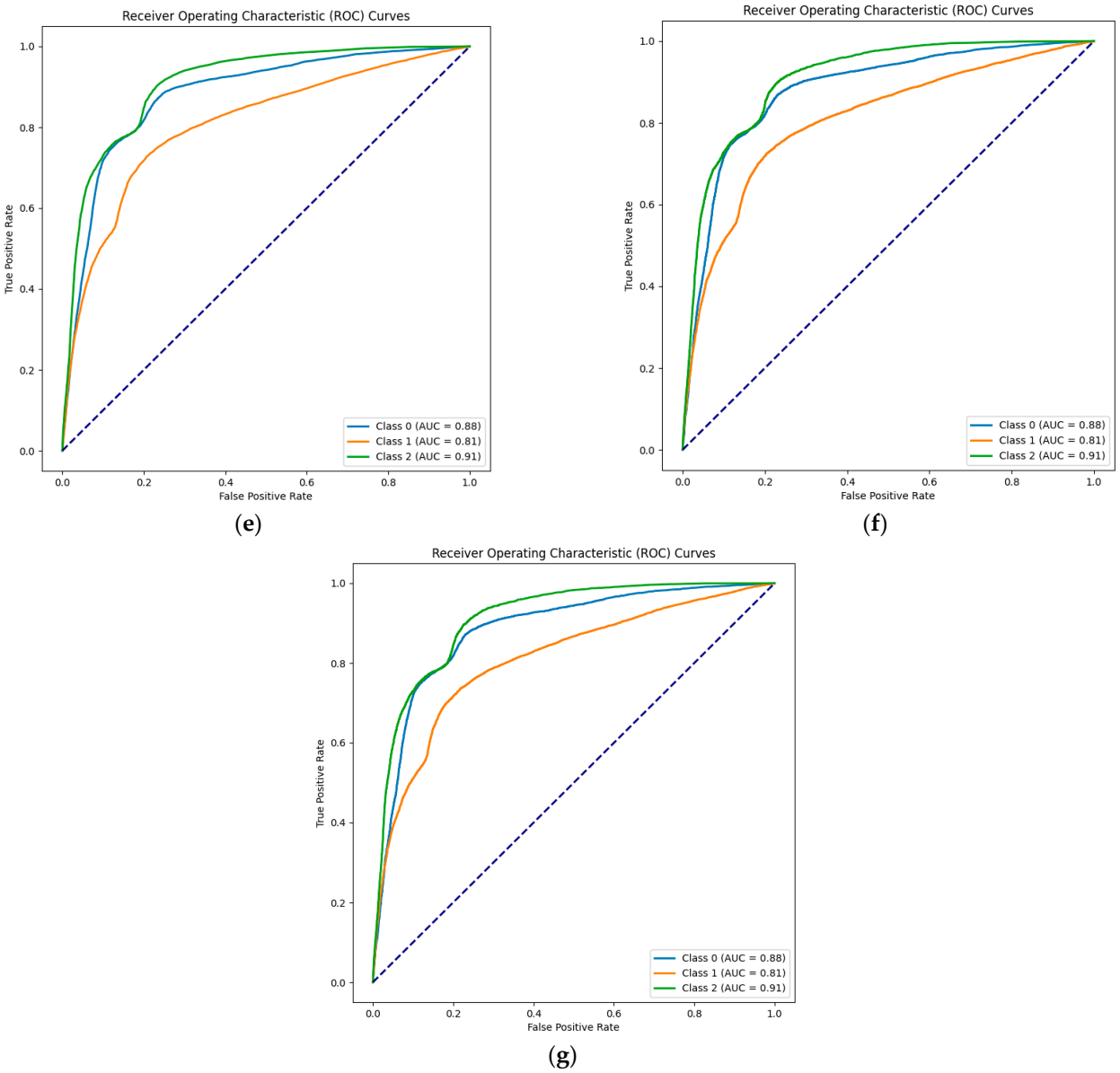


Figure 5. ROC curves for the global FL models of Use Case 1, Part 1: (**a**) Test Case 1; (**b**) Test Case 2; (**c**) Test Case 3, 4, 5; (**d**) Test Case 6; (**e**) Test Case 7; (**f**) Test Case 8; (**g**) Test Case 9.

The results for the test cases are given in Table 7 and the ROC curves are shown in Figure 6. In Test Case 10, the global FL model is aggregated using a combination of the local models and no weights are applied. The ROC curve shows significantly good results; there is no serious deviation in the quality of the global model influenced by the fake dataset as a result of the combination. The results for Test Cases 11 - 14 show that aggregation using weighed average approaches ensures the high quality of the model and the small influence of the fake dataset on the global FL model. Test Cases 12, 13, and 14 have three candidates for the final model, and the results show the models are of comparable quality, which means that the requestor can choose one of them. This can be a good approach in the specific situation in which the local trainers are required to use private datasets to create the final model proposition. Test Cases 15, 16, and 17 use the ensemble technique for global FL model aggregation: without weights (Test Case 15) and with different weighting approaches (Test Case 16 using the MSE and Test Case 17 using the MCC). The results show that these approaches ensure comparable results with the others and the quality of the global FL model remains significantly good despite the influence of the fake dataset.

Table 6. Test case scenarios for Part 2 of Use Case 1.

Scenario	Global Model Aggregation	Validation	Weights	Weight Type
10	Combine (Algorithms 1 and 2)	Validate dataset	No	n/a
11	Combine (Algorithms 3 and 4)	Validate dataset	Yes	MSE
12	Combine (Algorithm 5)	Local test datasets	Yes	MSE
13	Combine (Algorithm 5)	Local test datasets	Yes	MSE
14	Combine (Algorithm 5)	Local test datasets	Yes	MSE
15	Ensemble (Algorithm 6)	Validate	No	n/a
16	Ensemble (Algorithms 6 and 7)	Validate	Yes	MSE
17	Ensemble (Algorithms 6, 8 and 9)	Validate	Yes	MCC

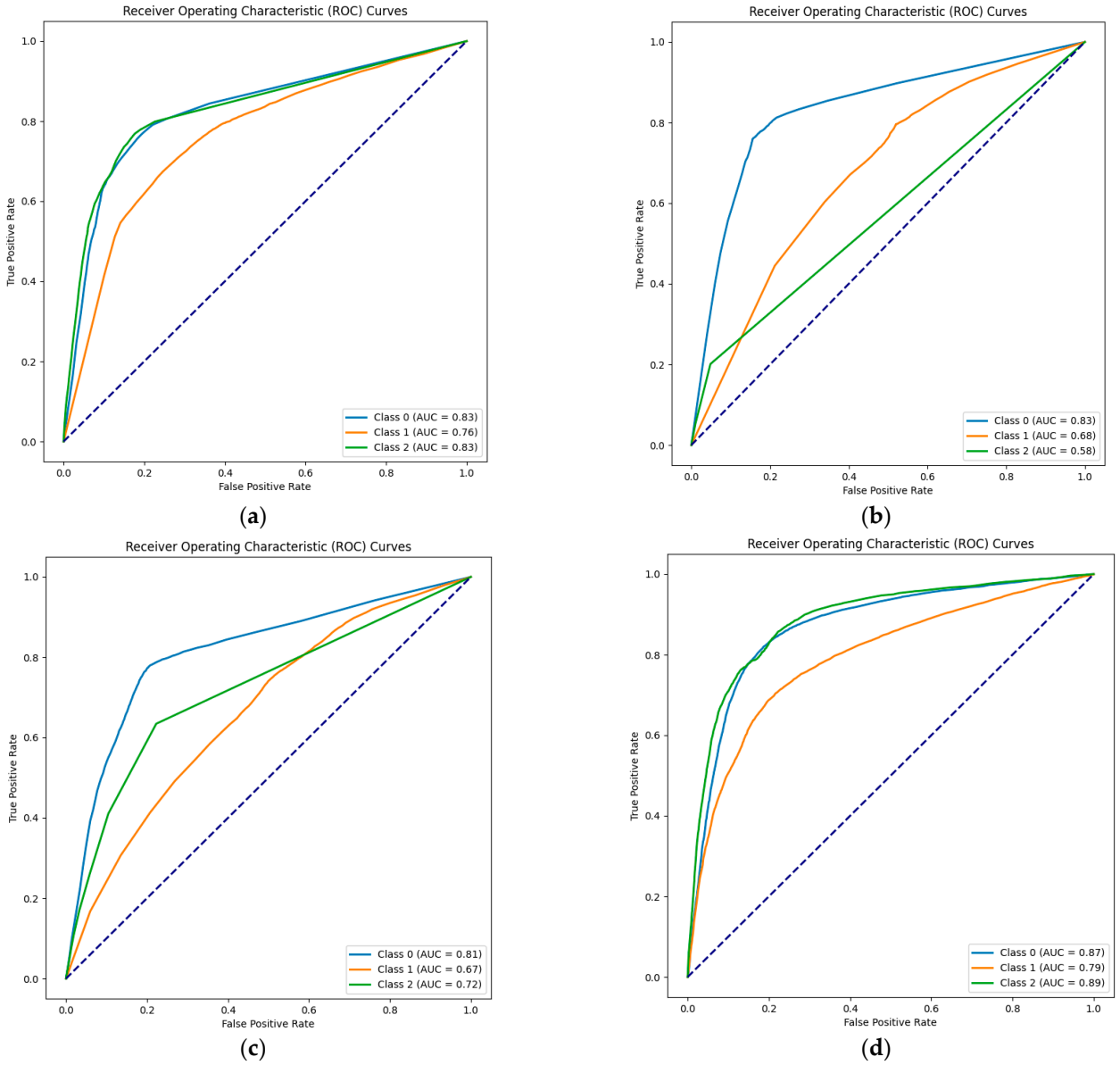


Figure 6. *Cont.*

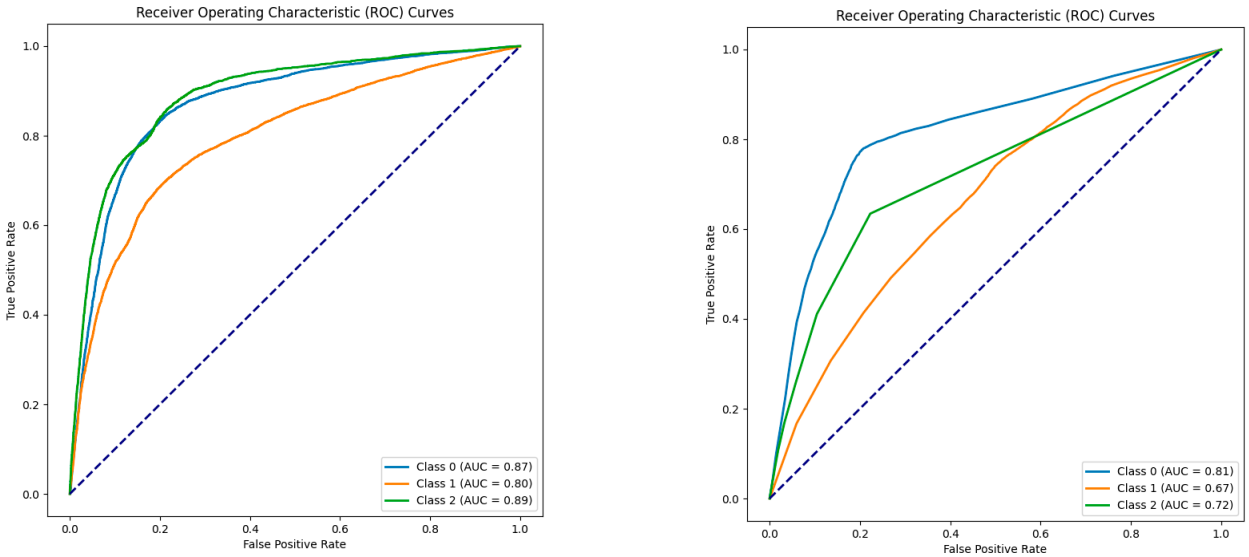


Figure 6. ROC curves for the global FL models of Use Case 1, Part 2: (**A**) Test Case 10; (**b**) Test Case 11; (**c**) Test Case 12, 13, 14; (**d**) Test Case 15; (**e**) Test Case 16; (**f**) Test Case 17.

Table 7. Experimental results for test case scenarios for Part 2 of Use Case 1.

Test Case	Accuracy	F1 Score	Precision	Recall	MSE	R^2
10	0.6876	0.6581	0.6937	0.6382	0.3369	0.2866
11	0.6385	0.4754	0.6070	0.5215	0.3778	0.2001
12	0.6134	0.4515	0.4121	0.4999	0.4044	0.1437
13	0.6203	0.4573	0.4162	0.5077	0.4019	0.1489
14	0.6146	0.4524	0.6137	0.4993	0.4023	0.1481
15	0.7039	0.6762	0.7119	0.6557	0.3170	0.3287
16	0.7132	0.6903	0.7134	0.6749	0.3122	0.3388
17	0.6499	0.5669	0.7012	0.5450	0.3619	0.2336

In general, the experimental results confirm that using a combination of local trained models with or without weights allows us to eliminate malicious behavior in the system and ensures the high quality of the global FL model even though some of the local trainers have datasets with lower quality or try to compromise the system results. The different scenarios can be chosen depending on the specific status of the participants in the system and the wish of the requestor to share a validation dataset or not.

5.4. Experimental Results for Credit Card Fraud Using FBLearn Platform

Use Case 2 aims to experiment with the same approaches for global FL model aggregation but using a logistic regression. Logistic regression is particularly well suited for binary classification problems for which the outcome variable has two classes and provides a straightforward probability estimate for class membership. The dataset for this use case is the Credit Card Fraud dataset [44]. The original dataset is split into three training datasets for three trainers (nodes): train1, train2, and train3 (csv files) and a validation file which is used in the experiments based on validations for global FL model aggregations. The datasets for each trainer are presented in Table 8. All data files are preprocessed and consist of 30 features: numeric columns, representing credit card transactions, the cardholder's personal data, financial and demographic data, historical data for the financial profile of the customer, and a target column of value of {0, 1}, in which a value of 0 denotes there is no fraud and a value of 1 denotes fraud. The data samples in the first dataset (train1) are significantly more than the samples in the other two. The training dataset of this use case

is appropriate for a binary case logistic regression that outputs a result as true or false for each prediction. The logistic regression parameters used for the experimental evaluation in Use Case 2 are based on the default settings of scikit-learn for the relevant Python class. A large-scale bound-constrained optimization solver is used due to its robustness with l2 regularization penalties, and the maximum number of iterations for the solvers to converge is set to 100. Hyperparameter tuning is not applied in the presented experimental evaluation as it focuses on the distributed learning and global model aggregation. The same parameters are used in model training, testing, and validation.

Table 8. Training and validation datasets for Use Case 2.

Dataset File Name	Data Rows Count	Number of Features
train1	342,884	30
train2	84,443	30
train3	84,443	30
validate	56,864	30

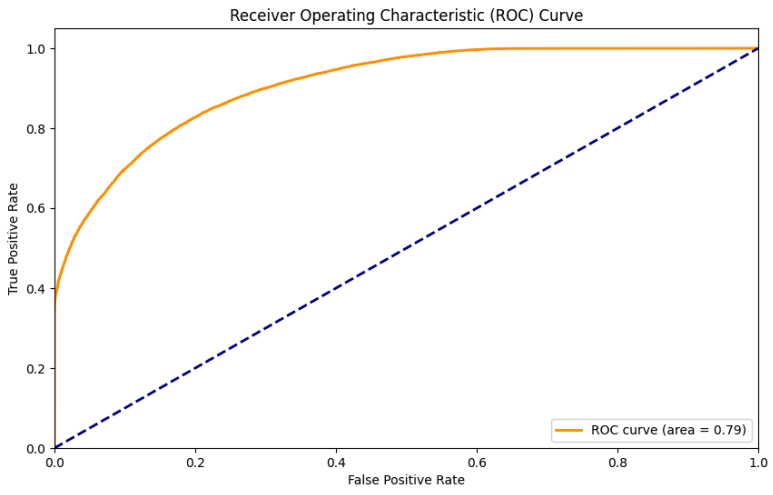
In order to assess the quality of the data in the different training datasets, the same two approaches as in Use Case 1 are used:

- Approach 1: The same datasets are used for training and validation

Each of the datasets is split into training and testing data using a ratio of 80:20. A logistic regression model is trained on the training samples and predictions are compared with the test sample's target values. Table 9 shows the results for the evaluation metrics and Figure 7 shows the ROC curves for the data sources.

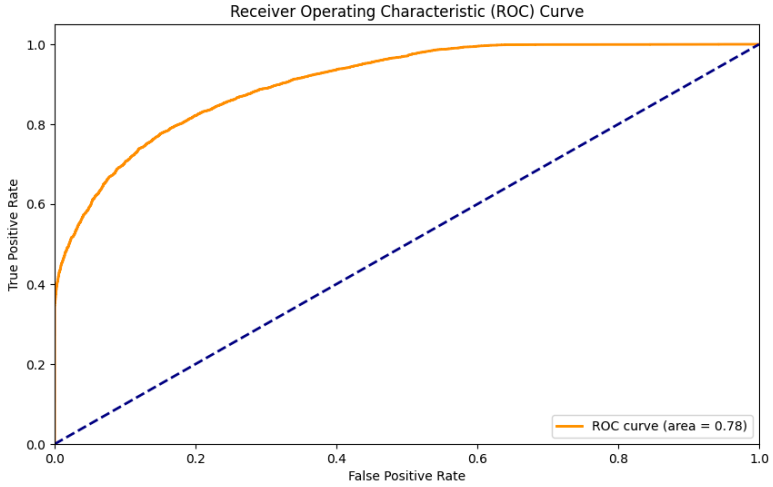
Table 9. Results for dataset assessment for Approach 1, Use Case 2.

File/Data	Accuracy	F1 Score	Precision	Recall	MSE	R^2
train1	0.7933	0.8169	0.7363	0.9175	0.2067	0.1730
train2	0.7853	0.8114	0.7289	0.9151	0.2147	0.1411
train3	0.7848	0.8098	0.7268	0.9142	0.2152	0.1391

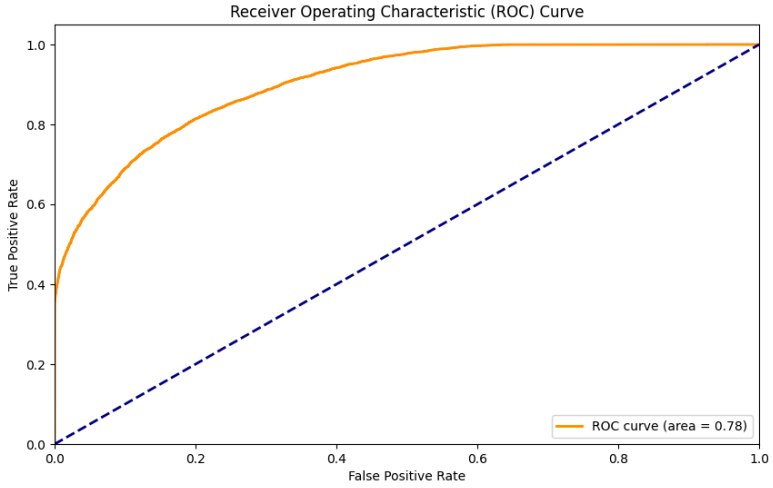


(**a**)

Figure 7. *Cont.*



(**b**)



(**c**)

Figure 7. ROC curves for dataset assessment for Approach 1, Use Case 2. Datasets: (**a**) train1; (**b**) train2; (**c**) train3.

- Approach 2: Validation using validation file

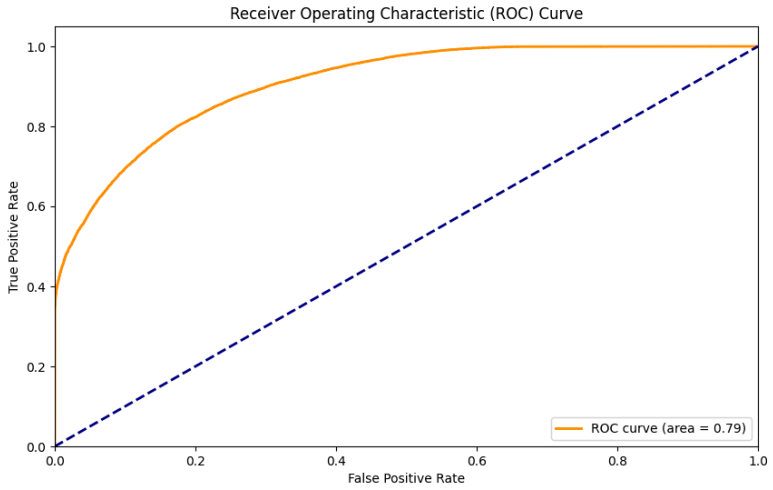
For each dataset, a logistic regression model is trained using all the samples from the dataset. Predictions are calculated for the validation samples and are compared with the target feature in the validation dataset. The results for the selected evaluation metrics are given in Table 10 and the ROC curves in Figure 8.

Table 10. Results for dataset assessment for Approach 2, Use Case 2.

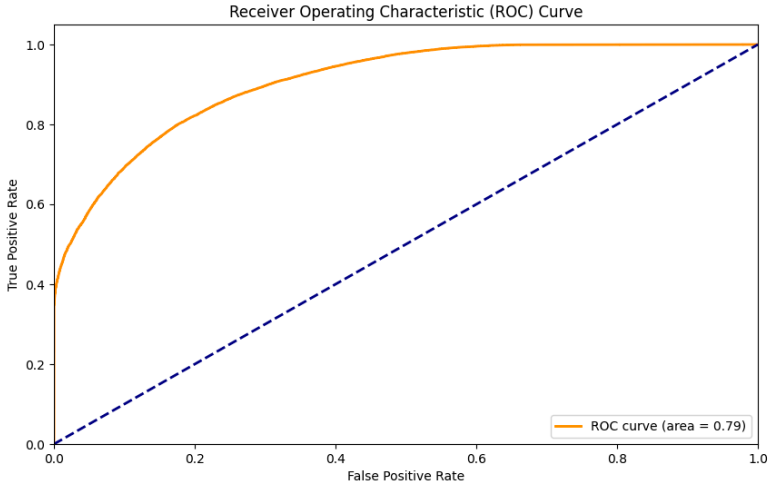
File/Data	Accuracy	F1 Score	Precision	Recall	MSE	R^2
train1	0.7900	0.8134	0.7303	0.9178	0.2100	0.1598
train2	0.7897	0.8128	0.7309	0.9154	0.2103	0.1587
train 3	0.7897	0.8130	0.7306	0.9163	0.2103	0.1588

Both assessment approaches show comparable results. The three data sources have almost equal evaluation metrics and ROC curves.

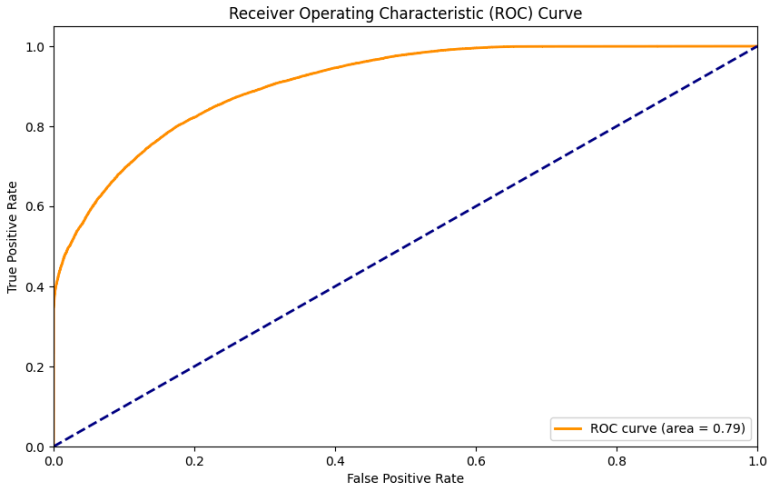
The main hypothesis of Use Case 2 is to evaluate the suggested approaches for global FL model aggregation when the logistic regression is used for distributed learning. The test case scenarios for Use Case 2 are presented in Table 11.



(**a**)



(**b**)



(**c**)

Figure 8. ROC curves for dataset assessment for Approach 2, Use Case 2. Datasets: (**a**) train1; (**b**) train2; (**c**) train3.

Table 11. Test case scenarios for Use Case 2.

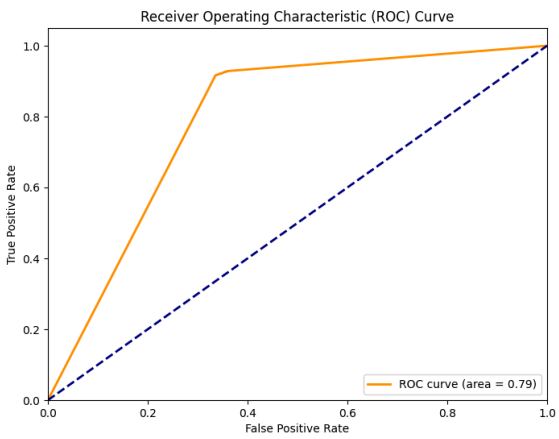
Scenario	Global Model Aggregation	Validation	Weights	Weight Type
18	Combine (Algorithms 1 and 2)	Validate dataset	No	n/a
19	Combine (Algorithms 3 and 4)	Validate dataset	Yes	MSE
20	Combine (Algorithm 5)	Local test datasets	Yes	MSE
21	Combine (Algorithm 5)	Local test datasets	Yes	MSE
22	Combine (Algorithm 5)	Local test datasets	Yes	MSE
23	Ensemble (Algorithm 6)	Validate	No	n/a
24	Ensemble (Algorithms 6 and 7)	Validate	Yes	MSE
25	Ensemble (Algorithms 6, 8 and 9)	Validate	Yes	MCC

The experimental results given in Table 12 show that the global FL model aggregation approaches are applicable for distributed model training using a logistic regression and result in a high-quality global FL model. The different approaches can be used in different scenarios in real cases depending on the requestor's intention to share their validation data or not.

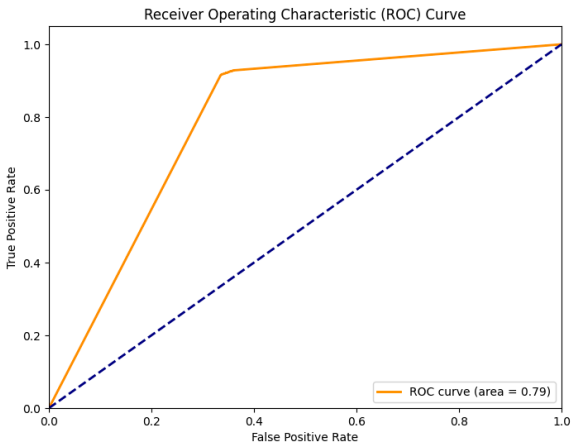
Table 12. Experimental results for test case scenarios for Use Case 2.

Test Case	Accuracy	F1 Score	Precision	Recall	MSE	R^2
18	0.7897	0.7864	0.8097	0.7900	0.2103	0.1589
19	0.7897	0.7863	0.8096	0.7900	0.2103	0.1587
20	0.7897	0.7864	0.8095	0.7900	0.2103	0.1587
21	0.7897	0.7864	0.8095	0.7900	0.2103	0.1587
22	0.7896	0.7863	0.8095	0.7899	0.2104	0.1585
23	0.7897	0.7863	0.8096	0.7900	0.2103	0.1586
24	0.9436	0.9434	0.9479	0.9437	0.0564	0.7743
25	0.9779	0.9779	0.9784	0.9780	0.0221	0.9117

The resulted final global models from the test cases of Use Case 2 show the quality of the initial training dataset files is not better than the quality of the final models if the evaluation metrics and the ROC curves (Figure 9) are compared. The ROC curves of the final models are closer to the top left corner of the frame, which is an indicator of a better performance, especially for Test Cases 24 and 25.



(**a**)



(**b**)

Figure 9. *Cont.*

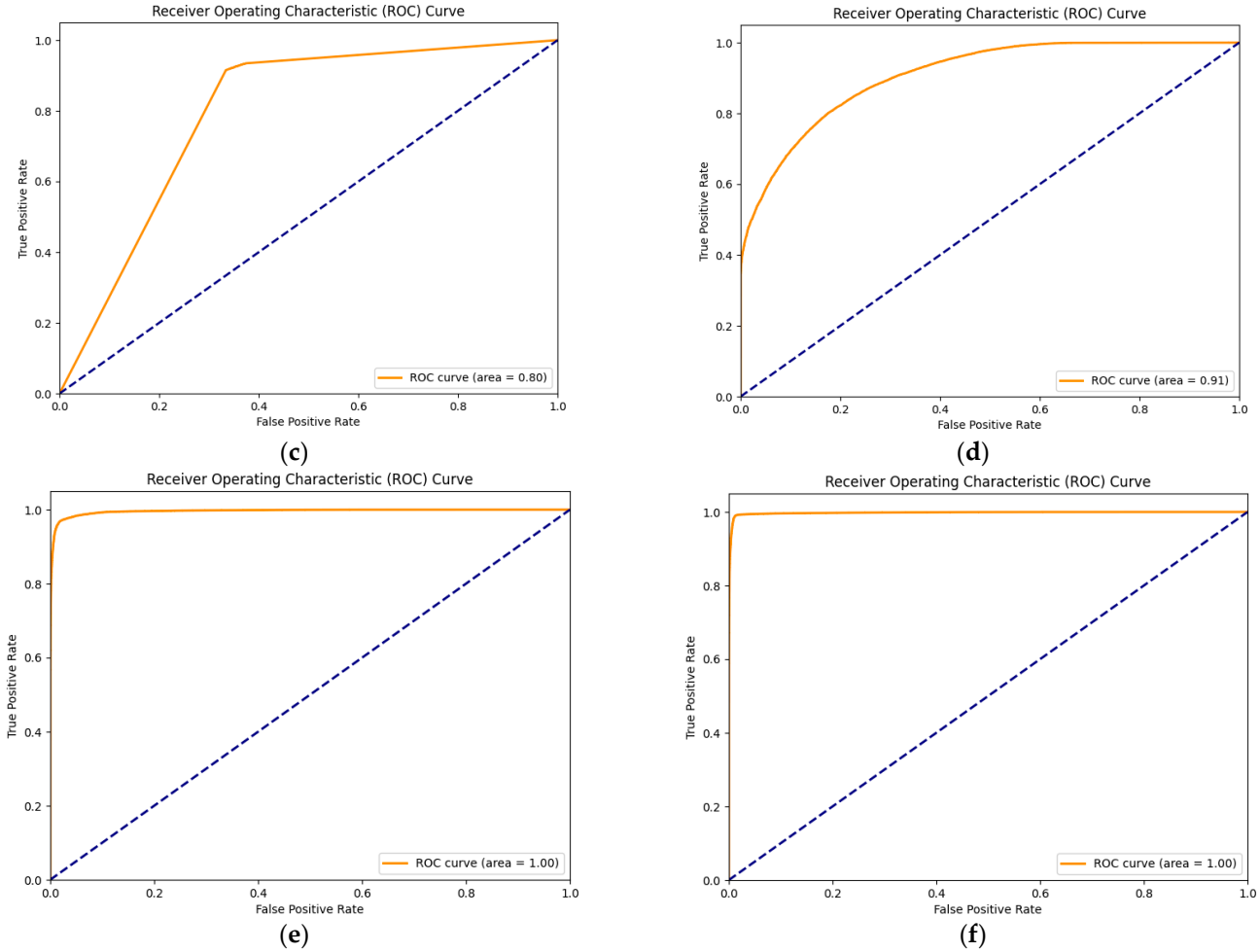


Figure 9. ROC curves for the global FL models of Use Case 2: (**a**) Test Case 18; (**b**) Test Case 19; (**c**) Test Case 20, 21, and 22; (**d**) Test Case 23; (**e**) Test Case 24; (**f**) Test Case 25.

6. Conclusions

Blockchain-based federated learning has gained a great amount of attention recently due to the resulting enhanced data security. However, model accuracy and model security remain challenges in implementing the BFL system, especially when the trainers participate with varying data quality or some of them misbehave, applying adversarial or poisoning attacks. The existing studies in the field use various consensus mechanisms, reputation evaluations, or modifications of federated averaging for global model aggregation. In comparison, the suggested global aggregation strategies based on combining or ensembling the local models offer several unique advancements, particularly in model aggregation, the preservation of privacy, and robustness against attacks.

This paper demonstrates the feasibility of implementing a distributed federated learning solution based on the designed and implemented FBLearn platform. For both use cases of the utilization of the FBLearn platform for credit score modeling and credit card fraud using distributed learning on blockchain, the experiments are designed to evaluate the influence of the suggested approaches for global FL model aggregation and to compare its quality against the locally trained models.

Although other BFL architectures incorporate smart contracts to manage aggregation and incentivize participation, the suggested incorporation of local and globally distributed validation datasets for model aggregation ensures the global FL model reflects the quality of each participating model, reducing the impact of low-quality or malicious contributors. Another key advancement in the proposed aggregation approaches is the emphasis on

adaptive weight calculations and ensemble techniques to enhance the robustness of the global model by implementing dynamic weight calculations based on the quality of local training data. By using MSE and MCC values for assessments of the training data quality in the calculations of the weights, we ensure that the local models trained on lower-quality data do not affect the global FL model, thus improving both its accuracy and robustness. The performance evaluation metrics and ROC curves prove that the proposed approaches successfully limit the influence of low-quality models on the final model. The global FL models have better performing metrics when compared to those of locally trained models based only on private datasets. With the integration of blockchain, the FL models are obtained securely and robustly without the requirement for private data to leave the organization's premises.

Additionally, the FBLearn platform's flexibility in supporting different machine learning algorithms, not only limited to random forest and logistic regression algorithms, provides a broader applicability across various domains and allows different entities (banks, financial institutions, hospitals, or others) to play the role of a requester or trainer, thus enabling the efficient utilization and monetization of private data.

Future development and research with the described platform for distributed federated learning using blockchain will focus on enhancing the security of the platform, in particular, ensuring the ethical and positive intentions of all participants. One key challenge is experimenting with a significantly scaled number of participants in the system which could improve its overall performance and robustness. Another challenge is the standardization of training data across different model types. This issue can be addressed by including pre-defined topics with predefined input formats tailored for specific machine learning models such as fraud detection, credit risk assessment, or customer resilience evaluation. By standardizing the data and model formats, the system can achieve more consistent and accurate results.

Author Contributions: Conceptualization, D.D., M.L. and O.N.; methodology, D.D., M.L. and O.N.; software, D.D.; validation, D.D. and M.L.; formal analysis, D.D.; investigation, D.D., M.L. and O.N.; resources, D.D. and M.L.; data curation, D.D.; writing—original draft preparation, D.D.; writing—review and editing, M.L. and O.N.; visualization, D.D.; supervision, M.L. and O.N.; project administration, M.L.; funding acquisition, M.L. All authors have read and agreed to the published version of the manuscript.

Funding: The research work presented in the paper is funded by European Union—NextGenerationEU via the National Recovery and Resilience Plan of the Republic of Bulgaria under project BG-RRP-2.004-0005 "Improving the research capacity and quality to achieve international recognition and reSilience of TU-Sofia (IDEAS)".

Data Availability Statement: Data are contained within the article.

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

References

1. Haber, S.; Stornetta, W. S. How to time-stamp a digital document. In *Advances in Cryptology-CRYPTO'90*; Menezes, A.J., Vanstone, S.A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1990; Volume 537, pp. 437–455.
2. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System, Technical Report. 2008. Available online: http://bitcoin.org/bitcoin.pdf (accessed on 4 September 2024).
3. Peng, C.; Akca, S.; Rajan, A. SIF: A Framework for Solidity Contract Instrumentation and Analysis. Technical Report, University of Edinburgh, UK, May 2019. Available online: https://arxiv.org/abs/1905.01659 (accessed on 4 September 2024).
4. Rudman, R.; Bruwer, R. Defining Web 3.0: Opportunities and challenges. *Electron. Libr.* **2016**, *34*, 132–154. [CrossRef]
5. Alabdulwahhab, F.A. Web 3.0: The decentralized web blockchain networks and protocol innovation. In Proceedings of the 2018 1st International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 4–6 April 2018; pp. 1–4.
6. Rumelhart, D.E.; McClelland, J.L. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*; MIT Press: Cambridge, MA, USA, 1987.

7. Fischer, L.; Ehrlinger, L.; Geist, V.; Ramler, R.; Sobieczky, F.; Zellinger, W.; Moser, B. Applying AI in practice: Key challenges and lessons learned. In *Machine Learning and Knowledge Extraction*; Holzinger, A., Kieseberg, P., Tjoa, A., Weippl, E., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2020; Volume 12279, pp. 451–471.
8. Shi, S.; Tse, R.; Luo, W.; D'Addona, S.; Pau, G. Machine learning-driven credit risk: A systemic review. *Neural Comput. Appl.* **2022**, *34*, 14327–14339. [CrossRef]
9. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
10. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
11. Gosselin, R.; Vieu, L.; Loukil, F.; Benoit, A. Privacy and Security in Federated Learning: A Survey. *Appl. Sci.* **2022**, *12*, 9901. [CrossRef]
12. Sun, N.; Wang, W.; Tong, Y.; Liu, K. Blockchain based federated learning for intrusion detection for Internet of Things. *Front. Comput. Sci.* **2023**, *18*, 185328. [CrossRef]
13. Li, Q.; Wang, W.; Zhu, Y.; Ying, Z. BOppCL: Blockchain-enabled opportunistic federated learning applied in intelligent transportation systems. *Electronics* **2024**, *13*, 136. [CrossRef]
14. Wang, L.; Guan, C. Improving security in the internet of vehicles: A blockchain-based data sharing scheme. *Electronics* **2024**, *13*, 714. [CrossRef]
15. Zhu, C.; Zhu, X.; Qin, T. An efficient privacy protection mechanism for blockchain-based federated learning system in UAV-MEC Networks. *Sensors* **2024**, *24*, 1364. [CrossRef]
16. Javed, A.R.; Hassan, M.A.; Shahzad, F.; Ahmed, W.; Singh, S.; Baker, T.; Gadekallu, T.R. Integration of blockchain technology and federated learning in vehicular (IoT) networks: A comprehensive survey. *Sensors* **2022**, *22*, 4394. [CrossRef]
17. Hai, T.; Zhou, J.; Srividhya, S.R.; Jain, S.K.; Young, P.; Agrawal, S. BVFLEMR: An integrated federated learning and blockchain technology for cloud-based medical records recommendation system. *J. Cloud Comput.* **2022**, *11*, 22.
18. Ashraf, E.; Areed, N.F.F.; Salem, H.; Abdelhay, E.H.; Farouk, A. FIDChain: Federated intrusion detection system for blockchain-enabled iot healthcare applications. *Healthcare* **2022**, *10*, 1110. [CrossRef] [PubMed]
19. Bi, L.; Muazu, T.; Samuel, O. IoT: A decentralized trust management system using blockchain-empowered federated learning. *Sustainability* **2023**, *15*, 374. [CrossRef]
20. Li, C.; Yuan, Y.; Wang, F.-Y. Blockchain-enabled federated learning: A survey. In Proceedings of the 2021 IEEE 1st International Conference on Digital Twins and Parallel Intelligence (DTPI), Beijing, China, 15 July–15 August 2021; pp. 286–289.
21. Wu, L.; Ruan, W.; Hu, J.; He, Y. A survey on blockchain-based federated learning. *Future Internet* **2023**, *15*, 400. [CrossRef]
22. Wang, Z.; Yan, B.; Dong, A. Blockchain Empowered Federated Learning for Data Sharing Incentive Mechanism. *Procedia Comput. Sci.* **2022**, *202*, 348–353. [CrossRef]
23. Xu, Y.; Lu, Z.; Gai, K.; Duan, Q.; Lin, J.; Wu, J.; Choo, K.-K.R. BESIFL: Blockchain-empowered secure and incentive federated learning paradigm in IoT. *IEEE Internet Things* **2021**, *10*, 6561–6573. [CrossRef]
24. Tian, Y.; Guo, Z.; Zhang, J.; Al-Ars, Z. 2023. DFL: High-performance blockchain-based federated learning. *Distrib. Ledger Technol. Res. Pract.* **2023**, *2*, 1–25. [CrossRef]
25. Yang, J.; Zhang, W.; Guo, Z.; Gao, Z. TrustDFL: A blockchain-based verifiable and trusty decentralized federated learning framework. *Electronics* **2024**, *13*, 86. [CrossRef]
26. Ouyang, K.; Yu, J.; Cao, X.; Liao, Z. Towards reliable federated learning using blockchain-based reverse auctions and reputation incentives. *Symmetry* **2023**, *15*, 2179. [CrossRef]
27. Wu, C.; Wu, F.; Lyu, L.; Huang, Y.; Xie, X. Communication-efficient federated learning via knowledge distillation. *Nat. Commun.* **2022**, *13*, 2032. [CrossRef]
28. Kim, G.; Kim, Y. The threat of disruptive jamming to blockchain-based decentralized federated learning in wireless networks. *Sensors* **2024**, *24*, 535. [CrossRef]
29. Wan, C.; Wang, Y.; Xu, J.; Wu, J.; Zhang, T.; Wang, Y. Research on privacy protection in federated learning combining distillation defense and blockchain. *Electronics* **2024**, *13*, 679. [CrossRef]
30. Begum, K.; Mozumder, M.A.I.; Joo, M.-I.; Kim, H.-C. BFLIDS: Blockchain-driven federated learning for intrusion detection in IoMT networks. *Sensors* **2024**, *24*, 4591. [CrossRef] [PubMed]
31. Zhang, H.; Zhang, P.; Hu, M.; Liu, M.; Wang, J. FedUB: Federated learning algorithm based on update bias. *Mathematics* **2024**, *12*, 1601. [CrossRef]
32. Marin Machado de Souza, R.; Holm, A.; Biczyk, M.; de Castro, L.N. A systematic literature review on the use of federated learning and bioinspired computing. *Electronics* **2024**, *13*, 3157. [CrossRef]
33. Shang, C.; Gu, F.; Jiang, J. Evolutionary multi-model federated learning on malicious and heterogeneous data. In Proceedings of the 2023 IEEE International Conference on Data Mining Workshops (ICDMW), Shanghai, China, 1–4 December 2023; pp. 386–395.
34. Ahmed, A.; Alabi, O. Secure and scalable blockchain-based federated learning for cryptocurrency fraud detection: A systematic review. *IEEE Access* **2024**, *12*, 102219–102241. [CrossRef]
35. Baabdullah, T.; Alzahrani, A.; Rawat, D.B.; Liu, C. Efficiency of federated learning and blockchain in preserving privacy and enhancing the performance of credit card fraud detection (CCFD) systems. *Future Internet* **2024**, *16*, 196. [CrossRef]

36. Liu, B.; Tang, Q. Secure data sharing in federated learning through blockchain-based aggregation. *Future Internet* **2024**, *16*, 133. [CrossRef]
37. Wang, H.; Gao, H.; Ma, T.; Li, C.; Jing, T. A hierarchical blockchain-enabled distributed federated learning system with model-contribution based rewarding. *Digit. Commun. Netw.* **2024**, *in press*. [CrossRef]
38. Djolev, D.; Lazarova, M.; Nakov, O. Blockchain based trusted distributed machine learning for credit scoring. In Proceedings of the 2023 International Scientific Conference on Computer Science (COMSCI), Sozopol, Bulgaria, 18–20 September 2023.
39. Djolev, D.; Lazarova, M.; Nakov, O. Federated learning for credit scoring model using blockchain. In *Communications in Computer and Information Science: Optimization, Learning Algorithms and Applications*; Pereira, A., Mendes, A., Fernandes, F., Pacheco, M., Coelho, J., Lima, J., Eds.; Springer: Cham, Switzerland, 2024; Volume 1981.
40. FELT—Federating Learning Token. Available online: https://github.com/FELT-Labs/federated-learning-token (accessed on 4 September 2024).
41. Gencturk, M.; Sinaci, A.; Cicekli, N. BOFRF: A novel boosting-based federated random forest algorithm on horizontally partitioned data. *IEEE Access* **2022**, *10*, 89835–89851. [CrossRef]
42. Matthews, B. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Et Biophys. Acta (BBA)—Protein Struct.* **1975**, *405*, 442–451. [CrossRef]
43. Home Credit Default Risk. Available online: https://www.kaggle.com/competitions/home-credit-default-risk (accessed on 4 September 2024).
44. Credit Card Fraud Detection Dataset 2023. Available online: https://www.kaggle.com/datasets/nelgiriyewithana/credit-card-fraud-detection-dataset-2023/data (accessed on 4 September 2024).
45. Data Preprocessing for Credit Score Classification. Available online: https://www.kaggle.com/code/bugaiovaolena/data-preprocessing-for-credit-score-classification (accessed on 4 September 2024).