PRIVACY-PRESERVING HORIZONTAL FEDERATED LEARNING METHODOLOGY THROUGH A NOVEL BOOSTING-BASED FEDERATED RANDOM FOREST ALGORITHM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT GENÇTÜRK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

JANUARY 2023

Approval of the thesis:

PRIVACY-PRESERVING HORIZONTAL FEDERATED LEARNING METHODOLOGY THROUGH A NOVEL BOOSTING-BASED FEDERATED RANDOM FOREST ALGORITHM

submitted by **MERT GENÇTÜRK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Nihan Kesim Çiçekli
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Özgür Ulusoy
Computer Engineering, Bilkent University _____

Prof. Dr. Nihan Kesim Çiçekli
Computer Engineering, METU _____

Prof. Dr. Pınar Karagöz
Computer Engineering, METU _____

Prof. Dr. Ferda Nur Alpaslan
Computer Engineering, METU _____

Assist. Prof. Dr. Engin Demir
Computer Engineering, Hacettepe University _____

Date: 04.01.2023

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Mert Gençtürk

Signature :

ABSTRACT

PRIVACY-PRESERVING HORIZONTAL FEDERATED LEARNING METHODOLOGY THROUGH A NOVEL BOOSTING-BASED FEDERATED RANDOM FOREST ALGORITHM

Gençtürk, Mert
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Nihan Kesim Çiçekli

January 2023, 136 pages

In this thesis, a novel federated ensemble classification algorithm for horizontally partitioned data called Boosting-based Federated Random Forest (BOFRF) is proposed, which not only increases the predictive power of all participating sites, but also provides significantly high improvement on the predictive power of sites having unsuccessful local models. In this regard, a federated version of random forest, which is a well-known bagging algorithm, is implemented by adapting the idea of boosting to it. In the integration step, a novel aggregation and weight calculation methodology is introduced that assigns weights to local classifiers based on their classification performance at each site instead of proportioning them with the sample size or site index without increasing the communication or computation cost. To increase the predictive power of the federated models built through the proposed algorithm, a personalized implementation is presented where each participant fine-tunes the hyperparameters of BOFRF locally and come up with a better-performing federated model on their own datasets. In addition, a clustered extension is proposed where participants are clustered according to their data distribution similarities or differences prior to running the algorithm. Finally, to prevent security breaches from happening and increase the

level of privacy, two different implementations are proposed for BOFRF, which are centralized implementation with a trusted third party and decentralized implementation using secure sum protocol. The performance of the proposed solution was evaluated in different federated environments that were set up by using four healthcare datasets. The empirical results show that the BOFRF algorithm and its extensions improve the predictive power of local random forest models in all cases. The advantage of the proposed methodology is that the level of improvement it provides for sites having unsuccessful local models is significantly high unlike existing solutions.

Keywords: Federated learning, Ensemble learning, Machine learning, Random Forest classification, Privacy-preservation

ÖZ

YENİ BİR GÜÇLENDİRMEYE DAYALI BİRLEŞİK RASTGELE ORMAN ALGORİTMASIYLA GİZLİLİĞİ KORUYAN YATAY BİRLEŞİK ÖĞRENİM YÖNTEMİ

Gençtürk, Mert
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Nihan Kesim Çiçekli

Ocak 2023 , 136 sayfa

Bu çalışmada, yatay olarak bölümlenmiş veriler için *Güçlendirmeye Dayalı Birleşik Rastgele Orman (BOFRF)* adı verilen, yalnızca tüm katılımcıların tahminleme gücünü artırmakla kalmayıp aynı zamanda başarısız yerel modellere sahip katılımcıların tahmin gücü üzerinde önemli ölçüde yüksek iyileştirme sağlayan yeni bir birleşik topluluk sınıflandırma algoritması önerilmiştir. Algoritma, iyi bilinen bir torbalama algoritması olan rastgele ormana artırma fikri uyarlanarak, onun bir birleşik öğrenme versiyonu olarak geliştirilmiştir. Entegrasyon adımında, iletişim ve hesaplama maliyetini artırmayan, yerel sınıflandırıcıların her bir katılımcının verisi üzerindeki sınıflandırma performansına dayalı yeni bir ağırlık hesaplama ve birleştirme metodolojisi sunulmuştur. Çalışmada ayrıca, önerilen algoritma aracılığıyla oluşturulan birleşik modellerin tahmin gücünü artırmak için, her bir katılımcının BOFRF'in hiper parametrelerine yerel olarak ince ayar yaptığı ve kendi veri kümesinde daha iyi performans gösteren bir birleşik model oluşturduğu kişiselleştirilmiş bir BOFRF algoritması sunulmuştur. Ek olarak, katılımcıların algoritmayı çalıştırmadan önce veri

dağılımı benzerliklerine veya farklılıklarına göre kümelenmesini sağlayan bir uzantı da önerilmiştir. Son olarak, güvenlik ihlallerinin oluşmasını önlemek ve mahremiyet seviyesini artırmak için BOFRF için güvenilir bir üçüncü taraf ile merkezileştirilmiş uygulama ve güvenli toplam protokolü kullanılarak merkezi olmayan uygulama olmak üzere iki farklı uygulama önerilmiştir. BOFRF'in performansı, sağlık sektöründen dört ayrı veri seti kullanılarak kurulan farklı federe ortamlarda değerlendirilmiştir. Sonuçlar, BOFRF algoritmasının ve uzantılarının, her durumda yerel rastgele orman modellerinin tahmin gücünü geliştirdiğini göstermiştir. Önerilen metodolojinin avantajı, başarısız yerel modellere sahip katılımcılar için sağladığı iyileştirme seviyesinin mevcut çözümlere kıyasla önemli ölçüde yüksek olmasıdır.

Anahtar Kelimeler: Birleşik öğrenme, Topluluk öğrenimi, Makine öğrenimi, Rastgele Orman sınıflandırması, Gizliliğin korunması

To my dear wife, Ceyda

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and appreciation to my supervisor Prof. Dr. Nihan Kesim Çiçekli for her guidance, encouragement and continuous support throughout this study. I would like to extend my gratitude to Prof. Dr. Pınar Karagöz and Assist. Prof. Dr. Engin Demir for providing me with crucial insights and comments throughout the study.

I am deeply grateful to my colleague Ali Anıl Sınacı, who always helped me with his support and stimulating suggestions during the research and writing of this thesis. I am also highly thankful to Gökçe Banu Laleci Ertürkmen and all my other colleagues at SRDC Corp. for their invaluable support.

My biggest gratitude goes to my dear wife Ceyda for her friendship, encouragement, patience and always being there for me. None of this would have been possible without her endless support and belief in me. I would also like to thank my sweet nephews Toprak and Taylan Dündar for always bringing joy to my life.

Finally, I would like to express my special thanks to my dear parents, my brother Semih Gençtürk, and my friends Ahmet Sami Küçük and Aydan Kaya Küçük for their help, support and cheerful presence through the course of this study.

The research leading to the results presented in this thesis was supported by the FAIR4Health project (Improving Health Research in EU through FAIR Data), which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement Number 824666.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AdaBoost	Adaptive Boosting
ADL	Activities of Daily Living
AUC	Area Under Curve
BMI	Body Mass Index
BOFRF	Boosting-Based Federated Random Forest
BOPPID	Boosting-Based Privacy-Preserving Integration of Distributed Data
CA	Certificate Authority
CFL	Clustered Federated Learning
CGI	Cognitive Impairment
CM	Confusion Matrix
DT	Decision Tree
EDC	Euclidean Distance of Decomposed Cosine Similarity
FedAvg	Federated Averaging
FedMA	Federated Matched Averaging
FL	Federated Learning
FTL	Federated Transfer Learning
GBT	Gradient-Boosted Trees
GI	Gini Index
HFL	Horizontal Federated learning
IFCA	Iterative Federated Clustering Algorithm
IG	Information Gain
IID	Independent Identical Distributions
IoT	Internet of Things

MERF	Mixed-Effects Random Forest
MultBoost	Multiparty Boosting
Non-IID	Non-Independent Identical Distributions
MAML	Model-Agnostic Meta-Learning
ML	Machine Learning
NH	Nursing Home
Per-FedAvg	Personalized Federated Averaging
PFL	Personalized Federated learning
RF	Random Forest
SHELTER	Services and Health for Elderly in Long TERm care
SGD	Stochastic Gradient Descent
TTP	Trusted Third Party
UCI	University of California, Irvine
VFL	Vertical Federated learning

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

In today's technology age, information systems are incessantly collecting massive amount of data in their repositories. The analysis of such data and extracting knowledge from them has become an important concept for many different domains such as security, finance, healthcare, and transportation where data are clustered in a number of different systems and organizations. As the size of data is huge, traditional statistical analysis methodologies cannot be used [1]. Instead, enhanced algorithms are needed to process vast amount of data. Machine learning algorithms can be used to interpret information by building mathematical models on existing data to make predictions or decisions without human intervention [2].

An example domain where machine learning is applied on large datasets is healthcare. Clinical data have various secondary uses for a significant number of diverse sectors of both public and private interest, as they are rich sources of longed-for intelligence, especially if they are combined at large scale. This also creates unprecedented opportunities for a wide range of applications including chronic disease management, clinical decision support services, advanced analytics systems and many more. However, sharing sensitive health data is strictly controlled by laws and regulations in all around the world such as the Health Insurance Portability and Accountability Act (HIPAA) [3] enacted by the United States Congress in 1996, and the General Data Protection Regulation (GDPR) [4] enforced by the European Union in 2018 to preserve the privacy of patient information. Due to the sensitive nature of these data, since they contain personal information of patients such as demographics, conditions, medica-

tions, and other health-related information, they typically remain in heterogeneous and isolated data silos, hampering the extraction of their inherent actionable insights and intelligence [5]. Consequently, data providers are reluctant to allow clinical data to leave their premises, hence patient data cannot be shared between clinical sites or collected by a third-party. This situation prevents the groundbreaking achievements of machine learning to advance the predictive capabilities of clinical data beyond a laboratory setting.

Privacy has become an important issue in many machine-learning applications as in all other fields that deal with sensitive data. In order to prevent the reveal of sensitive data to the outside world, several privacy preservation methods have been developed, such as k-anonymity and l-diversity [6]. Such techniques mainly achieve privacy preservation by modifying or removing some parts of the original data. Although the transformation of data may provide privacy, it may reduce the quality of the data, which is formally known as utility, causing defective results in machine learning. In addition, these methods are open to adversarial attacks that are used to reveal hidden sensitive information, especially when the attacker has some background knowledge of the data or when combined with some publicly available data [7]. Therefore, new algorithms are needed to enable data owners whose data are in different data silos to perform machine learning operations collaboratively without sharing their sensitive data in neither raw nor encrypted format.

A way of dealing with these challenges is to apply the concept of federated learning. The aim of federated learning is to build a joint machine learning model based on the data residing at multiple sites by exchanging only information about locally trained models, not actual data [8]. Federated learning has been proven to be a powerful mechanism as it enables multiple parties to build a better global model than the individual local models in a privacy-preserving setting. It can be applied to both horizontally partitioned data, where the datasets at different sites share the same set of features for different ID spaces, and vertically partitioned data, where the datasets contain different sets of features for the same ID space [9]. Although federated learning is a new concept proposed by Google in 2016 [10], in the literature, there have been many implementations of it using different techniques and focusing on different aspects, such as providing more security [11, 12], decreasing communication cost

[13, 14], reducing computation cost [15], and increasing model performance.

The application of federated learning on ensemble methods is a common practice with the goal of increasing the power of the predictive model. The idea of ensemble learning is to combine the predictions of multiple classifiers to produce a single classifier which is more accurate than any of the individual classifiers constituting the ensemble [16]. Bagging and Boosting are the two most popular techniques among the many ensemble methods developed by researchers. Most state-of-the-art horizontal federated learning approaches utilizing ensemble methods are built on boosting techniques, especially AdaBoost, which is a well-known and extensively studied algorithm in the literature [17, 18]. In these approaches, the sites first execute the AdaBoost algorithm locally on their training data, and then the local models are transferred to a third-party coordinator or shared between sites to build an integrated model. Various algorithms handle this integration process differently. For instance, AdaBoost.PL aims to combine like-minded classifiers, hence sorts the weak classifiers in the local models with respect to their weights and merges the classifiers with similar correctness at the same sorted level [17]. BOPPID focuses on the data distribution differences between the sites during the integration and assigns weights to the local models according to the sampling size of the sites and by giving more importance to the site's own local model [18]. However, the weakness of existing federated solutions utilizing ensemble methods is that although they can successfully improve the prediction power of local models when the datasets of sites are balanced and of good quality (i.e., the local models are already above a certain accuracy threshold), they usually fail to provide the same level of improvement to the models of sites that have an unsuccessful classifier because of their bad quality or imbalanced data. For example, BOPPID, which is one of the most successful federated implementations of AdaBoost [18], always assigns more weight to the site's own local model compared to any other local model with the same sample size, because it assumes that other sites might have a different data distribution; hence, they should not be given more importance. This prevents a site having an unsuccessful classifier from taking advantage of successful classifiers of other sites to improve its local model, which results in the site not achieving an accuracy as good as the others.

1.2 Proposed Methods and Models

In this study, first, a novel and practical federated ensemble classification algorithm for horizontally partitioned data called Boosting-based Federated Random Forest (BOFRF) is proposed, which not only increases the predictive power of all participating sites, but also provides significantly high improvements on the predictive power of sites having unsuccessful local models. In this regard, a federated version of random forest, which is a well-known bagging algorithm, is implemented by adapting the idea of boosting to it. In the integration step, a novel aggregation and weight calculation methodology is introduced that assigns weights to local classifiers based on their classification performance at each site instead of proportioning them with the sample size or site index without increasing the communication or computation cost. The algorithm operates in six steps: First, random forest models consisting of a number of decision trees are built at each site. Second, these local models are shared with every other site and the performance statistics (true/false positive/negative values) of all decision trees are calculated at each site. Then, global statistics are computed by aggregating local performance statistics, the weight for each decision tree is calculated by utilizing the Matthews correlation coefficient (MCC), and finally the final federated ensemble classifier is generated.

In the integration step, weak classifiers having an MCC value below a certain threshold value are removed to improve the prediction performance of the final federated ensemble classifier. In the experiments, it was discovered that instead of producing a single global federated model with a fixed threshold value, different participants could have federated models that produce better results by personalizing the global model using different threshold values. In this regard, second in this study, a personalized BOFRF implementation is presented where each participant fine-tunes the hyperparameters of BOFRF, including the threshold, locally to come up with a better-performing federated model on their own datasets.

The proposed BOFRF algorithm provides an adequate level of privacy in its base and personalized forms as the actual data is not shared among the participants. However, sharing evaluation metrics in the presence of a sneaky site in the federated environment may still result in a privacy breach. To prevent privacy violations from hap-

pening and increase the level of privacy, two different implementations are proposed for BOFRF: centralized implementation with a trusted third party and decentralized implementation using secure sum protocol. In centralized implementation with a trusted third party, participating sites send their decision trees and confusion matrices to an orchestrator who is responsible for sending the decision trees to the other sites, retrieving output confusion matrices, and calculating the final confusion matrix for each decision tree without knowing anything about the information provided by the sites. In decentralized implementation, instead of communicating with a third party, participating sites communicate with each other in a circular way. This approach uses the secure sum protocol, which allows participating sites to calculate the sum of their individual data without exposing their data to other sites.

In federated environments, some participants may produce well-performing local models, so in the federated model, they may not need weak classifiers from participants whose data distribution is quite different than their own. On the other hand, some other participants may generate poorly performing local models, so they may need weak classifiers from participants whose data distribution is different than their own. Consequently, fourth and lastly in this study, a clustered BOFRF implementation is proposed, where participants are grouped into clusters according to their data distribution differences. To achieve this, a characterization vector definition is introduced and data distribution difference between different sites is calculated. Then, the participants form a cluster by choosing among other participants according to their needs and develop their federated model.

1.3 Contributions and Novelties

The primary contributions of this work can be listed as follows:

- A novel Boosting-based Federated Random Forest (BOFRF) algorithm is proposed that combines both bagging and boosting ensemble techniques and adapts them to horizontal federated learning to enable different sites to perform joint machine learning operations without sharing any real data between them or with a third party, hence preserving privacy. Specifically, the decision trees

in each local random forest model are considered as weak classifiers and their weights are calculated in a federated manner.

- A novel aggregation and weight calculation methodology is introduced that enables participating sites to generate a global federated model that can improve prediction capability of all sites, regardless of whether they had a successful or unsuccessful local model.
- A personalized version of BOFRF is proposed to further increase the predictive power of participating sites.
- Two privacy-preserving implementations of the proposed algorithm are provided to prevent a sneaky site in the federated environment from identifying any individual in the participants' datasets and thus violating privacy.
- An extension of BOFRF, namely clustered BOFRF, is proposed to cluster the participating sites in the federated environment according to their data distribution similarities or differences prior to running the BOFRF algorithm.
- The performance of the proposed algorithm is evaluated in several federated environments that are set up by using four healthcare datasets.
- The experiments show that the proposed algorithm improves the prediction power of the baseline local random forest model in all cases and produces better results than similar approaches. In particular, the percentage of improvement is significantly high for sites having unsuccessful local models because of their poor quality or imbalanced data.

1.4 The Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 presents the general concepts and the related work in the field of data management, ensemble learning, federated learning, application of ensemble strategies in federated settings, real-world federated learning applications in different domains, followed by personalized and clustered federated learning. The proposed Boosting-based Federated Random Forest algorithm, its personalized implementation, and the hyperparameters that can be tuned in

it are explained in Chapter 3. In this chapter, complexity anaysis of BOFRF is studied as well. In Chapter 4, the details of the privacy-preserving implementations of BOFRF, i.e., centralized implementation with a trusted third party and decentralized implementation with secure-sum protocol are presented, while the clustered extension of BOFRF is explained in Chapter 5. The experimental studies and the results obtained from observational and statistical experiments are presented in Chapter 6 and 7, respectively. Finally, the limitations and potential future improvements to the proposed approach are discussed and the thesis is concluded in Chapter 8.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, general concepts and related work performed in the field of ensemble learning, federated learning, and federated learning utilizing ensemble techniques are presented.

2.1 Data Management

In information systems, data is managed either centrally or in distributed manner. Machine learning techniques differ based on this management style of data. Collecting data residing at dispersed silos through proprietary or well-established standard interfaces into a central database is a way to manage data centrally. On the other hand, data continues to live in silos and performing any kind of machine learning tasks by connecting to these disparate data sources is the key concept of distributed data management within machine learning.

2.1.1 Central Data Management

Managing data centrally is the easiest approach for nearly any kind of data processing task. However, keeping all data in a central database or collecting the distributed data in a central database for processing can create major problems. Among them, the most concerned issue is privacy. Keeping data at one place can be an effective approach for machine learning operations and preserving privacy, as all tasks can be computed in an isolated environment where no data goes out. However, when the security of this environment is compromised, sensitive data can be leaked to the outside

world, resulting in a significant privacy breach. Therefore, distributed applications have become increasingly popular for protecting sensitive data. However, these applications, face another privacy issue, where data owners do not want to share their sensitive data for joint machine learning operations due to technical and ethico-legal challenges. For this reason, researchers have recently focused on federated learning techniques that leave data in silos and thus do not present any privacy violation concerns.

2.1.2 Distributed Data Management

Distributed data is managed either horizontally or vertically. Horizontally partitioned data is managed by different data owners through the same set of features. Healthcare information systems are a good example of horizontal data management, where different hospitals maintain the same set of features for patients such as demographic information like name, surname, gender, age, address etc., and diagnostic data including laboratory results and prescriptions. Fig. 2.1 illustrates an example of a horizontally partitioned data.

HOSPITAL A				
SSN	**Gender**	**Age**	**Diagnosis**	**…**
13579	M	29	Type 1 Diabetes	…
…	…	…	…	…
24680	F	53	Hypertension	…

HOSPITAL B				
SSN	**Gender**	**Age**	**Diagnosis**	**…**
12895	M	40	Atherosclerosis	…
…	…	…	…	…
71903	M	32	Asthma	…

Figure 2.1: Horizontally partitioned data.

In contrast, in vertically partitioned data, one set of features is managed by one data owner, while another set of features is managed by another data owner. Healthcare data can also be a good example of vertically partitioned data. For example, a patient's

demographic information and examination data including diagnosis may be held by *Hospital A*, while laboratory test results and/or radiology images may be held by *Hospital B* as illustrated in Fig. 2.2. This means the patient summary is split vertically between different databases.

HOSPITAL A		
SSN	**Diagnosis**	**…**
13579	Type 1 Diabetes	…
…	…	…
24680	Hypertension	…

HOSPITAL B		
SSN	**Triglyceride**	**…**
13579	172 mg/dL	…
…	…	…
24680	137 mg/dL	…

Figure 2.2: Vertically partitioned data.

Although horizontal and vertical partitioning has pros and cons in different settings, distributed data management alone has major advantages over central data management for increasing performance and preserving privacy, but for this to happen, it requires several security measures to be taken. In the literature, while researchers focus on privacy, they mostly offer solutions on either horizontally or vertically partitioned data. However, many real-life requirements indicate that a data set can be managed both horizontally and vertically in disparate data silos.

2.2 Ensemble Learning

The idea of ensemble learning is to combine the predictions of multiple classifiers (weak classifiers) to produce a single classifier (strong classifier) that is more accurate than any of the individual classifiers constituting the ensemble [19]. The concept of ensemble systems was first introduced by Dasarathy and Sheela in 1979 [20]. Hansen and Salamon implemented one of the earliest ensemble solutions to improve the classification performance of neural networks in 1990 [21]. In a nutshell, an ensemble classifier can be built in three steps [22]. First, data sampling/selection is performed in a way that prevents weak classifiers from producing the same output, thus providing diversity. In the second step, weak classifiers are trained on these data. In the final step, weak classifiers are combined with a strategy such as majority voting to form

the final ensemble classifier.

It has been shown that the probability of an ensemble classifier making an incorrect prediction is usually lower than that of a single classifier. For instance, suppose there are 25 classifiers in the ensemble model, each with an error rate of 0.35 ($\varepsilon = 0.35$). The ensemble classifier makes a wrong prediction if the majority of the classifiers makes a wrong prediction. The probability of 13 or more classifiers making a wrong prediction is approximately 0.06 as calculated in Eq. 2.1, which is much less than 0.35.

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} \approx 0.06 \tag{2.1}$$

For this reason, ensemble techniques have attracted the attention of many researchers. As a result, several ensemble methodologies have been developed, including bagging, boosting, arching, and stacking [23]. These methods have been applied in a variety of research fields, such as time-series forecasting, image segmentation, and classification [24, 25, 26, 27]. The remainder of this section presents the combination strategies used in ensemble algorithms first, followed by the details of bagging and boosting, which are the two most commonly used ensemble techniques in the literature.

2.2.1 Combination Strategies

The final step of any ensemble algorithm consists of a strategy for combining the weak classifiers. Various strategies are available in the literature for different type of problems. Among them, *Averaging* is the simplest yet the most popular combination strategy used in regression problems in which a numerical value is predicted. Given an ensemble classifier C with n weak classifiers such as $\{c_1, c_2, ..., c_n\}$, the final prediction $H(x)$ is calculated as in Eq. 2.2, where $h_i(x)$ is the prediction of c_i on dataset x.

$$H(x) = \frac{1}{n} \times \sum_{i=1}^{n} h_i(x). \tag{2.2}$$

Weighted Averaging or *Weighted Sums* is another important strategy that has been widely used since its first usage in ensemble learning [28]. In this strategy, each weak classifier c_i is given a weight of w_i such that $w_i \geq 0$ and $\sum_{i=1}^{n} w_i = 1$. Then, the final prediction $H(x)$ becomes as in Eq. 2.3.

$$H(x) = \sum_{i=1}^{n} (w_i \times h_i(x)). \tag{2.3}$$

Majority Voting is the most commonly used combination strategy in classification problems where the prediction is class labels rather than continuous values. As its name suggests, in majority voting, the class label that is predicted by the majority of weak classifiers becomes the final prediction. More formally, in classification problems, a weak classifier c_i predicts a class label from a set of m labels $\{l_1, l_2, ..., l_m\}$. Let $d_i^j(x)$ denote the decision of weak classifier c_i for class label l_j on sample x such that if the prediction of c_i on sample x is class label l_j, set $d_i^j(x) = 1$, otherwise $d_i^j(x) = 0$. Let L be an m-dimensional array $\{D_1(x), D_2(x), ..., D_m(x)\}$, where $D_j(x)$ denotes how many times l_j was predicted on sample x by the weak classifiers. Then, $D_j(x)$ and the final prediction $H(x)$ can be formulated as in Eq. 2.4 and 2.5, respectively.

$$D_j(x) = \sum_{i=1}^{n} d_i^j(x) \tag{2.4}$$

$$H(x) = max\{D_1(x), D_2(x), ..., D_m(x)\} \tag{2.5}$$

2.2.2 Bagging

Bagging is an abbreviated term that stands for *Bootstrap Aggregating*. Bootstrapping, or in other words bootstrap resampling, is a methodology of re-sampling from the original dataset many times to create multiple random datasets. Bagging is a bootstrap resampling based ensemble algorithm that takes a number of sub-samples (with or without replacement) from the initial dataset, trains individual predictive models on those sub-samples and obtains the final classifier by averaging the bootstrapped

models, calculating weighted sums or using majority voting [29]. The general idea of Bagging is illustrated in Fig. 2.3.

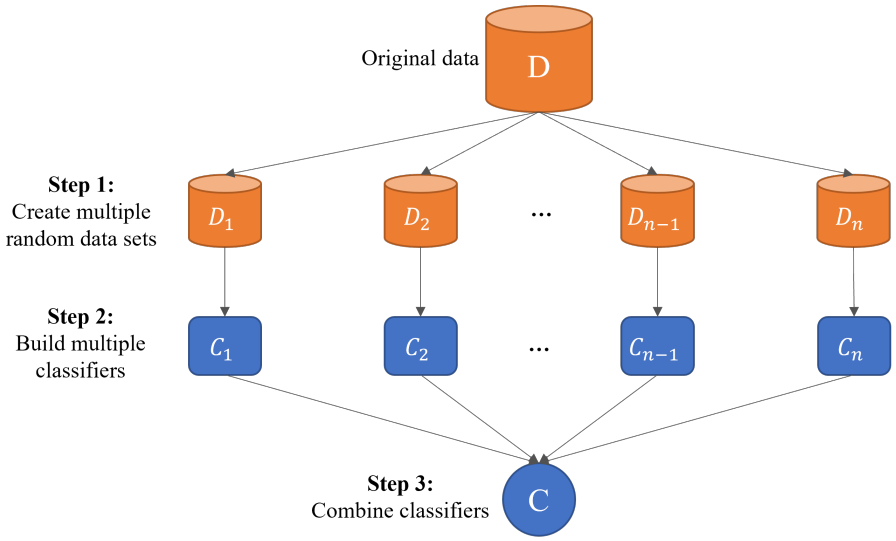

Figure 2.3: Overview of bagging technique.

In Bagging, when taking sub-samples with replacement, a sample is randomly selected from the original dataset and copied to the newly created dataset, but it is kept in the original dataset so that it can still be selected in the next rounds. The models are then trained on these randomly generated datasets and the final result is obtained using one of the combination strategies explained in Section 2.2.1. In this way, bagging helps to reduce variance. Therefore, it is an effective technique especially when there is not enough data in the original dataset.

2.2.2.1 Random Forest

Random Forest (RF) [30] is the best-known bagging algorithm that generates a number of decision trees by not only taking the random subset of data, but also using a random subset of features rather than all, to prevent strong predictors in the dataset from generating highly correlated models. RF can be used for both classification and regression problems.

Given a dataset D containing N data points and M features, the standard Random Forest algorithm executes the following three steps while building each decision tree:

1. A subset of N data points is randomly selected with replacement from the training data,
2. At each node of the decision tree, a random subset of M features is selected,
3. The feature that provides the best splitting among the others is used to split the node.

While aggregating the results of individual decision trees, it uses majority voting approach in classification problems and averaging approach in regression problems. The pseudocode of Random Forest is presented in Alg. 1.

Algorithm 1 Random Forest

Input: Training data $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)\}$,
Feature set $F\{f_1, f_2, ..., f_M\}$,
Number of decision trees T,

Procedure:

$R \leftarrow \{\}$ ▷ *Initialize random forest empty*
for $i \leftarrow 1$ to T **do**
  $D_i \leftarrow BUILD\_DT(D_i, F)$ ▷ *Build a decision tree*
  $R \leftarrow R \cup t_i$ ▷ *Add the decision tree to the forest*
end for

Output: R

Function: $BUILD\_DT(D, F)$

At each node:
  $F^* \leftarrow$ random subset of F ▷ *Select a random subset of M features in F*
  Split on best feature in F^*
return The decision tree

Despite being a simple and easy to implement algorithm, Random Forest is one of the best performing machine learning algorithms as it produces highly accurate results. Unlike the standard Decision Tree algorithm, it decreases variance by using different sub-samples from the original dataset and random subset of features for training, thus reducing the risk of overfitting. It is also not affected by data skewness, hence performs well with imbalanced datasets as well. For these reasons, Random Forest has

been a topic that researchers have studied for many years. In the literature, there are many extensions of Random Forest with different bootstrapping approaches focusing on aspects such as clustered or unbalanced data. Hornung *et al.* [31] developed BlockForest algorithm by modifying the split point selection of decision trees in random forest, which normally uses Information Gain or Gini Index. Hajjem *et al.* [32] proposed Mixed-Effects Random Forest (MERF) algorithm as an extension to Random Forst for clustered data. Field *et al.* [33] analyzed the effects of using different bootstrapping methodologies on clustered data, while Samanta *et al.* [34] performed a similar work for highly imbalanced clustered data.

2.2.3 Boosting

Boosting [35] is another popular ensemble method that is widely used in machine learning applications. In Boosting, the aim is to create a strong classifier from a number of weak classifiers that are trained sequentially by using the information coming from the preceding ones [36]. In both Bagging and Boosting, datasets are generated by taking a number of sub-samples with replacement from the initial dataset. In the case of Bagging, the probability of any data point to appear in a dataset is the same. However, in Boosting, weights are assigned to data points in a way the subsequent model can focus more on the instances that the previous classifier misclassified; therefore, some data points may appear more in datasets, while others may appear less. The comparison of these two approaches in terms of data resampling and model generation is illustrated in Fig. 2.4 and 2.5, respectively. The figure contents were retrieved from [37].

2.2.3.1 AdaBoost

AdaBoost, which is a short for *Adaptive Boosting*, is one of the most popular boosting algorithms owing to its high-performance and effective prediction capability [38, 39]. In AdaBoost, the weak classifiers are decision trees with only one node and two leaves, which is called a stump. In each iteration, a stump is generated, and weights are assigned to both the data points (instances) and weak classifiers based on the

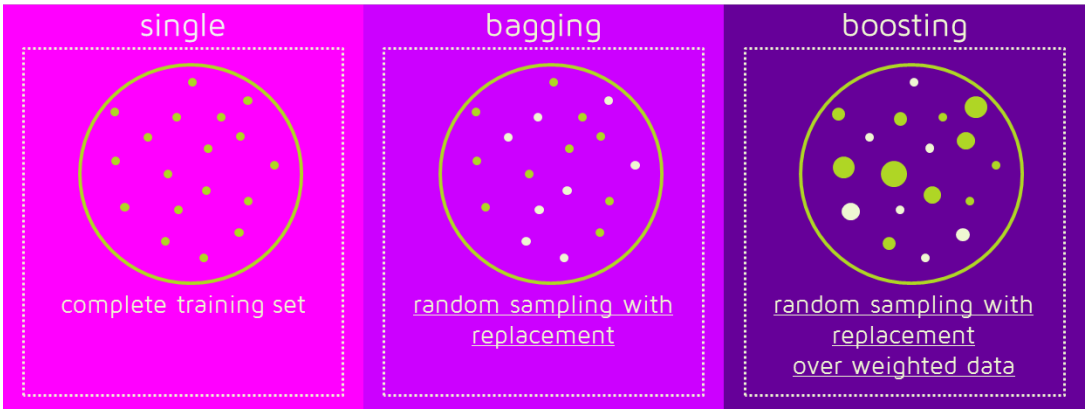

Figure 2.4: Comparison of data resampling in Bagging and Boosting.

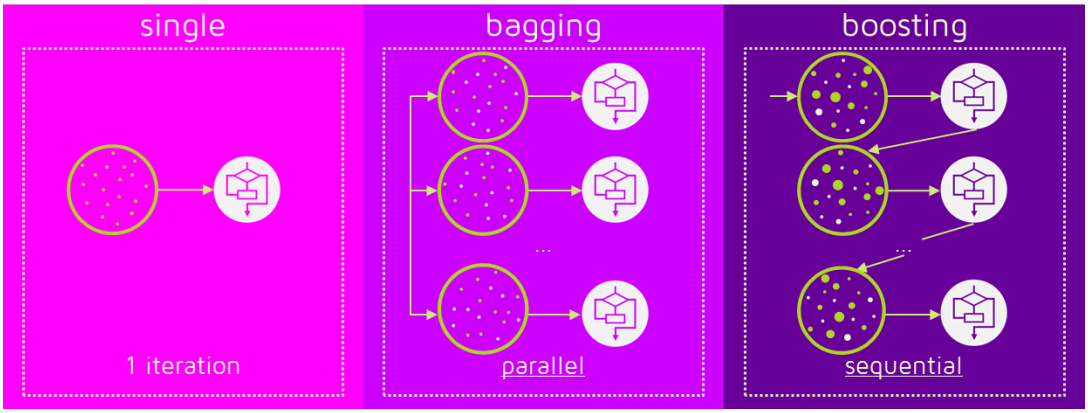

Figure 2.5: Comparison of model generation in Bagging and Boosting.

correctness of the predictions. Having trained a weak classifier, AdaBoost increases the weight of the misclassified instances and decreases the weight of the correctly classified instances so that subsequent classifiers can focus more on the samples on which the previous classifier made an error. The weights are calculated based on the error rate of the preceding weak classifiers.

The pseudocode of the AdaBoost algorithm is presented in Alg. 2. The algorithm starts by initializing the weights. Initially, to all instances are given equal weights (lines 1 & 2). Then, a classifier c_j is fitted by using the current weights W_j on dataset D (line 5). The error ε_j of c_j is computed as a weighted sum of misclassified instances, where I_i represent the classification performance of c_j on instance x_i, such that I_i is set 1 if the classification is incorrect, and 0 if the classification is correct as

shown in line 6. The expression indicating the sum of weights in the denominator is for normalization, which ensures that the sum of the weights in each step is always equal to 1. The classifier weight α_j is then estimated via the formula presented on line 7, which lets $\alpha_j c_j$ minimize the exponential loss function [40]. In the next step, the instance weights are updated (lines 8 & 9). If c_j classifies instance x_i correctly, its weight is decreased by multiplying the current weight by $\exp(-\alpha_j)$, which always returns a number less than 1. On the other hand, if the classification is incorrect, the weight is increased by multiplication with $\exp(\alpha_j)$ which is always greater than 1. This procedure is repeated M times, which is the pre-defined number of classifiers in the ensemble, and consequently, the weighted sum of the weak classifiers constitutes the final ensemble classifier.

Algorithm 2 AdaBoost

Input: Dataset $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\}$,

Weights $W = \{w_1, w_2, ..., w_n\}$,

Number of classifiers in the ensemble M

Training function β,

Procedure:

1: **for** $i \leftarrow 1$ to n **do** ▷ *Initialize weights*

2: $w_i \leftarrow 1/n$ ▷ *Give equal weights to all instances at the beginning*

3: **end for**

4: **for** $j \leftarrow 1$ to M **do**

5: $c_j \leftarrow \beta(D, W_j)$ ▷ *Fit a classifier using the current weights*

6: $\varepsilon_j \leftarrow \dfrac{\sum w_i I_i}{\sum w_i}$ where $I_i = \begin{cases} 1, & \text{if } y_i \neq c_j(x_i) \\ 0, & \text{if } y_i = c_j(x_i) \end{cases}$ ▷ *Estimate the error of* c_j

7: $\alpha_j \leftarrow 0.5 \ln(\dfrac{1-\varepsilon_j}{\varepsilon_j})$ ▷ *Calculate the weight of* c_j

8: **for** $i \leftarrow 1$ to n **do** ▷ *Update the weights of all instances*

9: $w_i \leftarrow w_i \times \begin{cases} \exp(\alpha_j), & \text{if } y_i \neq c_j(x_i) \\ \exp(-\alpha_j), & \text{if } y_i = c_j(x_i) \end{cases}$

10: **end for**

Output: $C(x^*) = sign(\sum_{j=1}^{M} \alpha_j c_j(x^*))$ ▷ *Classify new data*

Gradient-boosted trees (GBT) [41], XGBoost [42], LightGBM [43] and CatBoost [44] are examples of other widely used boosting techniques. GBT follows the same path as AdaBoost in building the ensemble model, but instead of creating a predefined number of weak classifiers (stumps) and using thier weighted sums as output in the final step, it combines the results along the way and repeats the model building process until the result of the error function remains unchanged in the additive model (i.e., until the differentiable loss function converges). It also uses short decision trees instead of stumps. XGBoost is an advanced implementation of GBT through system optimizations and algorithm improvements. It is much faster than GBT as it parallelizes the tree building process, which is sequential in both AdaBoost and GBT. It uses regularization to help reduce overfitting and has built-in cross-validation and missing value handling capability. LightGBM is an extension of GBT that uses "bins" instead of single data points when finding splitting thresholds. It performs faster than others for extremely large datasets.

2.3 Federated Learning

Federated learning (FL) enables multiple sites to build a joint machine learning model by exchanging only information about locally trained models, rather than actual data [8]. In FL, each site first builds a local model using their respective training data. The local models are then either transferred to a third-party coordinator, who is responsible for aggregating local models to build a global model, or exchanged between the sites in a peer-to-peer manner without the involvement of a third-party coordinator. The resulting federated model is expected to perform as well as the ideal model generated by collecting all the data in one place.

An example federated learning architecture with a third-party coordinator is illustrated in Fig. 2.6. In this architecture, the third-party coordinator acts as a global-level aggregator, in other words central aggregation server, that sends the initial model parameters to participating sites (or clients) to start the federated learning process. Each site then trains a local model using its own dataset, and sends model updates to the aggregator. The aggregator combines these values, and sends the aggregated model updates back to the participating sites. This loop is repeated until the model converges

or the maximum number of iterations is reached.

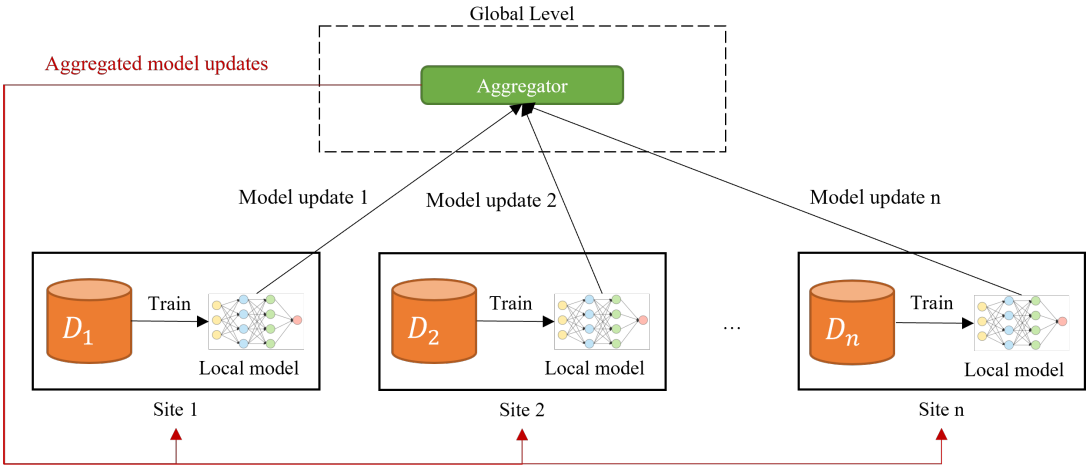

Figure 2.6: An example federated learning architecture with a third-party coordinator.

Federated learning architectures can also be built in peer-to-peer manner without the involvement of a third-party coordinator. An example of such an architecture is shown in Fig. 2.7. In this architecture, the participating sites communicate directly with each other instead of a third-party to exchange model updates, and aggregation is performed locally at each site. Since no third-party is involved at the global level, participants must agree in advance the order in which they send and receive the model updates.

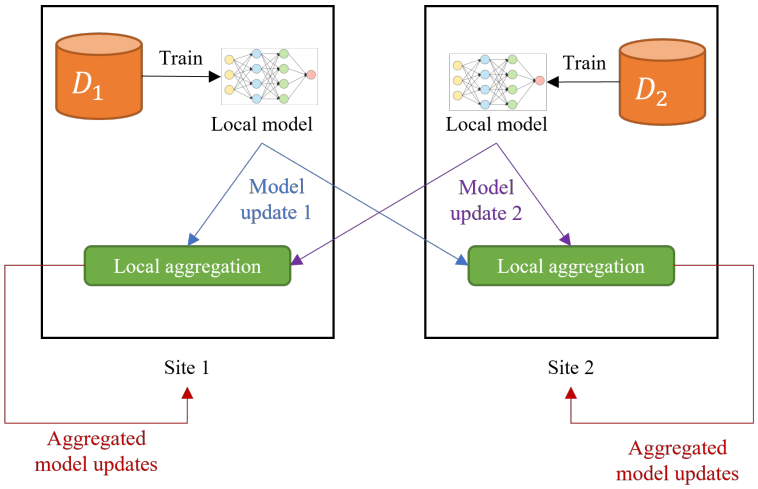

Figure 2.7: An example federated learning architecture in peer-to-peer manner.

Federated learning methodologies can be divided into three categories as Horizontal

Federated Learning (HFL), Vertical Federated Learning (VFL) and Federated Transfer Learning (FTL) based on the type of data partitioning among participants in terms of feature and sample space. HFL refers to the cases where participants share the same set of features, but different sample space; whereas in VFL, participants share the different set of features for the same sample space. In other words, in HFL, datasets are similar to the horizontally partitioned data, while they are similar to vertically partitioned data in VFL, as presented in Section 2.1.2. In FTL, on the other hand, participants share neither the same set of features not the same sample space.

The remainder of this section presents existing studies on HFL and VFL, and applications of FL in various fields. FTL is not presented as it is not in the scope of this study.

2.3.1 Horizontal Federated Learning

In 2016, McMahan *et al.* [10] developed the first federated learning algorithm, namely federated averaging (*FedAvg*) to address the challenge of performing machine learning tasks with datasets having an unbalanced and non-independent identical distributions (non-IID) among huge number of participants with limited and unreliable communication bandtwidth. They proposed an HFL-based framework with a third-party coordinator [45] where Android smartphones are the clients that update model parameters locally and send them to the cloud for aggregation, thus enabling these smartphones to build a joint machine learning model without sharing any data. *FedAvg* is an umbrella term for both *gradient averaging* and *model averaging*. When sending model updates, the participants can send either gradients or local model weights to the aggregator at global-level. In the former, the aggregator performs *gradient averaging* for combining the gradients retrieved from participants, whereas it performs *model averaging* in the latter.

HFL solutions can be implemented either with a third-party coordinator or in peer-to-peer manner. In HFL with a third-party coordinator, the participants are considered honest, i.e., trusted, while the aggregator is considered honest-but-curious [46], i.e., semi-trusted. Therefore, many studies on this topic have focused on preventing privacy leaks by encrypting model updates before sending to the aggregator [47], adding

noise with differential privacy [11, 12], or with secret sharing [48]. In the peer-to-peer architecture, since a third-party coordinator is not involved, privacy leaks are prevented through secure multi-party computation [49] or additively homomorphic encryption [47].

In addition to the studies focusing on privacy issues in federated learning, researchers have also made significant efforts to advance existing studies on reducing communication cost, decreasing computation cost, and increasing model performance, in various fields of machine learning such as classification, regression, association rule learning, and deep learning [50, 51, 52]. Nishio and Yonetani [53] designed a protocol referred to as *FedCS*, which gives participants a limited time to update model parameters and integrates only the updates of participants who could achieve to do so, thereby reducing the overall training time. Wang *et al.* [14] proposed the Federated matched averaging (FedMA) algorithm, which reduces the communication cost in federated neural network architectures by matching and averaging elements in hidden layers with similar feature extraction signatures. Liu *et al.* [54] accelerated the convergence speed via momentum gradient descent as opposed to FL solutions utilizing first-order gradient descent.

This thesis focuses on the application of ensemble strategies in horizontal federated learning to improve the classification performance. In the literature, studies on the application of federated learning for horizontally partitioned data mainly utilize the boosting technique, particularly AdaBoost. MultBoost, Adaboost.PL and BOPPID are three well-known federated implementations of AdaBoost, which are explained in detail below.

2.3.1.1 MultBoost

Gambs *et al.* [55] proposed MultBoost (Multiparty Boosting) algorithm as a distributed implementation of AdaBoost, where the dataset is split between two or more participants, and the model is built in a privacy-preserving manner. The idea of MultBoost is to merge weak classifiers trained by participants in each iteration and compute the weights based on data instances misclassified by the merged weak classifier. In each iteration of MultBoost, the participants first agree on a subset of weak clas-

sifiers that they will use to return their weak classifiers separately by minimizing the weighted error over their respective datasets. The weak classifiers are then merged into a classifier using the majority voting approach at the global level. After obtaining the merged classifier, participants test it on their own datasets and computes the weighted error rates. The error rates are then also combined at the global level and the weight of the merged classifier is calculated. Finally, the global error rate is returned to all participants, and their instance weights are updated accordingly. Although the authors showed that MultBoost can produce a result as good as AdaBoost run on combined datasets, one of the biggest shortcomings of this algorithm is that it requires intense communication between the participants and a central coordinator, which creates a crucial performance issue in a federated environment.

2.3.1.2 AdaboostPL

To reduce the communication cost and make computations in participants independent of each other, Palit *et al.* [17] introduced the AdaBoost.PL algorithm. The pseudocode of Adaboost.PL is presented in Alg. 3. In AdaBoost.PL, participants (or workers as the authors named) first train a local AdaBoost classifier H^p on their own datasets by completing all the iterations of the standard AdaBoost algorithm. H^p is formulated in Eq. 2.6, where $h^{p(i)}$ is the weak classifier (stump) trained at ith step and $\alpha^{p(i)}$ is the respective weight of it.

$$H^p = \{(h^{p(1)}, \alpha^{p(1)}), ..., (h^{p(T)}, \alpha^{p(T)})\} \tag{2.6}$$

The workers then sort the weak classifiers in the local AdaBoost classifier in increasing order with respect to their weights and obtain H^{p^*} as shown in Eq. 2.7, where $(h^{p^*(1)}, \alpha^{p^*(1)})$ is the weak classifier with the smallest weight, whereas $(h^{p^*(T)}, \alpha^{p^*(T)})$ is the weak classifier with the largest weight.

$$H^p = \{(h^{p^*(1)}, \alpha^{p^*(1)}), ..., (h^{p^*(T)}, \alpha^{p^*(T)})\} \tag{2.7}$$

In the next step, the algorithm merges the weak classifiers located at the same index of

each participant's sorted AdaBoost classifiers (line 6) by taking their majority votes. The merged classifier $h^{(t)}$ at tth index can be formulated as in Eq. 2.8.

$$h^{(t)}(x) = \begin{cases} 1, & \text{if } \sum_{p=1}^{M} h^{p^*(t)}(x) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

Its weight α^t is calculated as the average of the weights of the respective weak classifiers as shown in line 7. Finally, the weighted sum of merged classifiers constitute the final classifier.

Algorithm 3 AdaBoost.PL

Input: The training set of M workers, $(D^1_{n^1}, ..., D^M_{n^M})$,

Number of iterations in AdaBoost T,

Procedure:

1: **for** $p \leftarrow 1$ to M **do**
2: $H^p \leftarrow ADABOOST(D^p_{n^p}, T)$ ▷ *Train a local AdaBoost classifier*
3: $H^{p^*} \leftarrow SORT(H^p)$ ▷ *Sort the weak classifiers w.r.t their weights*
4: **end for**
5: **for** $t \leftarrow 1$ to T **do**
6: $h^{(t)} \leftarrow MERGE(h^{1^*(t)}, ..., h^{M^*(t)})$ ▷ *Merge the sorted weak classifiers*
7: $\alpha^t \leftarrow \frac{1}{M} \sum_{p=1}^{M} \alpha^{p^*(t)}$ ▷ *Calculate the weight of merged classifier*
8: **end for**

Output: $H = \sum_{t=1}^{T} \alpha^t h^{(t)}$

2.3.1.3 BOPPID

Li *et al.* [18] implemented the BOPPID (Boosting-based privacy-preserving integration of distributed data) algorithm by following a different methodology while combining the ensemble classifiers generated by the workers. In BOPPID, local classifiers are shared between participants in peer-to-peer manner, instead of being transferred to a central coordinator. Unlike AdaBoost.PL, which merges the weak classifiers of local ensemble classifiers, BOPPID updates their weights based on the following three criteria.

1. The local model of the participant performing the combination process is always assigned more weight compared to other local models with the same sample size.

2. The weights of participants' local model are proportional to their sample size such that the more data the participant has, the more weight its local model is assigned.

3. The weight of the local model of a participant performing the combining process has an upper bound; otherwise, the other models become insignificant.

The pseudocode of BOPPID is outlined in Alg. 4. Let $(D^1_{n^1}, ..., D^M_{n^M})$ denote the datasets of M workers, where $D^p_{n^p}$ is the dataset of pth participant as in Eq. 2.9.

$$D^p_{n^p} = \{(X^p_1, y^p_1), (X^p_2, y^p_2), ..., (X^p_{n^p}, y^p_{n^p})\} \tag{2.9}$$

The BOPPID algorithm starts with training an AdaBoost classifier H^p that contains T weak classifiers in each participant as formulated in Eq. 2.10, where $h^{p(i)}$ is the weak classifier trained at ith iteration and $\alpha^{p(i)}$ is the respective weight of it (line 2).

$$H^p = \{(h^{p(1)}, \alpha^{p(1)}), (h^{p(2)}, \alpha^{p(2)}), ..., (h^{p(T)}, \alpha^{p(T)})\} \tag{2.10}$$

The training error rate ε_p of H^p is computed in line 3, where $I(A)$ is the indicator function that return 0 if A is false and 1 otherwise. Each participant then shares its local AdaBoost classifier and sample size with all other participants, so each participant retrieves $M - 1$ models from the others.

After each participant receives $M - 1$ models from the other participants, they run these models on their own datasets and compute the respective error rates (line 9). Here ε^q_p represents the error rate when qth participant's model runs on the pth participant dataset. If the difference between ε_p and ε^q_p is less than a pre-defined τ threshold value, then the model retrieved from the qth participant is included in the ensemble model created in the pth participant (lines 10 & 11) assuming that these two participants have a similar data distribution.

Algorithm 4 BOPPID (Boosting-based privacy-preserving integration of distributed data)

Input: The training set of M workers, $(D_{n^1}^1, ..., D_{n^M}^M)$,

Number of iterations in AdaBoost T,

Threshold value τ,

Procedure:

for $p \leftarrow 1$ to M **do in parallel**

  $H^p \leftarrow ADABOOST(D_{n^p}^p, T)$ ▷ *Train a local AdaBoost classifier*

  $\varepsilon_p \leftarrow \frac{1}{n^p} \sum_{i=1}^{n^p} I(H^p(X_i^p) \neq y_i^p)$ ▷ *Compute the training error rate*

  Share H^p with all other participants

end for

for $p \leftarrow 1$ to M **do in parallel** ▷ *At each participant*

  $S \leftarrow \{p\}$ ▷ *Initialize the index set of selected participants*

  for $q \leftarrow 1$ to $M-1$ **do** ▷ *For each model received*

    $\varepsilon_p^q \leftarrow \frac{1}{n^p} \sum_{i=1}^{n^p} I(H^q(X_i^p) \neq y_i^p)$ ▷ *Compute the error rate of qth model*

    if $(\varepsilon_p - \varepsilon_p^q) \leq \tau$ **then** ▷ *Select models to be included in the ensemble*

      $S \leftarrow S \cup q$

    end if

  end for

  $\lambda \leftarrow \{\lambda_q : \frac{n^q}{\sum_{s \epsilon S} n^s} | q \epsilon S\}$ ▷ *Instance proportion of the selected participants*

  $\lambda_{min} \leftarrow min(\lambda)$ ▷ *Minimum instance proportion value*

  $\lambda_{max} \leftarrow max(\lambda)$ ▷ *Maximum instance proportion value*

  for each $q \in S$ **do** ▷ *Compute* σ_q^p

    if $p = q$ **then**

      $\sigma_q^p \leftarrow \frac{\lambda_q}{\lambda_{max}^2} \times \lceil \frac{\lambda_{max}^2}{\lambda_{min}} \rceil$

    else

      $\sigma_q^p \leftarrow 1$

    end if

  end for

end for

Output: $H(x) = \sum_{q \epsilon S} \sum_{t=1}^{T} h^{q(t)}(x) \alpha^{q(t)} \sigma_q^p \lambda_q$

In the next step, the algorithm updates the weights of weak classifiers in the selected models, namely the local AdaBoost classifiers, by taking into account the three criteria listed above. In this regard, it calculates the instance proportion λ of each participant whose local model will be included in the integrated model, and find the minimum and maximum values (lines 14-16). Then, a value of σ_q^p is computed using the formula presented in lines 17 to 23 (mathematical basis and proofs of this formula can be found in [18]). The weights of each weak classifier are then multiplied with σ_q^p and λ_q, and consequently the federated ensemble classifier is formed as the weighted sum of the weak classifiers.

In this work, a different ensemble strategy than BOPPID is followed where the weak classifiers were the decision trees in the local random forest models generated at each site. Moreover, a different weight calculation methodology is used when merging the weak classifiers. In the experiments, the results of the proposed algorithm are compared with only BOPPID, as Li *et al.* [18] showed that BOPPID performs better than its competitors, namely MultBoost and AdaBoost.PL.

2.3.2 Vertical Federated Learning

Existing work on VFL in the literature is quite limited when compared to studies on HFL. After Vaidya *et al.* [56] implemented privacy-preserving decision trees over vertical data and Gascón *et al.* [57] performed linear regression on vertically partitioned data, the first VFL solution was developed by Hardy *et al.* [58] utilizing homomorphic encryption to preserve the privacy of data. In recent years, researchers have performed several studies on implementing tree-based algorithms in VFL settings. Cheng *et al.* [59] proposed SecureBoost, a lossless privacy-preserving tree-boosting system for vertically partitioned data. Liu *et al.* [60] implemented a federated forest algorithm in which participants calculate the impurity improvement values for each feature locally, and the central coordinator selects the feature that gives the best split so that a joint random forest can be built. Ge *et al.* [61] improved the solution proposed by Liu *et al.* by optimizing the feature selection and pruning steps to create models with more accurate results.

2.3.3 Applications of Federated Learning

Compared to the traditional centralized machine learning approaches, federated learning provides significant advantages such as protecting data privacy, providing more diverse data, requiring less hardware and saving time. For these reasons, many industries and sectors have incorporated federated learning into their business cycles and developed their FL-based solutions.

Healthcare is one of the largest industries in which FL has received a great deal of attention. The survey studies conducted by Li *et al.* [62], Aledhari *et al.* [63] and Xu *et al.* [64] provide a comprehensive summary of real-life applications of FL in healthcare. Wang and Zhou [65] implemented a federated learning framework, named FedSPL, for early diagnosis of diseases. Choudhury *et al.* [66] predicted adverse drug reactions using real-world health data of 1 million patients in a federated learning framework. Dou *et al.* [67] utilized federated learning for detecting COVID-19 related lung abnormalities. Abdul *et al.* [68] studied detecting COVID-19 on chest x-ray images and showed that federated model gave better prediction accuracy than the local models. Vaid *et al.* [69] predicted mortality in COVID-19 patients within seven days after hospitalization.

In the scope of the FAIR4Health project [70], Alvarez-Romero *et al.* [71] predicted readmission risk of patients with chronic obstructive pulmonary disease 30-days after discharge. Carmona-Pírez *et al.* [72] applied federated association rule algorithm for identifying the most frequent chronic disease combinations and their association with mortality risk.

Federated learning is also being widely explored in the Internet of Things (IoT) field as it helps reduce communication delays and saves network resources [73]. Traffic flow estimation [74], path control for unmanned air vehicles (UAVs) [75], energy prediction in smart cities [76], industry 4.0 WiFi networks [77] and mobile edge networks [78] are some examples among the numerous federated learning studies in the IoT field.

2.4 Personalized Federated Learning

In federated learning, the main aim is to build a common global model that produces better results than each participant's own local models. However, when the heterogeneity of the data distribution (e.g., non-IID) among the participants increases, the global model might not generalize well on the datasets of all participants, hence might not yield the expected good results when run on the local dataset of some participants [79]. To overcome this problem, various studies have been made by researchers to *personalize* the global model for each participant and as a result, the concept of Personalized Federated Learning (PFL) has emerged.

Based on the approach they use to improve the performance of the global model in each participant, existing PFL solutions in the literature can be grouped into several categories, including but not limited to local fine-tuning, multi-task learning, model-mixing and regularization [80, 81].

Among them, local fine-tuning is the most common approach, where each participant tunes the parameters of the global model locally on their respective datasets. Fallah *et al.* [82] studied the convergence of Model-Agnostic Meta-Learning (MAML) [83] methods for the federated learning systems and implemented the personalized variant of the FedAvg algorithm, named Per-FedAvg, with convergence guarantees. Wang *et al.* [84] and Arivazhagan *et al.* [85] worked on personalized training of federated neural networks and proposed solutions where only the parameters in the last layer are updated through stochastic gradient descent (SGD) for each participant. Although local fine-tuning can be an effective approach to address the heterogeneity problem among participants in FL, it is very prone to overfit.

Smith *et al.* [86] developed a federated multi-task learning framework, named MOCHA, considering optimization of each participant as a separate task. Hanzely *et al.* [87] and Deng *et al.* [81] proposed solutions mixing participant's local model with the global model. Dinh *et al.* [88] and Huang *et al.* [89] accomplished personalization using regularization between the global and local models. Regularization methods are generally easier to implement as they only require modifying the FedAvg algorithm slightly [90].

2.5 Clustered Federated Learning

Clustered Federated Learning (CFL) is a fairly new concept that has emerged to address the challenges that arise when the heterogeneity of data distribution (e.g., non-IID) among the participants increases. In CFL, the aim is to group multiple participants in a way that they build federated models that are more specialized than the global model but provide better accuracy for the participants in that group.

The first CFL solution was proposed by Sattler *et al.* [91], in which they clustered participants by using the cosine similarity between their weight updates. Inspired by their work, Duan *et al.* [92] proposed another method, called the euclidean distance of Decomposed Cosine similarity (EDC), for clustering the participants. Briggs *et al.* [93] introduced hierarchical clustering, in which participants are clustered based on the similarity of gradient updates in their local models. Yu *et al.* [94] clustered participants based on their data distribution and built federated model for each cluster separately.

In these approaches, a central coordinator is present to identify cluster identities of all the participants, which brings additional computational cost. To reduce this cost, Ghosh *et al.* [95] implemented the Iterative Federated Clustering Algorithm (IFCA), aiming to minimize the participants' loss values while estimating their cluster identities without a central machine.

CHAPTER 3

BOOSTING-BASED FEDERATED RANDOM FOREST (BOFRF) ALGORITHM

In this chapter, first, the base Boosting-based Federated Random Forest (BOFRF) algorithm is described and its mathematical formulation is given. Second, the computational complexity of BOFRF is analyzed. Third, the personalized implementation of BOFRF is described, and then its complexity analysis is presented. Lastly, the hyperparameters and how hyperparameter tuning can be performed on BOFRF are explained. Description of the notations used in this chapter while explaining the details of algorithm is shown in Table 3.1. The details of the base BOFRF algorithm and its mathematical formulation have also been published in [96].

Table 3.1: Notations used in the thesis while explaining the details of algorithm.

Notation	Description
N	Number of sites
D_n	Dataset of the nth site
m_n	Number of instances at the nth site
X	Vector of feature values
y	Label values $\in \{+1, -1\}$
S_n	Training set of the nth site
T_n	Test set of the nth site
R_n	The random forest classifier trained at the nth site
k_n	Number of decision trees at the nth site

Continued on next page

Table 3.1: Notations used in the thesis while explaining the details of algorithm. (Continued)

Notation	Description
d_i^n	The decision tree at the ith index of the nth site
α_i^n	The weight of decision tree at the ith index of the nth site
$c_i^n(q)$	The confusion matrix calculated by the qth site for decision tree at the ith index of the nth site
C_i^n	The final confusion matrix for decision tree at the ith index of the nth site
tp	True positive, i.e., the number of positive examples predicted positive
tn	True negative, i.e., the number of negative examples predicted negative
fp	False positive, i.e., the number of negative examples predicted positive
fn	False negative, i.e., the number of positive examples predicted negative
$M(C_i^n)$	The Matthews correlation coefficient value of confusion matrix C_i^n
τ	The threshold value
F_n	The local ensemble classifier at the nth site
F	The final global ensemble classifier
$\alpha_i^{n^*}$	The "initial" weight of decision tree at the ith index of the nth site
F_n^*	The "initial" local ensemble classifier at the nth site
F^*	The "initial" global ensemble classifier
Z	Number of threshold values
$\alpha_i^n(\tau_z)$	The weight of decision tree at the ith index of the nth site based on the threshold value τ_z
$F_n(\tau_z)$	The local ensemble classifier at the nth site based on the threshold value τ_z
$F(\tau_z)$	The final personalized global ensemble classifier based on the threshold value τ_z

3.1 The Base Algorithm

The federated environment consists of several sites, each of which contains datasets with both common and local features. Given N sites, the dataset of the nth site containing m_n instances can be represented as

$$D_n = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_n}, y_{m_n})\} \tag{3.1}$$

where X_i is the vector of the feature values and $y_i \in \{+1, -1\}$ is the label used for binary classification. The datasets at each site are split into two sets: training set S_n and test set T_n, where $S_n \cup T_n = D_n$.

In the base BOFRF algorithm, first, a random forest model is trained on the training set, S, of each site. A random forest model comprises several decision trees, each of which is built by using a random subset of features at each split within the tree algorithm as explained in Sec. 2.2.2.1. The random forest classifier having k_n decision trees trained at the nth site is represented as

$$R_n = \{d_1^n, d_2^n, \ldots, d_{k_n}^n\} \tag{3.2}$$

where d_i^n is the ith decision tree generated in the random forest. In the standard random forest algorithm, after the model is built, each decision tree makes a prediction on the test set T, and the label predicted by the majority of decision trees constitutes the final prediction. In BOFRF, however, a boosting methodology is applied in a federated manner to calculate the weight α_i^n for each decision tree[1] so that their combination will constitute the final ensemble classifier. For this purpose, first, each decision tree d_i^n of classifier R_n is run on the training set S_n. Then, for each d_i^n the confusion matrix c_i^n is calculated which is a two-dimensional array of the number of true positive, true negative, false positive and false negative predictions, such that

$$c_i^n(n) = (tp, tn, fp, fn). \tag{3.3}$$

[1] In the proposed solution, decision trees are the weak classifiers of the final ensemble classifier. In this thesis, both terms are used interchangeably; however, they always refer to the same thing.

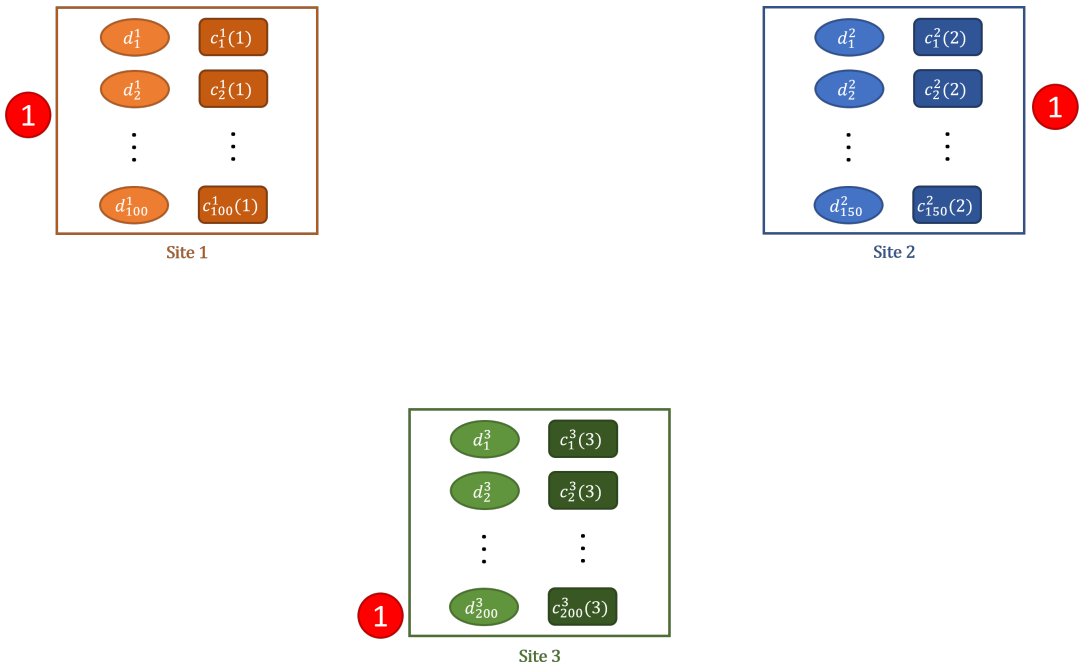

Figure 3.1: The first step of BOFRF in an example federated setup in which 3 sites participated.

Fig. 3.1 illustrates the first step of the algorithm in an example federated setup in which three sites participated. In the figure, each ellipse within a site corresponds to a decision tree, whereas the rounded rectangle next to it represents the corresponding confusion matrix. The order of the steps is shown by circles with numbers in this figure and in the figures shown hereinafter.

In the second step, all sites share their own decision trees, in other words weak classifiers, with every other site, hence the qth site retrieves $((\sum_{i=1}^{N} k_i) - k_q)$ number of decision trees as shown in Fig. 3.2. Each decision tree is then run on the site's training set S_q, and the corresponding confusion matrix is calculated. The confusion matrix calculated by the qth site for the decision tree d_i^n retrieved from the nth site can be represented as

$$c_i^n(q) = (tp, tn, fp, fn). \tag{3.4}$$

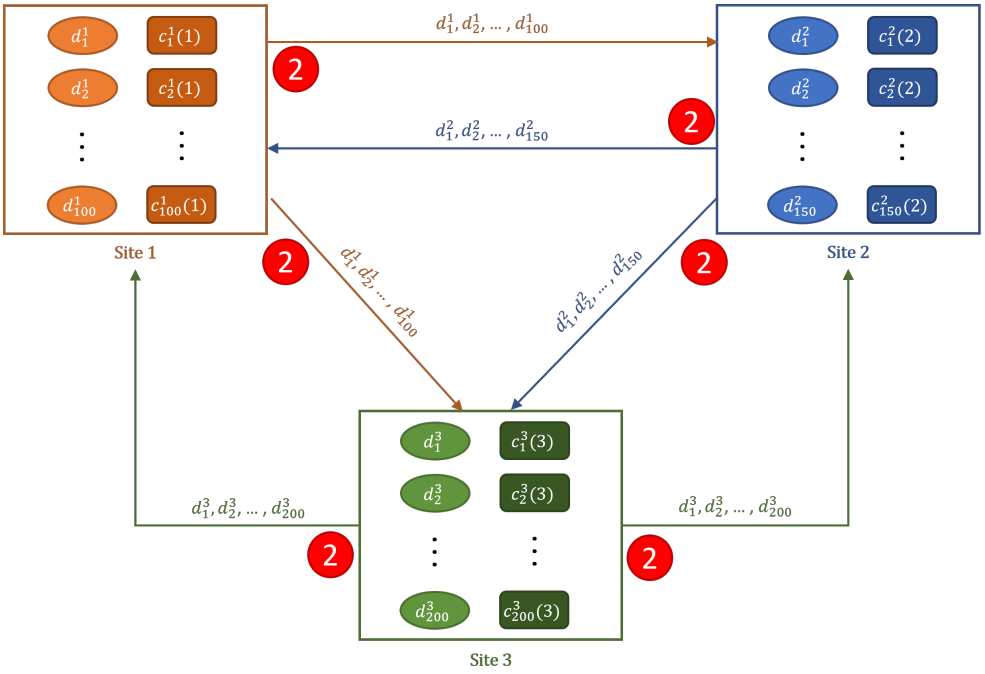

Figure 3.2: The second step of BOFRF in an example federated setup in which 3 sites participated.

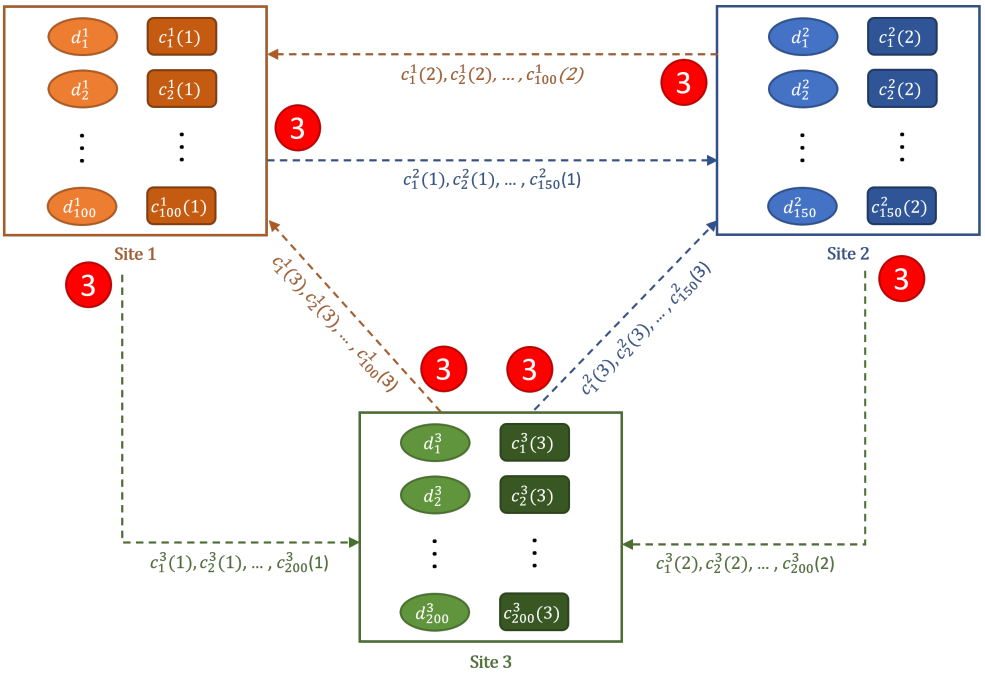

Figure 3.3: The third step of BOFRF in an example federated setup in which 3 sites participated.

In the third step, each site sends the confusion matrices generated in the second step to the sites from which the corresponding decision tree was received. The final confusion matrix for a decision tree d_i^n is generated by adding up true positives with true positives, true negatives with true negatives, and so on, as in

$$C_i^n = \sum_{j=1}^{N} c_i^n(j). \tag{3.5}$$

Fig. 3.3 illustrates the third step of the algorithm, while Fig. 3.4 shows the first three steps of the algorithm together in an example federated setup in which three sites participated. In the figures, the arrows indicate the communication between sites, where straight lines show decision trees sent and dashed lines show confusion matrices retrieved as a response. Note that although all straight lines are part of step 2 and all dashed lines are part of step 3, these are shown in only one place for visual convenience in Fig. 3.4.

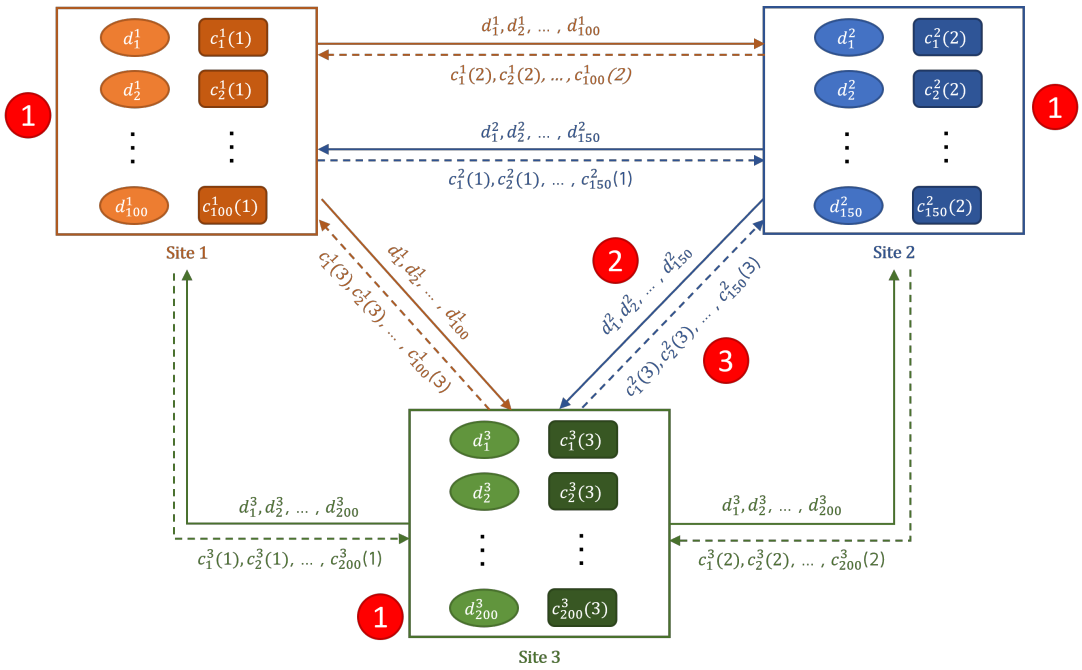

Figure 3.4: The first three steps of BOFRF in an example federated setup in which 3 sites participated.

Once the final confusion matrices are calculated, the next step is to assign weights to each decision tree. Most ensemble methods utilize the accuracy or error rate, ratio of

misclassified instances, and total number of instances to calculate weights for weak classifiers. However, this is not always the best solution, especially when one of the sites has imbalanced data. Assume that a site contains 98% of the data labeled as positive and 2% of the data labeled as negative. If a learning model that always classifies all instances as positive is generated, then an accuracy of 98% is obtained. This model appears to be one of the best models that can be generated; however, it is useless. This problem is known as the "accuracy paradox" in the literature [97]. Instead, other performance metrics can be utilized such as area under curve (AUC), precision, recall, and F-score, depending on the type of problem being solved. Among them, the AUC is the most widely used performance metric, which can produce good results for both balanced and imbalanced datasets. However, the success of the AUC decreases with an increase in the imbalance or skewness in the dataset. In such cases, precision and recall would evaluate the model's performance better [98]. Precision is a useful metric for cases in which the number of false positives should be minimized, whereas recall, which is the accuracy on positive label, aims to minimize the number of false negatives. In order to benefit from both precision and recall at the same time, the F1-score was introduced as the harmonic mean of the two metrics. Although it is an effective metric for imbalanced datasets, it cannot be used as a generic weight calculation metric in a federated environment either, because it does not consider true negative predictions; hence, it is not able to detect true negative rates. In this work, first, accuracy and F1-score were used to calculate the weights of each decision tree, but it was observed that neither the federated model generated with accuracy nor the one generated with the F1-score always outperformed the local models. Looking deeper into the results, it was noticed that when a decision tree that predicts negative values well is combined with a decision tree that predicts positive values well, one degrades the performance of the other because of the aforementioned reasons.

The Matthews correlation coefficient (MCC), or phi coefficient, is a statistical measure used to discover the association between two binary variables [99]. It was adapted to the machine learning domain by Baldi *et al.* [100] in 2000 as a performance metric to show the correlation between predictions and real values. MCC considers all values in the confusion matrix; hence, it is not affected by imbalanced or skewed datasets [101]. It has been shown that it is more robust and reliable than

the aforecited performance metrics when evaluating the classifier performance in both balanced and imbalanced cases [102]. Therefore, in the proposed methodology, the MCC is utilized to calculate the classifier weights. The MCC is calculated as

$$MCC = \frac{(tp \times tn) - (fp \times fn)}{\sqrt{(tp + fp) \times (tp + fn) \times (tn + fp) \times (tn + fn)}}. \tag{3.6}$$

The values of MCC range between [-1, +1], where +1 indicates perfect classifier, 0 indicates random guessing classifier, while -1 indicates completely opposite classifier which predicts everything wrongly. Therefore, the proposed algortihm is only interested in classifiers having positive MCC value. If all instances in the dataset belongs to only one label, either positive or negative, or if the classifier predicts everything as positive or negative, both the nominator and denominator in MCC become zero, which is undefined. In such cases, the MCC can be set directly to zero and the classifier is ignored. Because the randomness of the classifier increases as the MCC converges to zero, it was observed that removing weak classifiers having an MCC value below a certain threshold τ increases the performance of the final ensemble classifier. In the experiments, the optimal result was initially obtained with threshold $\tau = 0.2$, but it was then discovered that tuning the threshold parameter could produce better results for each site. Details of this are explained in Sec. 3.3. Based on these, the weight α_i^n of decision tree d_i^n having a final confusion matrix C_i^n is calculated as

$$\alpha_i^n = \begin{cases} M(C_i^n) & \text{if } M(C_i^n) > \tau \\ 0 & otherwise \end{cases} \tag{3.7}$$

where $M(C_i^n)$ is the MCC value calculated from confusion matrix C_i^n. Illustration of the weight calculation, which is the fourth step of the algorithm, is presented in Fig. 3.5.

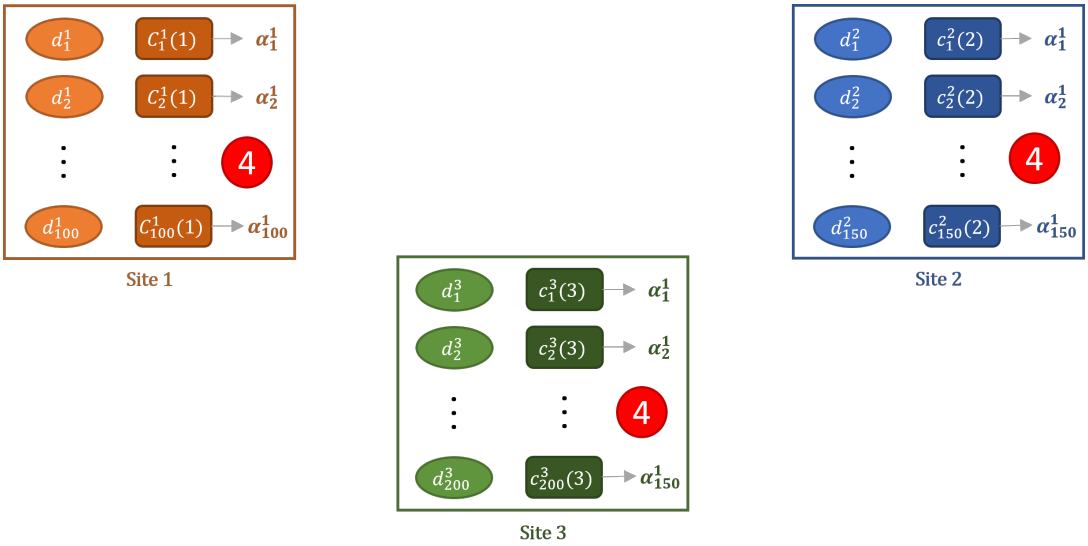

Figure 3.5: The fourth step of BOFRF in an example federated setup in which 3 sites participated.

After the weights of decision trees are calculated, the local ensemble classifier F_n at the nth site is generated as

$$F_n = (\alpha_1^n \times d_1^n) + (\alpha_2^n \times d_2^n) + ... + (\alpha_{k_n}^n \times d_{k_n}^n). \tag{3.8}$$

Finally, the linear combination of local ensemble classifiers, that is, all decision trees with their corresponding weights, constitutes the final global ensemble classifier $F = F_1 + F_2 + ... + F_n$, which can also be represented as

$$F = (\alpha_1^1 \times d_1^1) + ... + (\alpha_{k_1}^1 \times d_{k_1}^1) + (\alpha_1^2 \times d_1^2) + ...+ \\ (\alpha_{k_2}^2 \times d_{k_2}^2) + (\alpha_1^n \times d_1^n) + ... + (\alpha_{k_n}^n \times d_{k_n}^n). \tag{3.9}$$

The final steps of the algorithm are illustrated in Fig. 3.6. In the figure, the rectangle on top represents the federated model generated as a combination of weighted decision trees coming from local RF models. In general, F can be formulated as

$$F = \sum_{n=1}^{N} \sum_{i=1}^{k_n} \alpha_i^n \times d_i^n. \tag{3.10}$$

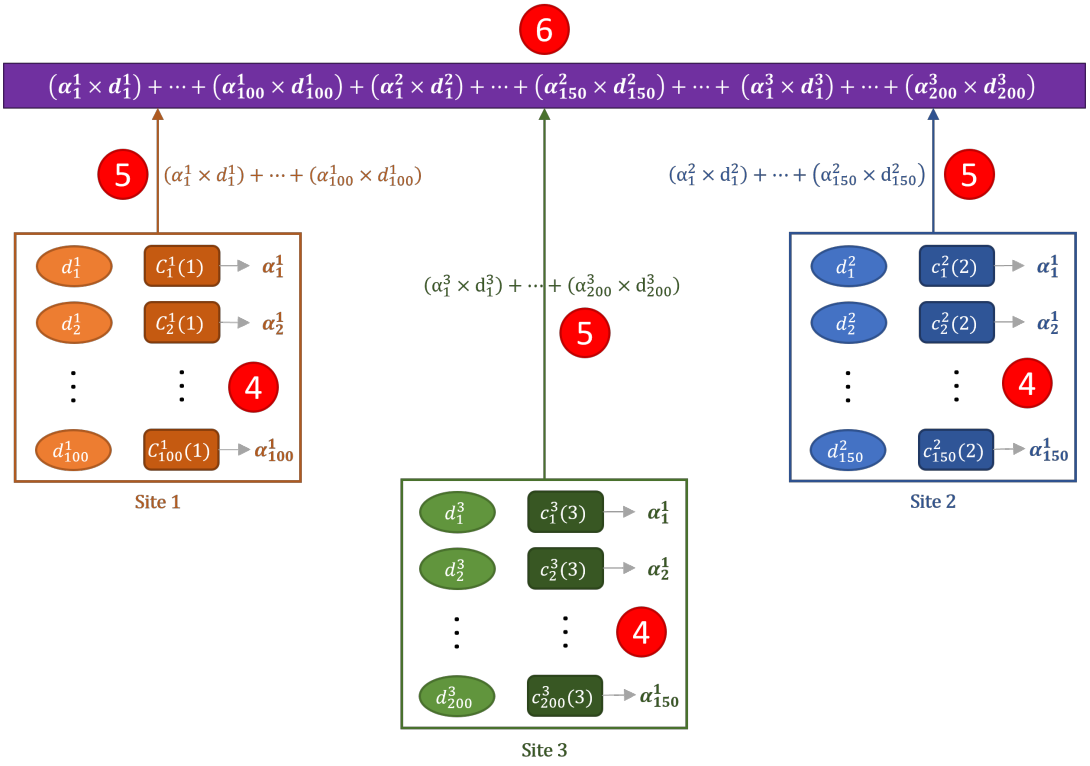

Figure 3.6: The weight calculation and aggregation steps of BOFRF in an example federated setup in which 3 sites participated.

The pseudocode for the proposed Boosting-based Federated Random Forest (BOFRF) algorithm is shown in Alg. 5. For the sake of simplicity, the first and second steps of the algorithm are provided as separate functions in Alg. 6 and Alg. 7, respectively. In the algorithm, first, each site trains a standard random forest model on its training set and calculates the values of the confusion matrix (line 2). The output random forest models, which contain a number of decision trees, are then sent to every other site (line 4). In lines 6-10, each site calculates the statistics for all decision trees of other sites on its own training data and sends back the results. The confusion matrices are then merged in line 13, and the corresponding weight is calculated in lines 14-19. In line 23, the weighted decision trees are combined, and the final ensemble classifier is built.

Algorithm 5 The Base Boosting-based Federated Random Forest (BOFRF)

Input: Dataset of N sites $\{D_1, D_2, \ldots, D_N\}$ where

$D_i = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_i}, y_{m_i})\}$,

Output: The final global ensemble classifier F

Procedure:

for $n \leftarrow 1$ to N **do in parallel**
  $R_n, C^n(n) \leftarrow TRAIN\_LOCAL\_RF(D_n, m_n, k_n)$ ▷ *Step 1, see Alg. 6*
  end for
  Send R_n to every other site ▷ *Step 2*
end for
for $q \leftarrow 1$ to N **do in parallel**
  $R^{'} = \{\{R_1, R_2, ..., R_N\} - \{R_q\}\}$
  $\{C^1(q), C^2(q), ..., C^n(q)\} \leftarrow RUN\_DTS(D_q, R^{'})$ ▷ *Step 2, see Alg. 7*
  Send all $c_i^n(q) \in C^n(q)$ for d_i^n to the nth site which is the owner ▷ *Step 3*
end for
for $n \leftarrow 1$ to N **do in parallel**
  for $i \leftarrow 1$ to k_n **do** ▷ *Step 4: Merge CMs and calculate weight*
    $C_i^n \leftarrow \sum_{j=1}^{N} c_i^n(j)$ ▷ *Eq. 3.5*
    $w \leftarrow M(C_i^n)$ ▷ *Eq. 3.6*
    if $(w > \tau)$ **then** ▷ *Eq. 3.7*
      $\alpha_i^n \leftarrow w$
    else
      $\alpha_i^n \leftarrow 0$
    end if
  end for
  $F_n \leftarrow \sum_{i=1}^{k_n} \alpha_i^n \times d_i^n$ ▷ *Step 5: Build the local ensemble model (Eq. 3.8)*
end for
return $F = \sum_{n=1}^{N} \sum_{i=1}^{k_n} \alpha_i^n \times d_i^n$ ▷ *Step 6: Build the global model (Eq. 3.9)*

Algorithm 6 Training local Random Forest model and calculating confusion matrices at the nth site (Step 1 of BOFRF)

Input: Dataset $D_n = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_n}, y_{m_n})\}$,

Number of instances m_n,

Number of decision trees k_n,

Function: $TRAIN\_LOCAL\_RF(D_n, m_n, k_n)$

$S_n \leftarrow$ Training set after D_n is split

$T_n \leftarrow$ Test set after D_n is split

$R_n \leftarrow RANDOM\_FOREST(S_n, k_n)$ ▷ *Alg. 1*

$C^n(n) \leftarrow \{\}$ ▷ *Initialize confusion matrix list empty*

for $i \leftarrow 1$ to k_n **do**

  $y_{pred} \leftarrow PREDICT(d_i, S_n)$

  $c_i^n(n) \leftarrow CONFUSION\_MATRIX(y_{pred}, S_n)$ ▷ *Eq. 3.3*

  $C^n(n) \leftarrow C^n(n) \cup c_i^n(n)$ ▷ *Add the confusion matrix to the list*

end for

return $R_n = \{d_1, d_2, ..., d_{k_n}\}$ and $C^n = \{c_1^n(n), c_2^n(n), ..., c_{k_n}^n(n)\}$

Algorithm 7 Running each decision tree retrieved from other sites at the qth site (Step 2 of BOFRF)

Input: Dataset $D_q = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_q}, y_{m_q})\}$,

Random forests of other sites $R^{'} = \{\{R_1, R_2, ..., R_N\} - \{R_q\}\}$,

Function: $RUN\_DTS(D_q, R^{'})$

1: $S_q \leftarrow$ Training set after D_q is split
2: $T_q \leftarrow$ Test set after D_q is split
3: $C^{'}(q) \leftarrow \{\}$ ▷ *Confusion matrices of every other site*
4: **for** R_n in $R^{'}$ **do**
5: $C^n(q) \leftarrow \{\}$ ▷ *Confusion matrices for the nth site*
6: **for** $i \leftarrow 1$ to k_n **do**
7: $y_{pred} \leftarrow PREDICT(d_i^n, S_q)$ ▷ $d_i^n \in R_n$
8: $c_i^n(q) \leftarrow CONFUSION\_MATRIX(y_{pred}, S_q)$ ▷ *Eq. 3.4*
9: $C^n(q) \leftarrow C^n(q) \cup c_i^n(q)$ ▷ *Add the confusion matrix to the list*
10: **end for**
11: $C^{'}(q) \leftarrow C^{'}(q) \cup C^n(q)$
12: **end for**
13: **return** $C^{'}(q) = \{C^1(q), C^2(q), ..., C^n(q)\}$ where $n \neq q$

As a result, using a federated ensemble classifier F, the ensemble prediction becomes the sign of F for a given vector of the feature values X. An important point that should be highlighted here is that unlike traditional weak classifiers, which return either +1 or -1, the weak classifiers in BOFRF might also return 0 as a way of "abstaining" from answering [103]. In BOFRF, a weak classifier (decision tree) predicts 0 if any of its nodes contains a feature which does not appear in the feature space of the site where the prediction was made (line 6 in Alg. 6 and line 7 in Alg. 7). In such cases, related sites do not contribute to updating the confusion matrix of the corresponding weak classifier.

Last, but not least, after the federated ensemble classifier F is generated, each site makes predictions on its test set T_n by using both F and F_n, and compares the results to check which one is better. If F is better than F_n, the nth site uses F for future predictions; otherwise, it uses F_n as formulated in Eq. 3.11. Thus, it is always

ensured that the final model performs at least as well as the local model for each site.

$$\begin{cases} F & \text{if } AUC(F) > AUC(F_n) \\ F_n & otherwise \end{cases} \tag{3.11}$$

3.2 Complexity Analysis of BOFRF

In this section, the computational complexity of BOFRF is analyzed by going through each step of the algorithm. The first step of BOFRF is to build a standard random forest model containing k_n decision trees. The cost of building a decision tree at the nth site is $O(f_n m_n \log(m_n))$, where f_n is the number of features in the dataset (i.e., the length of X_n), m_n is the number of instances, and $\log(m_n)$ is the depth of the tree in the worst-case scenario. Thus, the complexity of Random Forest becomes $O(k_n f_n m_n \log(m_n))$. It requires one pass through the training set for each decision tree to make a prediction and generate the confusion matrix; hence, the cost is $O(k_n m_n)$. As a result, the computational complexity of the first step (lines 1-5 in Alg. 5) is $O(k_n f_n m_n \log(m_n) + k_n m_n)$, which can be asymptotically rewritten as $O(k_n f_n m_n \log(m_n))$.

In the second step (lines 6-10 in Alg. 5), decision trees are shared between sites and confusion matrices are generated. From a complexity point of view, the only difference between steps 1 and 2 is the increase in the number of decision trees; hence, the cost is $O(K m_n)$, where K is the number of trees retrieved from other sites.

In the last step (lines 11-22 in Alg. 5), for each decision tree, the algorithm iterates through the N number of confusion matrices to calculate weight. Hence, the computational complexity is $O(k_n N)$.

Since all the three steps are performed in parallel at different sites, the complexity depends on the maximum number of decision trees, features, and instances among all the participating sites. Consequently, the computational complexity of BOFRF be-

comes $O(k'f'm'\log(m') + K'm' + k'N)$, where k',f',m',and K' represent the maximum values. Since $K' \leq k' * N$, because $k' * N$ also represents the total number of decision trees at all sites, the computational complexity of BOFRF becomes $O(k'f'm'\log(m') + k'm'N)$, which can be simplified to $O(k'm'(f'\log(m') + N))$.

3.3 Personalized BOFRF

In the base BOFRF algorithm, weak classifiers having an MCC value below a fixed threshold value are removed in the integration step to improve the prediction performance of the global federated ensemble classifier. That is, a single global federated ensemble classifier is built by using a single threshold value for each participant. In the experiments that were carried out within the scope of this study, however, it was discovered that instead of producing a single global federated model with a single fixed threshold value, different participants could have federated models that produce better results than the global model by personalizing the global model using different threshold values. On the other hand, it was also observed that in some cases the global ensemble classifier could not outperform the local ensemble classifier in some participants although different threshold values were tried while building federated models of respective participants. As a result of these two observations, the base BOFRF algorithm was improved with a personalized implementation using the local fine-tuning approach, which is the most commonly used personalized federated learning approach as explained in Sec. 2.4.

In personalized BOFRF, each participant locally fine-tunes two hyper-parameters, which are named *threshold* and *ensemble strategy*, to come up with a better-performing federated model on their own datasets. Unlike the existing personalized FL solutions which tune model parameters at each communication round of federated model generation, in personalized BOFRF, fine-tuning is performed only in the integration step. Therefore, the first four steps of the base BOFRF algorithm, which can be regarded as the "FL training" step altogether, remain unchanged (lines 1-13 in Alg. 5). After calculating an MCC value based on the final confusion matrix for each decision tree, the value is set directly as the "initial" weight of the respective decision tree rather than using the conditional expression shown in Eq. 3.7, where the result is compared

to the threshold. In personalized BOFRF, the "initial" weight $\alpha_i^{n^*}$ of decision tree d_i^n having a final confusion matrix C_i^n is calculated as

$$\alpha_i^{n^*} = M(C_i^n) \tag{3.12}$$

where $M(C_i^n)$ is the MCC value calculated from confusion matrix C_i^n. Then, the "initial" local ensemble classifier F_n^* at the nth site is generated as

$$F_n^* = (\alpha_1^{n^*} \times d_1^n) + (\alpha_2^{n^*} \times d_2^n) + ... + (\alpha_{k_n}^{n^*} \times d_{k_n}^n). \tag{3.13}$$

In the base BOFRF algorithm, after the local ensemble classifiers are built at each site, the linear combination of them constitutes the final global ensemble classifier $F = F_1 + F_2 + ... + F_n$. In personalized BOFRF, however, first, an "initial" global ensemble classifier is built as $F^* = F_1^* + F_2^* + ... + F_n^*$ and shared with all the sites, and then the "local adaptation" step begins. In this step, given a set of threshold values $\{\tau_1, \tau_2, ..., \tau_z\}$, where Z is the number of different threshold values provided for hyper-parameter tuning, each site builds local and federated ensemble classifiers for each τ_z by keeping only the decision trees whose initial weight is above the threshold value τ_z and updating the others to zero, such that

$$\alpha_i^n(\tau_z) = \begin{cases} \alpha_i^{n^*} & \text{if } \alpha_i^{n^*} > \tau_z \\ 0 & otherwise \end{cases} \tag{3.14}$$

Then, the local ensemble classifier $F_n(\tau_z)$ and the federated ensemble classifier $F(\tau_z)$ that are built with the threshold value τ_z can be formulated as Eq. 3.15 and Eq. 3.16, respectively.

$$F_n(\tau_z) = (\alpha_1^n(\tau_z) \times d_1^n) + (\alpha_2^n(\tau_z) \times d_2^n) + ... + (\alpha_{k_n}^n(\tau_z) \times d_{k_n}^n). \tag{3.15}$$

$$F(\tau_z) = \sum_{n=1}^{N} \sum_{i=1}^{k_n} \alpha_i^n(\tau_z) \times d_i^n. \tag{3.16}$$

In the last step, out of the Z number of federated ensemble classifiers, the one that outperforms the others becomes the final federated ensemble classifier for that site.

Similarly, the local ensemble classifier that produce better results than the other $Z-1$ local ensemble classifiers becomes the final local ensemble classifier. Eq. 3.17 and Eq. 3.18 formulates the last step of personalized BOFRF. The pseudocode of personalized BOFRF algorithm is presented in Alg. 8.

$$F_n = argmax\{F_n(\tau_1), F_n(\tau_2), ..., F_n(\tau_z)\}. \tag{3.17}$$

$$F = argmax\{F(\tau_1), F(\tau_2), ..., F(\tau_z)\}. \tag{3.18}$$

Algorithm 8 Personalized Boosting-based Federated Random Forest (BOFRF)

Input: Dataset of N sites $\{D_1, D_2, \ldots, D_N\}$ where

$D_i = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_i}, y_{m_i})\}$,

Set of threshold values $\tau^{'} = \{\tau_1, \tau_2, ..., \tau_z\}$.

Output: The final global ensemble classifier F

Procedure:

for $n \leftarrow 1$ to N **do in parallel**
  $R_n, C^n(n) \leftarrow TRAIN\_LOCAL\_RF(D_n, m_n, k_n)$ ▷ *Step 1, see Alg. 6*
  end for
  Send R_n to every other site ▷ *Step 2*
end for
for $q \leftarrow 1$ to N **do in parallel**
  $R^{'} = \{\{R_1, R_2, ..., R_N\} - \{R_q\}\}$
  $\{C^1(q), C^2(q), ..., C^n(q)\} \leftarrow RUN\_DTS(D_q, R^{'})$ ▷ *Step 2, see Alg. 7*
  Send all $c_i^n(q) \in C^n(q)$ for d_i^n to the nth site which is the owner ▷ *Step 3*
end for
for $n \leftarrow 1$ to N **do in parallel**
  for $i \leftarrow 1$ to k_n **do** ▷ *Step 4*
    $C_i^n \leftarrow \sum_{j=1}^{N} c_i^n(j)$ ▷ *Merge CMs (Eq. 3.5)*
    $\alpha_i^{n^*} \leftarrow M(C_i^n)$ ▷ *Calculate the initial weight (Eq. 3.6, 3.12)*
  end for
  $F_n^* \leftarrow \sum_{i=1}^{k_n} \alpha_i^{n^*} \times d_i^n$ ▷ *Step 5: Build the initial local model (Eq. 3.13)*
end for

18: $F^* = \sum_{n=1}^{N} \sum_{i=1}^{k_n} \alpha_i^{n^*} \times d_i^n$ ▷ *Step 6: Build the initial global model*

19: **for** $n \leftarrow 1$ to N **do in parallel**

20: $F_n^{'} \leftarrow \{\}$ ▷ *List of local ensemble classifiers for the nth site*

21: $F^{'} \leftarrow \{\}$ ▷ *List of federated ensemble classifiers for the nth site*

22: **for** $z \leftarrow 1$ to Z **do** ▷ *Step 7: Try different threshold values*

23: **for** $j \leftarrow 1$ to N **do**

24: **for** $i \leftarrow 1$ to k_j **do**

25: **if** $(\alpha_i^{j^*} > \tau_z)$ **then** ▷ *Eq. 3.14*

26: $\alpha_i^j(\tau_z) \leftarrow \alpha_i^{j^*}$

27: **else**

28: $\alpha_i^j(\tau_z) \leftarrow 0$

29: **end if**

30: **end for**

31: **end for**

32: $F_n(\tau_z) \leftarrow \sum_{i=1}^{k_n} \alpha_i^n(\tau_z) \times d_i^n$ ▷ *Eq. 3.15*

33: $F(\tau_z) = \sum_{j=1}^{N} \sum_{i=1}^{k_n} \alpha_i^j(\tau_z) \times d_i^j$ ▷ *Eq. 3.16*

34: **end for**

35: $F_n^{'} \leftarrow F_n^{'} \cup F_n(\tau_z)$ ▷ *Add local ensemble classifier to the list*

36: $F^{'} \leftarrow F^{'} \cup F(\tau_z)$ ▷ *Add federated ensemble classifier to the list*

37: **end for**

38: **end for**

39: $F_n = argmax\{F_n(\tau_1), F_n(\tau_2), ..., F_n(\tau_z)\}$ ▷ *Eq. 3.17*

40: $F = argmax\{F(\tau_1), F(\tau_2), ..., F(\tau_z)\}$ ▷ *Eq. 3.18*

3.4 Complexity Analysis of Personalized BOFRF

In this section, the computational complexity of personalized BOFRF is analyzed. Since the first four steps of the base BOFRF algorithm are the same in personalized BOFRF, the complexity remains the same, thus calculation can be directly copied from Sec. 3.2, which is $O(k_n f_n m_n \log(m_n)) + O(K m_n)$ (lines 1-10).

In the next step (lines 11-27 in Alg. 8), for each decision tree, the algorithm iter-

ates through the N number of confusion matrices to calculate "initial" weights. The computational complexity of this step is $O(k_n N)$.

In the last step (lines 19-37), for each threshold value τ_z in $\{\tau_1, \tau_2, ..., \tau_z\}$, the algorithm iterates through Nk_n number of weighted decision trees to compare their initial weights to the threshold value τ_z. Then, it iterates through Z number of local ensemble classifier and Z number of federated ensemble classifier to find the best-performing models. Therefore, the computational complexity of the last step becomes $O(ZNk_n) + O(Z) + O(Z)$.

Since all the steps of personalized BOFRF are performed in parallel at different sites as in the base BOFRF, the complexity depends on the maximum number of decision trees, features, and instances among all the participating sites as well. Consequently, the computational complexity of BOFRF becomes $O(k'f'm' \log(m') + K'm' + k'N + ZNk' + Z + Z)$, where k',f',m',and K' represent the maximum values. Since $K' \leq k' * N$, because $k' * N$ also represents the total number of decision trees at all sites, the computational complexity of personalized BOFRF becomes $O(k'f'm' \log(m') + k'm'N + k'ZN)$, which can be simplified to $O(k'(m'(f' \log(m') + N) + ZN)$.

3.5 Hyperparameter Tuning

In BOFRF, hyperparameter tuning can also be performed in the first step of the algorithm, where local Random Forest models consisting of a number of decision trees are built at each participating site. The Random Forest hyperparameters that can be tuned in BOFRF are as follows.

- ***criterion*:** The measure that is used to find the best splitting feature at each node of the decision tree like *Gini Index (Impurity)* and *Information Gain*.
  - *Gini Index (Impurity)* is a criterion to minimize the probability of misclassification. If a dataset D contains a set of instances from N classes, the Gini index G is calculated as $G(D) = 1 - \sum_{i=1}^{N} p_i$, where p_i is the relative frequency (probability) of class i in D. For each feature, $G_{split}(D)$ is calculated as in Eq. 3.19, where D_1 and D_2 are two subsets with sizes N_1 and

N_2 after D is split with the related feature. Then, the feature providing the smallest $G_{split}(D)$ is chosen to split the node.

$$G_{split}(D) = \frac{N_1}{N} G(D_1) + \frac{N_2}{N} G(D_2) \tag{3.19}$$

– *Information Gain* is a commonly used splitting criterion which is based on a purity measure called "Entropy". Entropy is a probabilistic measure that tells how random a dataset is. For each feature, information gain I is calculated as the decrease in entropy after node is split. Then, the node is split on the feature that provides the largest I.

$$Entropy = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{3.20}$$

$$I = Entropy_{\text{before split}} - Entropy_{\text{after split}} \tag{3.21}$$

- ***min_samples_split*:** Minimum number of instances that must be in an internal node to be able to split it.
- ***max_features*:** Maximum number of features in a node when looking for the best split. $sqrt$ and $log2$ are the most common values for this parameter, which gives the square root or log of the number of features in the datasets. A specific number can also be provided.
- ***max_depth*:** Maximum allowed depth for a decision tree in the forest.
- ***min_samples_leaf*:** Minimum number of instances that must be in a leaf node after splitting a node.
- ***n_estimators*:** The number of decision trees in the forest.

CHAPTER 4

PRIVACY-PRESERVING IMPLEMENTATIONS OF BOFRF

4.1 Privacy Issues

The proposed BOFRF algorithm provides an adequate level of privacy in its current form as the actual data is not shared among the participants. However, sharing evaluation metrics in the presence of a sneaky site in the federated environment may still result in a privacy breach. For instance, if the sneaky site knows the characteristics of a particular patient, it can generate two models, one of which labels the individual as Class 0, and the other as Class 1. Then, by comparing the confusion matrices from other sites to find exactly 1 difference, the sneaky site can find out which site the patient belongs to. To prevent this from happening and increase the level of privacy, in this chapter, two different implementations of BOFRF are presented, which are centralized implementation with a trusted third party and decentralized implementation using secure sum protocol.

4.2 Centralized Implementation with a Trusted Third Party

A *trusted third party* (TTP) is an entity in a distributed/federated architecture that all parties trust to perform a particular service [104]. The concept of TTP has been utilized in many domains such as cryptography and distributed data mining. Certificate Authority (CA) is a good example of a TTP in cryptography that issues and signs digital certificates and is trusted both by the certificate owner and the party using the certificate [105]. Gurevich *et al.* [106] explains the idea of TTP in distributed data mining with a simple example illustrated in Fig. 4.1. The architecture in the example

scenario consists of a set of databases, a miner, and a calculator. The role of the miner is to decide which computation to be done, while the calculator's role is to make the calculation without knowing anything about the given or published information. The result of the computation is known only to the miner and participant databases. In this scenario, the miner and calculator are the trusted third parties.

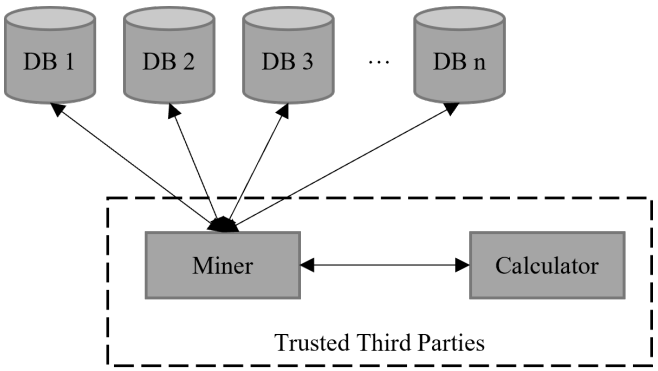

Figure 4.1: Trusted Third Parties in distributed data mining.

In the centralized implementation of BOFRF, an *Orchestrator* is introduced as the trusted third party, who is responsible for sending the decision trees to the other sites, retrieving output confusion matrices, and calculating the final confusion matrix for each decision tree without knowing anything about the information provided by the sites. It acts like an *Aggregator* at global level as previously explained in Fig. 2.6 in Sec. 2.3. In the first step of the algorithm in this implementation, after a random forest model consisting of a number of decision trees is trained and the confusion matrices are calculated for each decision tree at each site, the decision trees and the corresponding confusion matrices are sent to the *Orchestrator* as illustrated in Fig. 4.2. Then, in the second step, the *Orchestrator* sends to each site the decision trees retrieved from other sites as shown in Fig. 4.3. In the third step, each site runs the decision trees retrieved from the *Orchestrator* on their respective datasets and sends the confusion matrices back to the *Orchestrator* (Fig. 4.4).

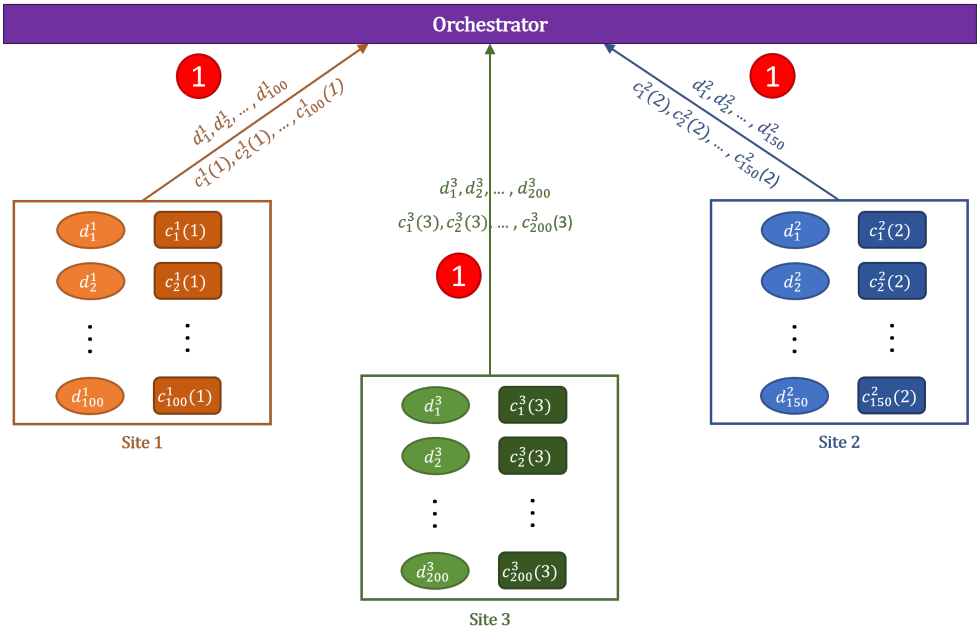

Figure 4.2: The first step of the centralized implementation of BOFRF with a trusted third party.

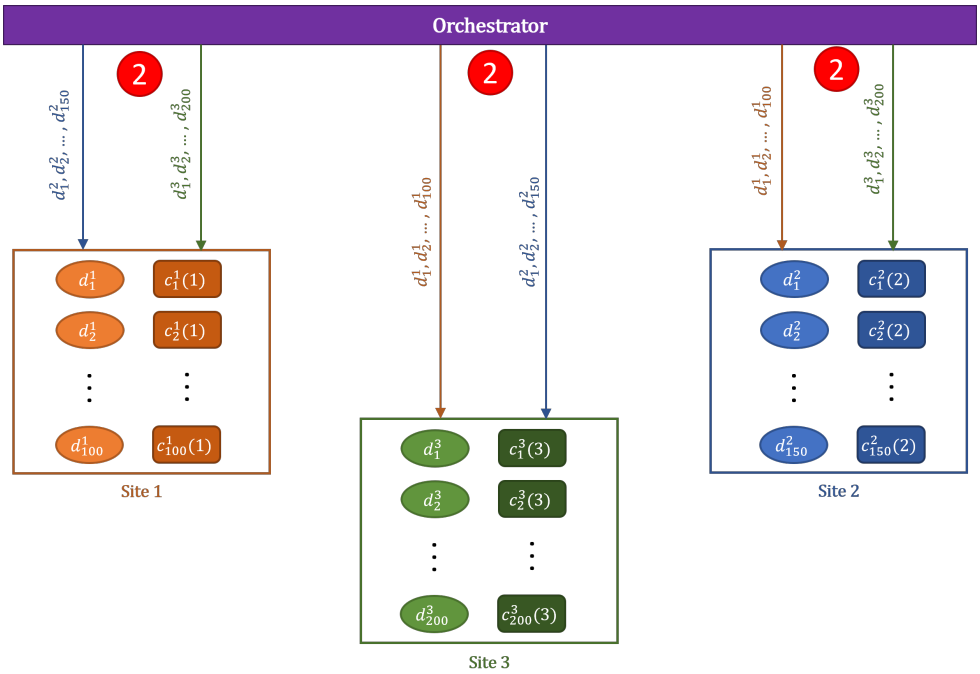

Figure 4.3: The second step of the centralized implementation of BOFRF with a trusted third party.

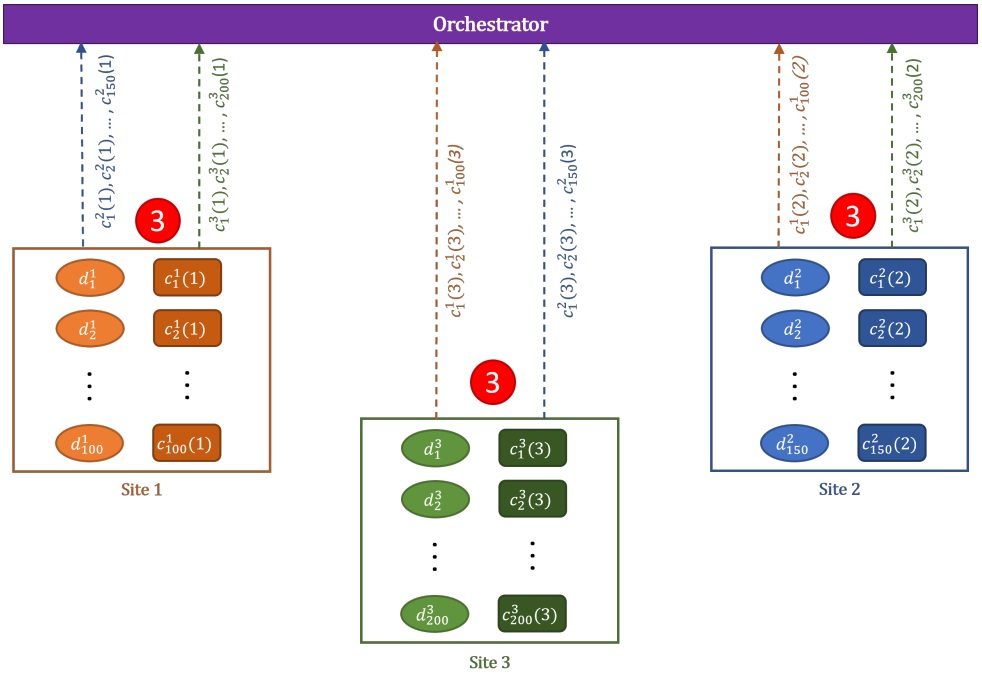

Figure 4.4: The third step of the centralized implementation of BOFRF with a trusted third party.

Once the confusion matrices are received from all sites, the *Orchestrator* starts the aggregation process as depicted in Fig. 4.5. In this process, it first combines all the confusion matrices generated by the participating sites for a decision tree and calculates its final confusion matrix. Then, based on the final confusion matrices, it computes a weight for each decision tree as shown by the red circles numbered 4 in in Fig. 4.5. In the fifth and sixth steps, final local ensemble classifiers are created as a weighted sum of decision trees and their linear combination constitutes the final federated ensemble classifier.

The centralized implementation with a trusted third party can also be applied for personalized BOFRF. In this case, instead of computing the weights of each decision tree and constructing a global federated ensemble classifier, the *Orchestrator* simply calculates the final confusion matrices for each decision tree and sends the decision trees and their confusion matrices to each participating site. The sites then perform the rest of the steps locally with hyperparameter tuning. In either case, participating sites never see the confusion matrices of other sites; hence, privacy is protected against

certain types of attacks like patient-site assignment attack.

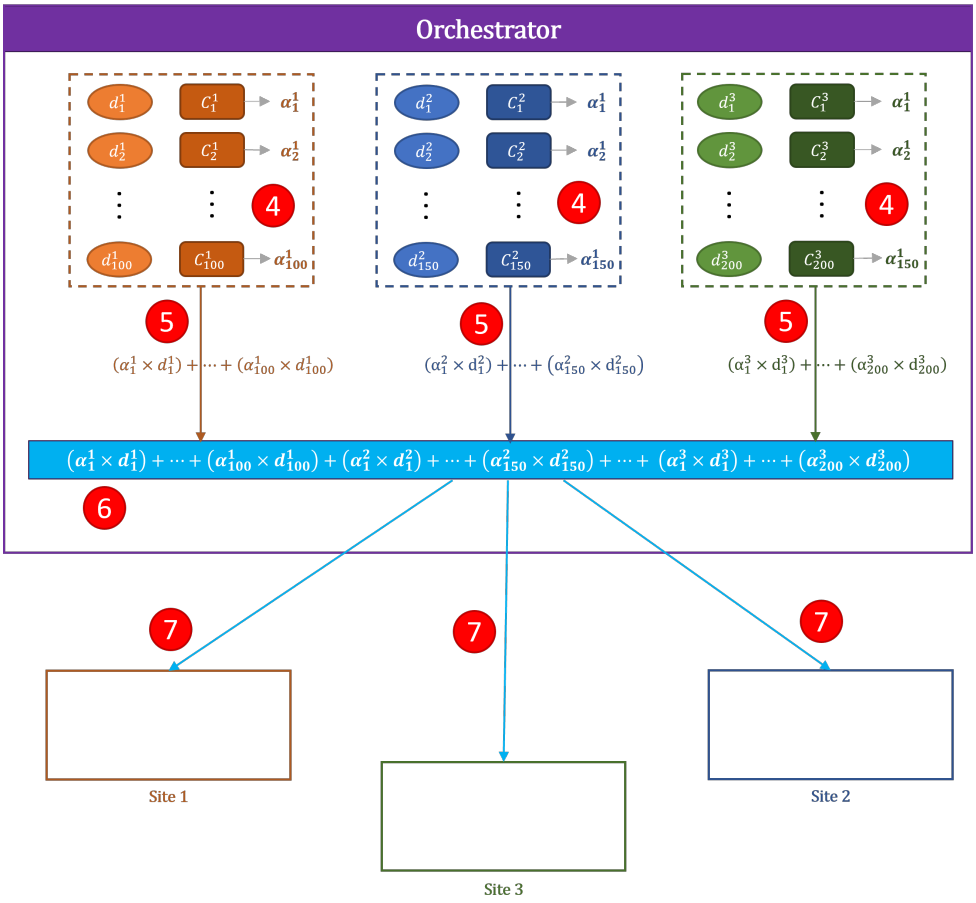

Figure 4.5: The aggregation procedure in the centralized implementation of BOFRF with a trusted third party.

4.3 Decentralized Implementation Using Secure Sum Protocol

In decentralized implementation, instead of communicating with a third party, participating sites communicate with each other in a circular way. This approach uses the secure sum protocol, which allows participating sites to calculate the sum of their individual data without exposing their data to other sites [107]. Secure sum is a simple and efficient protocol which can compute the sum in a secure way for three or more participants in the following way. In the first step of the protocol, one of the sites acts as the initiator. It adds a random noise to its own data and sends it to the next site. The receiving site adds the incoming noisy data to its own data and sends it to the

next site. This process is repeated until the circle is complete and the site that started the protocol retrieves the sum. The site then subtracts the noise that is known only to itself from the sum and finds out the actual sum.

More formally, secure sum can be formulated as follows. Let $\{P_1, P_2, \ldots, P_n\}$ denote the set of participants, v_i denote the value that ith participant holds as its local output, and P_1 be the initiator of the protocol. P_1 generates a random value R existing in the range $[0..r]$, computes $R_1 = R + v_1 \pmod r$ and sends R_1 to P_2. When P_2 receives this value, it cannot learn anything about the value of v_1, because R_1 is some uniformly distributed number in the range $[0..r]$. P_2 and all other participants continue the protocol by calculating their outputs with the formula $R_i = R_{i-1} + v_i \pmod r$. When the last participant P_n computes its sum, it sends the output R_n to the first participant P_1. What P_1 receives is

$$R_n = R + \sum_{i=1}^{n} v_i \pmod r. \tag{4.1}$$

In the final step of the protocol, since P_1 knows the value of R, it can subtract R from R_n and obtain the final result. Fig. 4.6 depicts how secure sum protocol works for three participants.

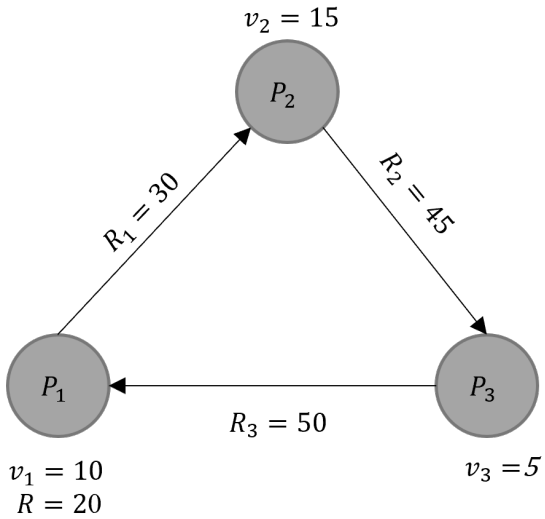

Figure 4.6: Example of a Secure Sum protocol with three participants.

In the decentralized implementation of BOFRF, each site executes the secure sum protocol separately for each confusion matrix. Therefore, they all act as an initiator of the protocol for their own decision trees. In the first step of this implementation, all the sites add a random noise to their local confusion matrices and send them to the next site along with their local decision trees as depicted in Fig. 4.7. The noise added to the confusion matrices by the ith site is represented as P_i. The noisy confusion matrix U_i^j of the jth decision tree at the ith site is calculated as

$$U_i^j = ((tp(c_i^j) + P_i), (tn(c_i^j) + P_i), (fp(c_i^j) + P_i), (fn(c_i^j) + P_i)). \tag{4.2}$$

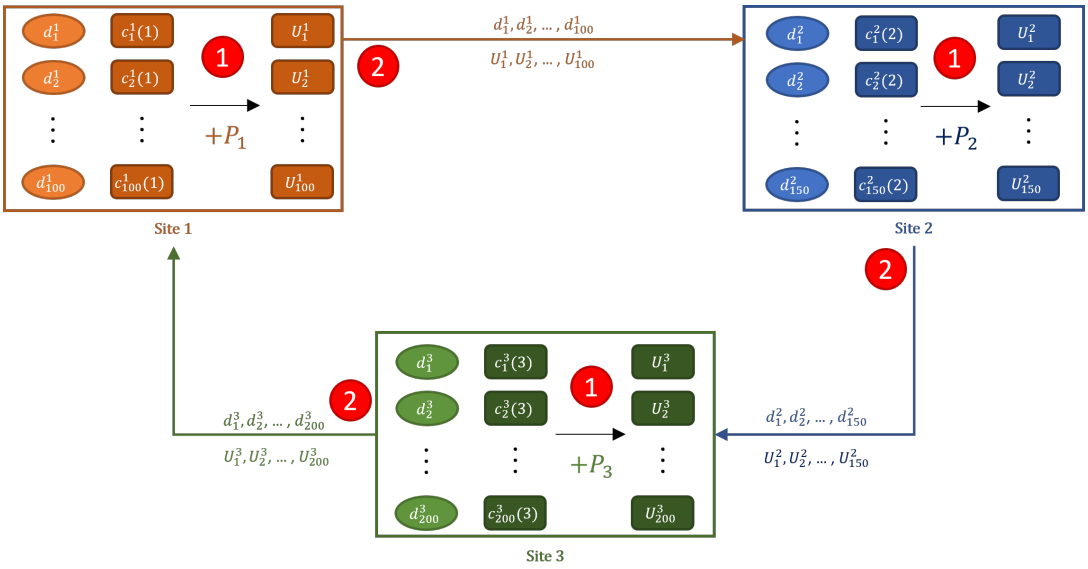

Figure 4.7: The first and second steps of the decentralized implementation of BOFRF using secure sum protocol.

In the third step, the recipient sites run the retrieved decision trees on its own data, calculates the respective confusion matrices, and adds these values to the retrieved confusion matrices. Afterwards, they send the decision trees and confusion matrices to the next site. The process continues until the circular cycle is completed as illustrated in Fig. 4.8 and Fig. 4.9.

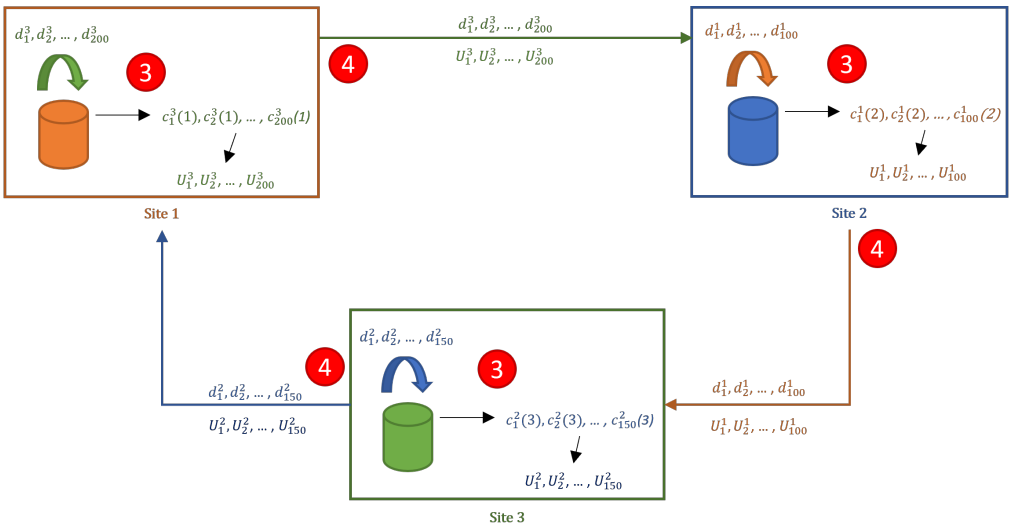

Figure 4.8: The third and forth steps of the decentralized implementation of BOFRF using secure sum protocol.

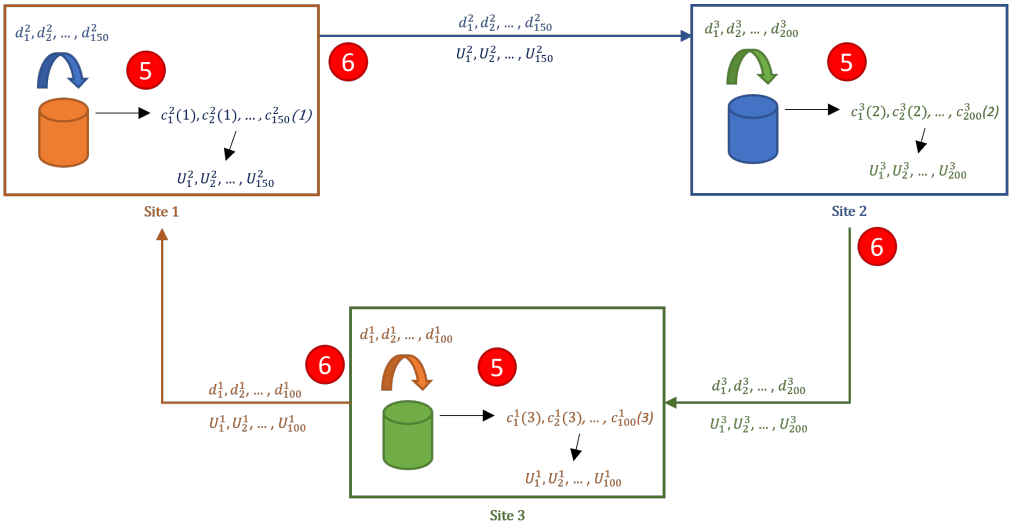

Figure 4.9: The fifth and sixth steps of the decentralized implementation of BOFRF using secure sum protocol.

After each site gets back the decision trees they initially created and their confusion matrices updated by the other sites, they subtract the noise they added to confusion matrices at the first step and obtain the final confusion matrices. The final confusion matrix C_i^j of the jth decision tree at the ith site is calculated as

$$C_i^j = ((tp(U_i^j) - P_i), (tn(U_i^j) - P_i), (fp(U_i^j) - P_i), (fn(U_i^j) - P_i)). \quad (4.3)$$

In this implementation, the participating sites see some values in confusion matrices, but they never know the actual values; hence, privacy is preserved. The algorithm then continues with the usual weight calculation and final model generation steps. The final steps of the decentralized implementation of BOFRF is depicted in Fig. 4.10.

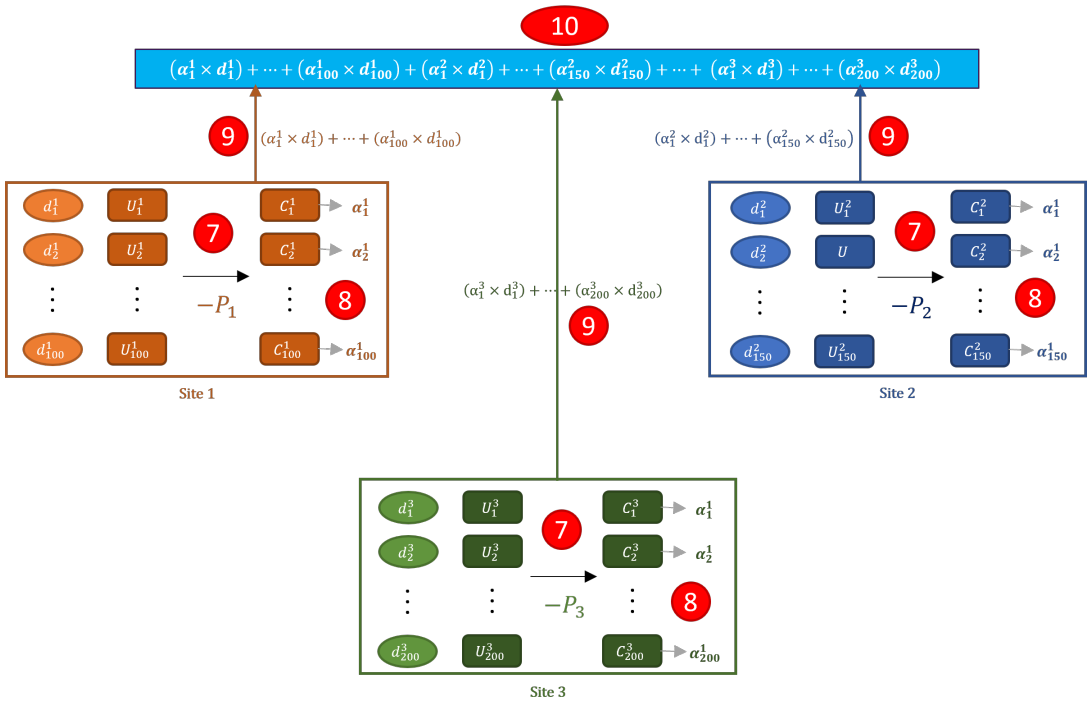

Figure 4.10: The final steps of the decentralized implementation of BOFRF using secure sum protocol.

4.4 Remarks

Although both implementations increase the level of privacy, it should be noted that there is a trade-off between the level of privacy and complexity. Centralized implementation increases the communication cost as all information exchange is done through a trusted third party, whereas decentralized implementation increases the time cost as execution is done sequentially rather than parallel.

CHAPTER 5

CLUSTERED BOFRF

The main purpose of federated learning is to enable different participants to come together and create a common strong model without sharing their private data with each other. However, when local models of some participants are poor or due to their bad quality or imbalanced data, it can be difficult to produce successful federated models. The BOFRF algorithm described in this work has been proposed to address this problem. BOFRF not only increases the predictive power of all participating sites, but also provides significantly high improvements on the predictive power of sites having unsuccessful local models. However, the effectiveness of the federated model produced with BOFRF depends on the data distribution of the sites participating in the federated environment. This can be considered from two perspectives. First, the increase in data distribution difference among participants means that weak classifiers having different characteristics are combined to create a federated model. In this case, if a participant is already satisfied with the predictive performance of its local model at a certain level and the goal is only to slightly improve it, then the federated BOFRF model built with weak classifiers of different characteristics may not provide the desired improvement. On the other hand, this kind of a federated model could be perfect for a participant whose local model is already poor. From the second perspective, when there is a correlated data distribution among participants, a participant who already has a well-performing local model and seeks only a minor improvement in that model may not achieve the desired improvement as the federated model will consist of weak classifiers with similar characteristics. Likewise, if a participant with a poor local model builds a federated model with participants having a data distribution similar to its own, the federated model may not provide an improvement at all. Furthermore, when the number of sites participating in the federated setting increases,

it will take longer to build a federated model with BOFRF, especially in decentralized architecture, and bring more computational costs.

To address these problems, in this chapter, a clustering-based extension to the BOFRF algorithm is proposed. A pre-step to the original BOFRF is introduced, where the participating sites form clusters based on their data distribution prior to running the BOFRF algorithm. The federated model is then created only with sites within that cluster. The remainder of this chapter is organized as follows. First, Sec. 5.1 explains the idea of clustering and how the most popular clustering algorithm, *K-means*, works. Then, the definition of the characteristic vector is given and the complete clustered BOFRF algorithm is presented in Sec. 5.2. Finally, the computational complexity of the clustering procedure is analyzed in Sec. 5.3. Description of the notations used in this chapter in addition to those given in 3.1 is shown in Table 5.1.

Table 5.1: Additional notations used in clustered BOFRF.

Notation	Description
K	Number of clusters & centroids
c_i	ith centroid
C	List of centroids such that $C = \{c_1, c_2, ..., c_K\}$
U_j	jth cluster
R	Number of features
f_r	rth feature of the dataset
V_n	Characteristic vector of the nth site
$S(f_r)$	The function for calculating data distribution statistics for the f_r column
$norm(f_r)$	The normalization function that normalizes the data in the f_r column to 0-1 range
$\gamma(f_r, X_i)$	The function returning the value in rth column and ith row in the dataset
$d(p, q)$	Distance function for calculating the distance between two points p and q
E_n	nth site

5.1 Clustering Approach

Clustering is an unsupervised learning technique for identifying some segments or clusters in a dataset without requiring any target variables. *K-means* is the most popular clustering algorithm aiming to group n observations into K clusters. Given a dataset $D = \{x_1, x_2, ..., x_n\}$, and the number of clusters K, the algorithm works in four steps:

1. Randomly pick at K points as the initial centroids, such that $C = \{c_1, c_2, ..., c_K\}$.
2. For each data point x_i, find out the nearest centroid using a distance metric like Eucledian distance, which is the default distance metric of *K-means*. So, for each x_i, calculate the distance between x_i and c_j and assign x_i to the centroid that has the smallest distance to x_i. This form the K clusters.
3. For each cluster, calculate the mean of data points assigned to it and set this value as the new centroid of that cluster, such that
$$c_j = \frac{1}{||U_j||} \sum_{x_i \in U_j} x_i. \tag{5.1}$$
4. Reassign each data point x_i to the new nearest centroid by repeating steps 2 and 3 until the centroids in the clusters no longer change.

An example visualization of the *K-means*, which is retrieved from [108], is presented in Fig. 5.1. The figure shows the initial dataset before clustering is shown on the left, and the 3 clusters generated by the *K-means* algorithm are shown on the right. The black dots in each cluster represent the centroid of that cluster.

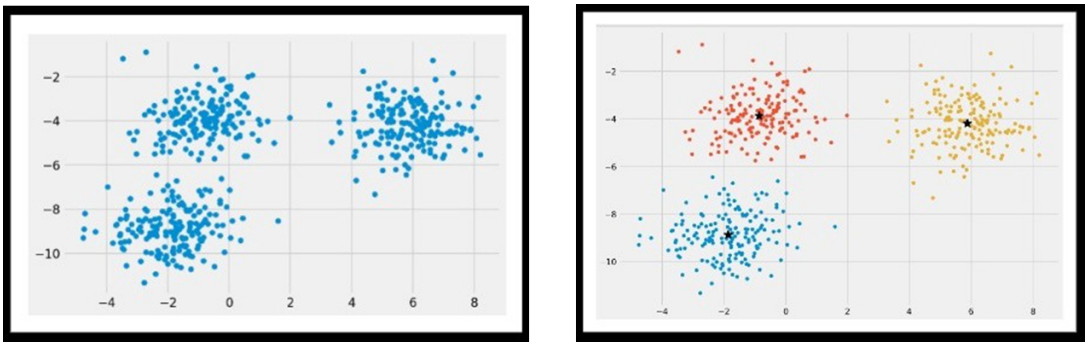

Figure 5.1: An example of how *K-means* work.

The successful clustering of *K-means* depends on how the centroids were initialized in the first step. If they are chosen very close to each other, the points are likely to find a cluster in their local area and form meaningless clusters [109]. As the centroids are randomly initialized in *K-means*, the probability of this happening is high. To tackle this problem, Arthur and Vassilvitskii [110] proposed the *K-means++* algorithm, which simply works as follows.

1. Randomly pick a centroid c_1.
2. Calculate the distance between c_1 and every other data point x_i in the dataset.
3. Find the farthest data point x_j from c_1 and make it the second centroid, such that $c_2 = x_j$ and $C = \{c_1, c_2\}$.
4. Form clusters around these centroids by assigning each data point to the nearest centroid.
5. Find the data point having the farthest distance from its centroid and make it the third centroid c_3, such that $C = \{c_1, c_2, c_3\}$.
6. Repeat steps 4 and 5 until the K number of centroids ($C = \{c_1, c_2, ..., c_K\}$) are found and K number of clusters are formed.

5.2 Clustered BOFRF Algorithm

Both *K-means* and *K-means++* have been designed to cluster single data points. In clustered BOFRF, however, the aim is to cluster sites that contain a number of data points based on their data distribution. Therefore, first, a solution was needed to formulate the data distribution of sites mathematically. In clustered BOFRF, this is handled through the "characteristic vector". Recall from Chapter 3 that the federated environment consists of N sites, where the dataset of the nth site containing m_n instances can be represented as

$$D_n = \{(X_1, y_1), (X_2, y_2), \ldots, (X_{m_n}, y_{m_n})\}. \tag{5.2}$$

In D_n, X_i represents the vector of the feature values at the ith row and $y_i \in \{+1, -1\}$ is the label used for binary classification. Let R be the total number of features in

the dataset, and f_r denote the rth feature. Then, the dataset can be represented in a tabular form as shown in Fig. 5.2

	f_1	f_2	...	f_r	y_i
X_1					
X_2					
...					
X_{m_n}					

Figure 5.2: Tabular representation of dataset D_n

The characteristic vector is thereafter defined as a vector containing the basic data distribution statistics for each column. In a dataset, a column can hold either continuous or categorical data. In the former, the data distribution statistics are calculated as the mean of values in that column, while in the latter, they are calculated as the percentage of values in each category. If one-hot encoding, which divides the categories into separate columns holding values 0 or 1, has already been performed on the categorical variables in the dataset, then either the mean or the percentage can be used to calculate statistics as they both give the same result. Consequently, the characteristic vector V_n of the nth site in the federated environment can be formulated as a one-dimensional vector of size R as shown in Eq. 5.3.

$$V_n = (S(f_1), S(f_2), ..., S(f_R)) \tag{5.3}$$

where $S(f_r)$ is the function for calculating data distribution statistics, such that

$$S(f_r) = \begin{cases} \dfrac{1}{m_n} \displaystyle\sum_{i=1}^{m_n} \gamma(norm(f_r), X_i) & \text{if } X_i \text{ is continuous} \\ \dfrac{1}{m_n} \displaystyle\sum_{i=1}^{m_n} \gamma(f_r, X_i) & \text{if } X_i \text{ is categorical} \end{cases} \tag{5.4}$$

In Eq. 5.4, $\gamma(f_r, X_i)$ represents the function that returns the value of feature f_r at the ith index (in other words, rth column and ith row) of the dataset, and $norm(f_r)$ is the normalization function which normalizes the data in the f_r column to 0-1 range. An

example of a characteristic vector for a simple dataset with 5 features and 6 instances is depicted in Fig. 5.3.

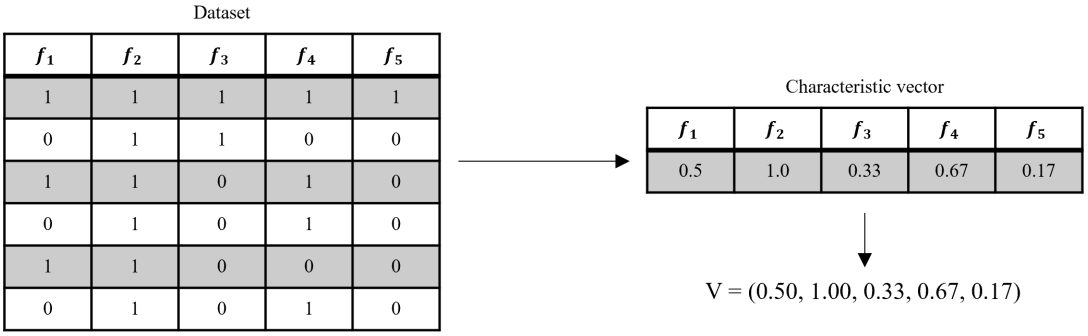

Figure 5.3: An example of a characteristic vector for a simple dataset with 5 features and 6 instances.

In the federated environment, there were originally n sites, each of which containing m_n instances. Currently, there are n sites, each with a characteristics vector V_n. Since the aim is to create clusters by apply a clustering algorithm on the participating sites, the question becomes how to calculate the distance between these sites, which is the crucial point in clustering algorithms for identifying similarities. In clustered BOFRF, the distance between sites is calculated as the distance between characteristic vectors.

In machine learning, Eucledian Distance, Manhattan Distance, Minkowski Distance and Hamming Distance are the most commonly used distance metrics [111]. Among them, Eucledian, Manhattan and Minkowski distance metrics are used for calculating the distance between two real-valued vectors, while Hamming Distance is used for calculating the distance between two binary vectors. Eucledian distance calculates the distance between two data points by drawing a straight line between them as formulated in Eq. 5.5. It is usually preferred when there are data points with numerical values such as floating point or integer. If the vectors contain data at different scales, columns with large values dominate the Eucledian distance. Therefore, to prevent this from happening, the columns holding continuous data are normalized in the $S(f_r)$ function shown in Eq. 5.4.

$$d(p,q) = \sqrt{\sum_{i=1}^{R}(q_i - p_i)^2} \tag{5.5}$$

Manhattan distance calculates the distance between two data points by following a grid-like path. Its formula is given in Eq. 5.6. It is generally preferred to the Euclidean distance as the dimension of the data increases [112].

$$d(p,q) = \sum_{i=1}^{R}|q_i - p_i| \tag{5.6}$$

Minkowski distance is a generalization of Eucledian and Manhattan metrics, which is calculated as

$$d(p,q) = (\sum_{i=1}^{R}|q_i - p_i|^t)^{1/t}. \tag{5.7}$$

The formula is the same as the Manhattan distance when $t = 1$, and the Eucledian distance when $t = 2$.

After the distance calculation metrics between the characteristics vectors are defined, the clustering step begins. In clustered BOFRF, *K-means++* is utilized as the clustering algorithm, because it is not only effective and fast, but also very suitable for use in federated environments where BOFRF is concerned as it makes real data points centroids instead of defining arbitrary centroids through Eq. 5.1. Consequently, given n sites $\{E_1, E_2, ...E_n\}$ in a federated environment, the clustered BOFRF algorithm works as follows.

1. Calculate a characteristic vector V_n at each site, such that $V = \{V_1, V_2, ..., V_n\}$.
2. Randomly pick a site E_i as the first centroid c_1.
3. Calculate the distance between c_1 and every other site in the federated environment by calculating the Eucledian or Manhattan distance between their characteristic vectors.

4. Find the site E_j whose distance is farthest from c_1 and make it the second centroid, such that $c_2 = E_j$, $C = \{c_1, c_2\}$.

5. For each site $E_k \in \{\{E_1, E_2, ...E_n\} - C\}$, find out the nearest centroid using the selected distance metric, and assign E_k to the cluster of that centroid.

6. Find the site having the farthest distance from its centroid and make it the new centroid c', such that $C = C \cup c'$.

7. Repeat steps 5 and 6 until the K number of centroids ($C = \{c_1, c_2, ..., c_K\}$) are found and K number of clusters are formed.

8. Execute BOFRF or personalized BOFRF in each cluster separately, either in a centralized or decentralized architecture.

The pseudocode of the Clustered Boosting-based Federated Random Forest algorithm is provided in Alg. 9.

5.3 Complexity Analysis of Clustered BOFRF

In this section, the computational complexity of clustered BOFRF is analyzed by going through each step of the algorithm. The first step of clustered BOFRF is to create a characteristic vector at each site by going over the entire dataset once. The cost of creating a characteristic vector at the nth site is $O(Rm_n)$, where R is the number of features in the dataset and m_n is the number of instances (lines 1-9). In the second step, a random site is selected as centroid, the cost of which is $O(1)$ (lines 11-12). In the third step, the algorithm iterates through the n number of sites, hence the cost of this step is $O(n)$ (lines 13-15). The cost of fourth step (lines 16-17) is also $O(1)$, as the algorithm already knows the farthest point while calculating distances in the previous step.

In steps 5 to 7 (lines 18-26), the algorithm loops $K - 2$ times to find other centroids and form the final clusters. For each cluster (or centroid), it requires one pass through the each site and one pass through the centroids to calculate the new distances and make new assignments, if needed. Then, the site having the farthest distance from its

centroid becomes the new centroid. Therefore, the cost is $O(KnC)$. In the worst case scenario, each site forms a cluster by its own, which would make $C = n$. As a result, the cost of steps 5 to 7 becomes $O(Kn^2)$.

When the costs of all these steps are combined, the computational complexity of clustering in the clustered BOFRF algorithm becomes $O(Rm_n) + O(1) + O(n) + O(1)+O(Kn^2)$, which can be simplified to $O(Rm'+Kn^2)$, where m' is the maximum number of instances among all the participating sites.

Consequently, the total computational complexity of clustered BOFRF algorithm becomes $O(Rm' + Kn^2 + k'm'(f' \log(m') + N))$.

Algorithm 9 Clustered Boosting-based Federated Random Forest (BOFRF)

Input: Given n sites $\{E_1, E_2, \ldots, E_n\}$, each having R features $\{f_1, f_2, ..., f_R\}$ and m_n instances

Output: The clusters U

Procedure:

for $n \leftarrow 1$ to N **do in parallel** ▷ *Step 1*
  for $r \leftarrow 1$ to R
    if f_r is continuous **then** ▷ *Eq. 5.4*
      $s_{f_r} \leftarrow \frac{1}{m_n} \sum_{i=1}^{m_n} \gamma(norm(f_r), X_i)$ ▷ *Mean of normalized values*
    else
      $s_{f_r} \leftarrow \frac{1}{m_n} \sum_{i=1}^{m_n} \gamma(f_r, X_i)$ ▷ *Percentage of values*
    end if
  end for
  $V_n \leftarrow (s_{f_1}, s_{f_2}, ..., s_{f_r})$ ▷ *Eq. 5.3*
end for
$c_1 \leftarrow RANDOM(\{E_1, E_2, \ldots, E_n\})$ ▷ *Step 2*
$C \leftarrow \{c_1\}, U_1 \leftarrow \{c_1\}$
for $n \leftarrow 1$ to N ▷ *Step 3*
  $d_i \leftarrow DISTANCE(V_{c_1}, V_{E_n})$ ▷ *Eq. 5.5 or Eq. 5.6*
end for
$c_2 \leftarrow argmin\{d_1, d_2, \ldots, d_n\}$ ▷ *Step 4*
$C \leftarrow C \cup \{c_2\}, U_2 \leftarrow \{c_2\}$
for $k \leftarrow 3$ to K
  for $n \leftarrow 1$ to N
    for $i \leftarrow 1$ to $LENGTH(C)$
      $d_i \leftarrow DISTANCE(c_i, E_n)$
    end for
    $U_j \leftarrow U_j \cup argmin\{d_1, d_2, \ldots, d_n\}$ ▷ *Step 5*
  end for
  $C \leftarrow C \cup argmax\{\forall u \in U$: distance between any two site $\}$ ▷ *Step 6*
end for
return $U = \{U_1, U_2, ..., U_K\}$

CHAPTER 6

EXPERIMENTS

In this chapter, first, the datasets that were considered in the experiments are explained in detail. Second, the environments that were set up in the experiments to evaluate the effectiveness of the proposed BOFRF algorithm is presented. Lastly, the evaluation procedure that were followed in the experiments is explained. The results of the experiments are provided in Chapter 7.

6.1 Datasets

In the experiments that were conducted in this study, four healthcare datasets were used, namely the Pima Indians Diabetes dataset, the Diabetic Retinopathy dataset, the South African Heart Disease dataset, and the SHELTER dataset. In this section, details of these datasets are explained. For each dataset, a bar chart showing the distribution of dependent variable and a correlation plot depicting the correlation between all possible pairs of features are presented. In the latter, the Pearson Correlation coefficient was utilized, which is a measure of discovering linear association between two variables where +1 indicates strong positive correlation, -1 indicates strong negative correlation, and 0 indicates no correlation.

6.1.1 The Pima Indians Diabetes dataset

The Pima Indians Diabetes dataset is originally created by the National Institute of Diabetes and Digestive and Kidney Diseases in Phoenix, Arizona, the United States. It was retrieved from Kaggle [113, 114], which is an online platform for sharing and

accessing public datasets for performing machine learning tasks. In the experiments, the dataset was used to predict whether a patient in the dataset had diabetes or not based on the measurements provided in it.

The dataset contains the data of 768 female patients who were at least 21 years old and of Pima Indian heritage. It has 8 independent features: age, body mass index (BMI), number of pregnancies, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, insulin, diabetes pedigree function; and 1 dependent feature in which 1 is interpreted as “diabetes positive” and 0 as "diabetes negative". In the dataset, 34.9% of the patients had diabetes, while 65.1% did not as illustrated in Fig. 6.1. No significant correlation was observed between the features as depicted in Fig. 6.2, hence all the features were kept in the final dataset.

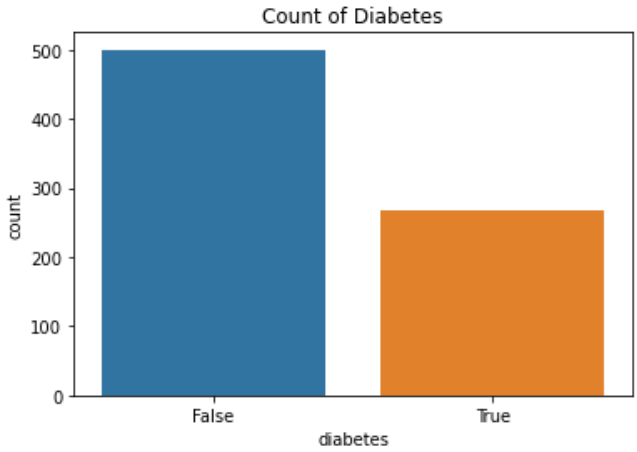

Figure 6.1: Distribution of dependent variable in the Pima Indian Diabetes dataset.

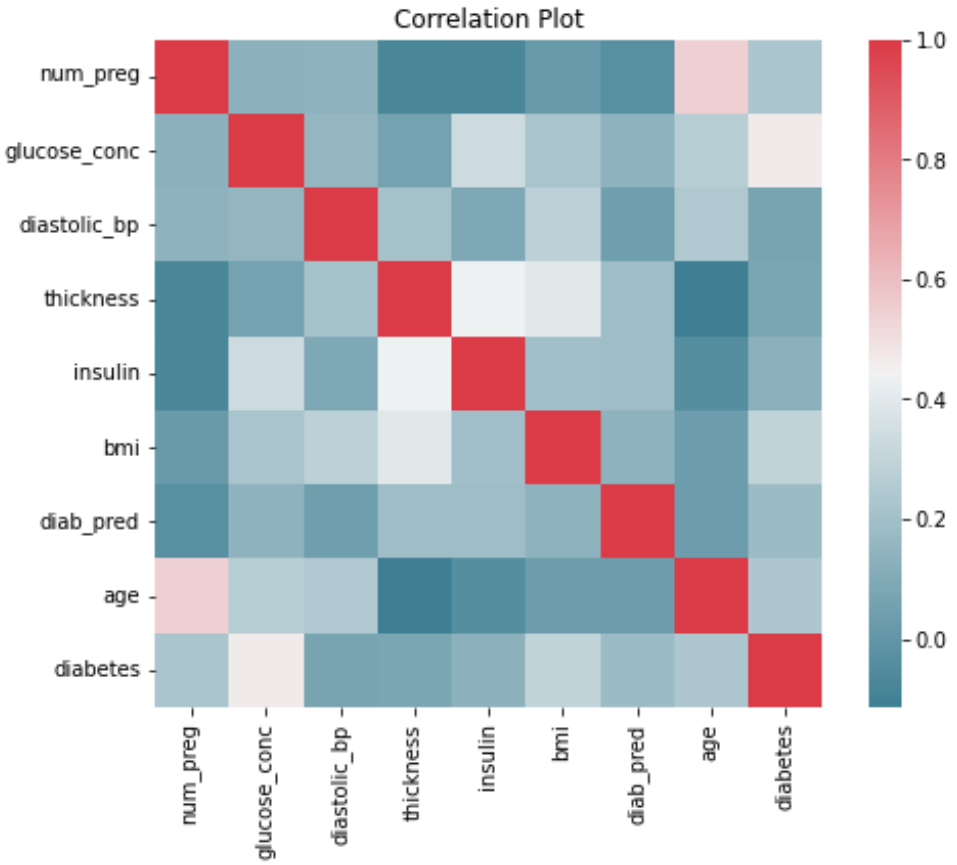

Figure 6.2: Correlation of features in the Pima Indian Diabetes dataset.

6.1.2 The Diabetic Retinopathy dataset

The Diabetic Retinopathy dataset from the University of California, Irvine (UCI) Machine Learning Repository [115, 116] contains data extracted from 1151 eye fundus images to predict whether an image contains the sign of diabetic retinopathy, which is a visual impairment caused by diabetes mellitus. It consists of 19 independent features related to a lesion in the eye, anatomical information, or an image-level descriptor; and one dependent feature, where 1 indicates a sign of diabetic retinopathy.

The Diabetic Retinopathy dataset is uniformly distributed on the dependent variable, where 53.1% of the images contained the sign of diabetic retinopathy, while 46.9% did not as shown in Fig. 6.3.

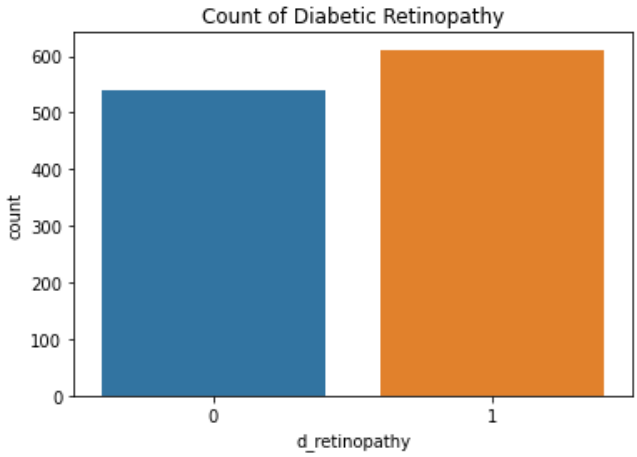

Figure 6.3: Distribution of dependent variable in the Diabetic Retinopathy dataset.

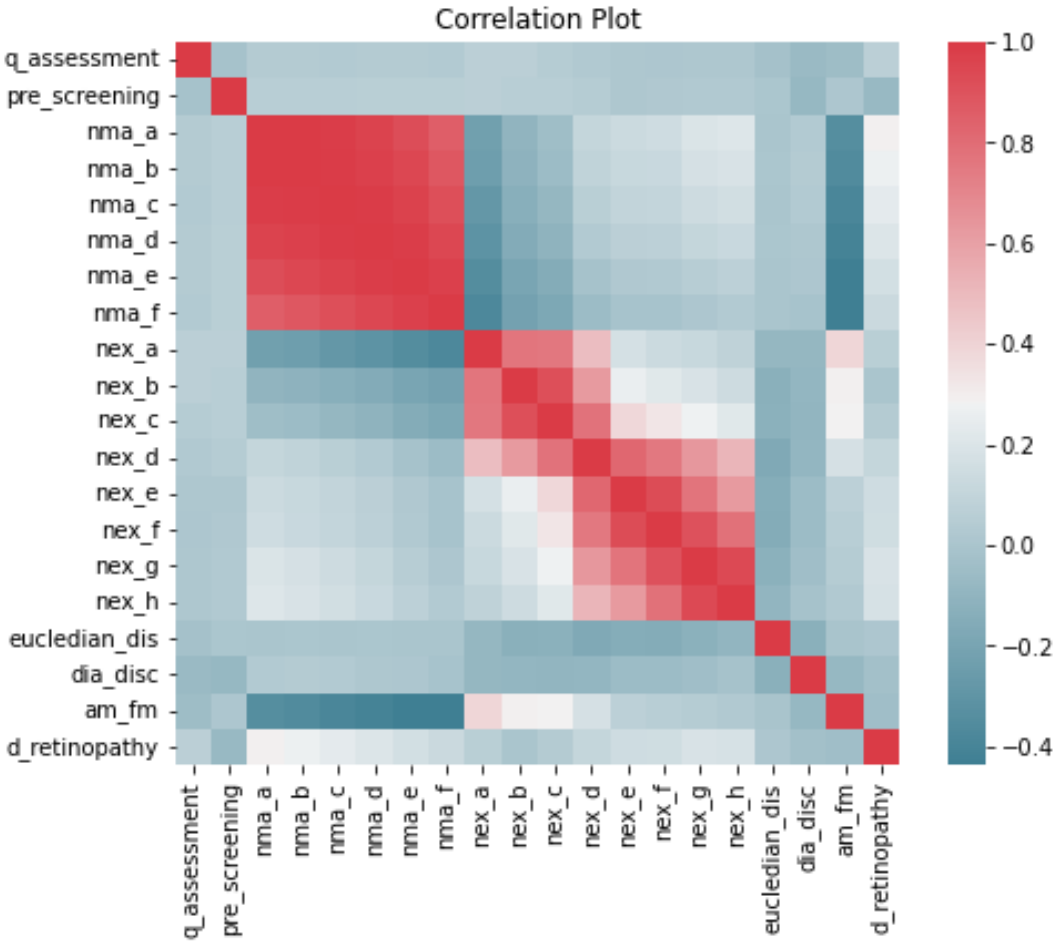

Figure 6.4: Correlation of features in the Diabetic Retinopathy dataset.

The correlation plot presented in Fig. 6.4 shows that there are strong correlations between all the nma_⋆ features and between the nex_⋆ features. nma_⋆ stands for the number of microaneurisms found at the confidence levels alpha = 0.5 (a), 0.6 (b), ..., 1 (f), respectively. nex_⋆ contains the same information as nma_⋆ for exudates. In supervised machine learning, having correlated features in the dataset may not always worsen the model, but it may not improve the model either. On the other hand, they bring additional computation cost, thus decreasing efficiency. However, if speed and time are not an issue and correlation is not with the target variable, these variables can be retained in the dataset. In the Diabetic Retinopathy dataset, it was first considered to remove some of the nma_⋆ features as they show the highest correlation between each other. However, Taveria-Gomes [117] showed in his study that a non-negligible number of points appear to be linearly inseparable in the paired projections of these features. Consequently, none of the features in the original datasets were removed in the final dataset.

6.1.3 The South African Heart Disease dataset

The South African Heart Disease dataset contains data of 462 men living in the heart-disease high-risk region of Western Cape, South Africa. Researchers took a subset of it consisting of 9 features, which are systolic blood pressure, LDL cholesterol, family history of heart disease, adiposity, obesity, type-A behavior, tobacco and alcohol usage, and age, for predicting whether a patient will suffer from coronary heart disease or not. In this study, the dataset published by Bartley in Harvard Dataverse was used [118].

In the dataset, the percentage of the patients who suffer from coronary heart disease was 34.6%, whereas the percentage of the patients who did not have this condition was 65.4% as illustrated in Fig. 6.5. None of the features showed a significant positive or negative correlation between each other as depicted in Fig. 6.6.

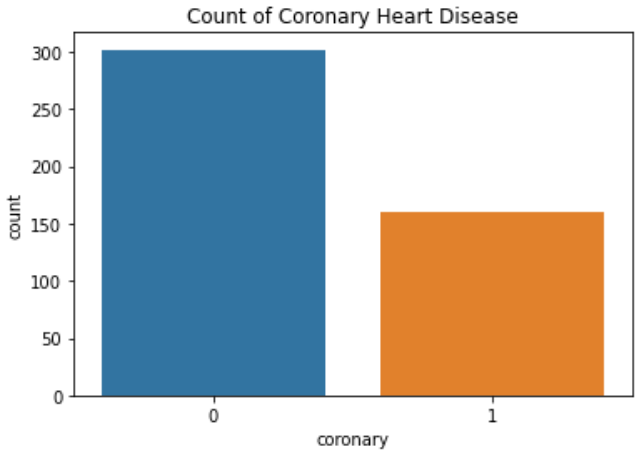

Figure 6.5: Distribution of dependent variable in the South African Heart Disease dataset.

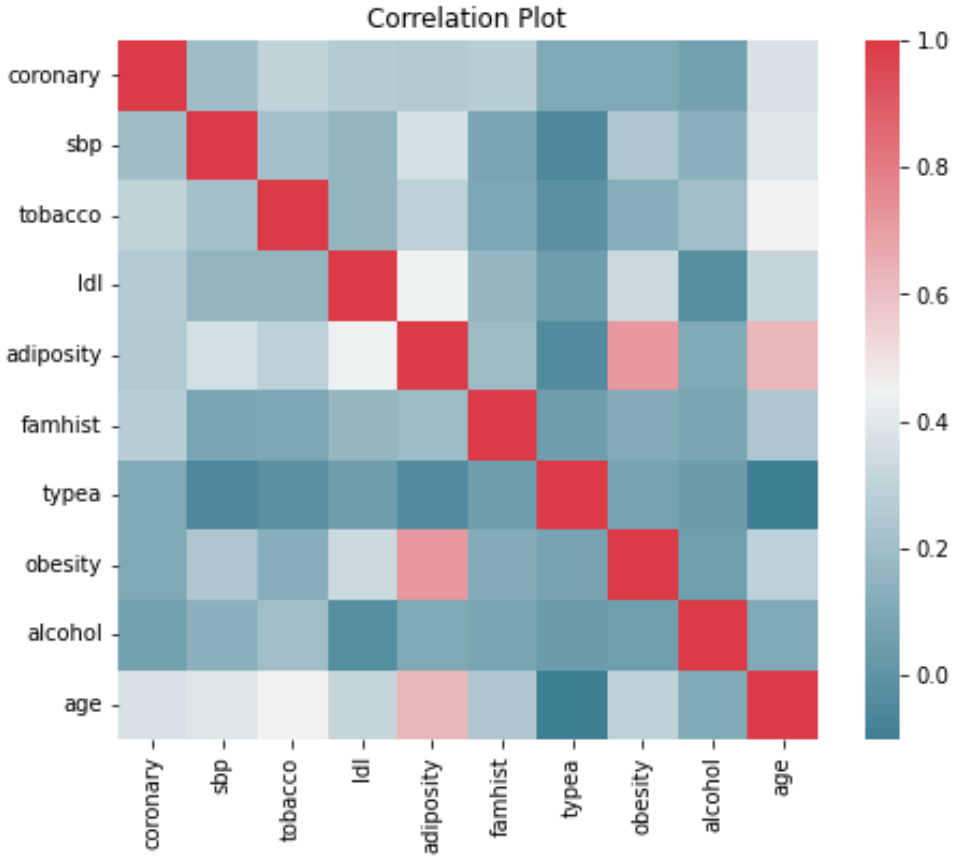

Figure 6.6: Correlation of features in the South African Heart Disease dataset.

6.1.4 The SHELTER dataset

The SHELTER (Services and Health for Elderly in Long TERm care) was an international project funded by the European Union's 7th Framework Programme under grant agreement number 223115 [119]. In this study, a research was conducted between 2009 and 2011 to assess the care needs and provision of care to nursing home (NH) residents in Europe and collected the data of 4156 NH residents in eight countries: 500 from Czechia, 507 from England, 484 from Finland, 493 from France, 496 from Germany, 580 from Israel, 548 from Italy, and 548 from the Netherlands [120]. Onder *et al.* [121] analyzed the determinants of excessive polypharmacy (usage of more than 10 drugs) in the SHELTER dataset and identified the factors that are associated with excessive polypharmacy.

Utilizing this information, in this study, 14 features were extracted from the SHELTER dataset, which are gender, assistance required in activities of daily living (ADL), dependent in ADL, mild/moderate cognitive impairment (CGI), severe CGI, depression, pain, dyspnea, dizziness, coronary heart disease, heart failure, Parkinson's disease, stroke, diabetes, cancer. This dataset then was used to predict excessive polypharmacy risk of NH residents.

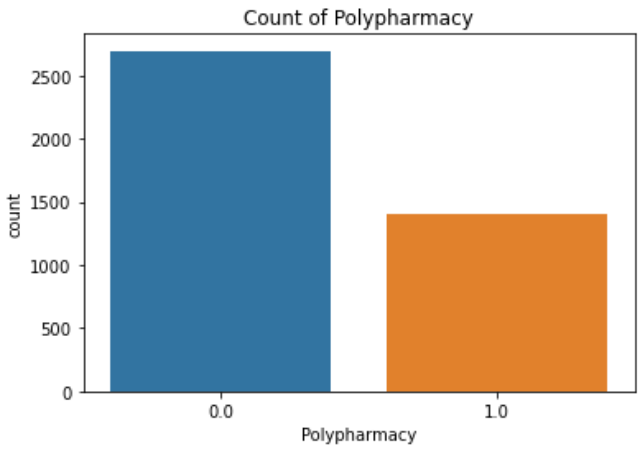

Figure 6.7: Distribution of dependent variable in the SHELTER dataset.

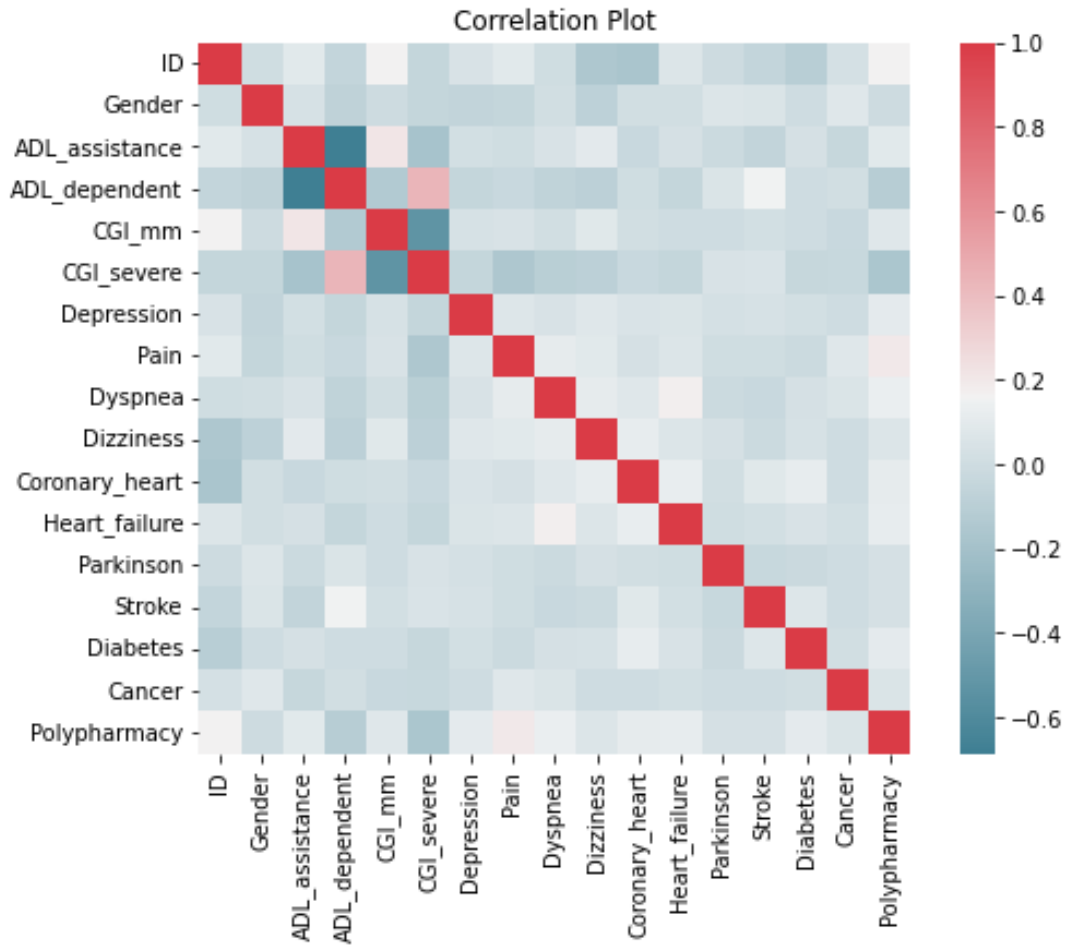

Figure 6.8: Correlation of features in the SHELTER dataset.

In the final dataset, 34.3% of the patients were using more than 10 drugs, thus in excessive polypharmacy status, while 65.7% were in polypharmacy (5 to 9 drugs) or non-polypharmacy (<5 drugs) status as shown in Fig. 6.7. The features were not correlated with each other as depicted in Fig. 6.8.

6.2 Experiment Setup

In this study, two types of experiments were conducted. In the first type, which is called observational experiments, the Pima Indians Diabetes, Diabetic Retinopathy, and South African Heart Disease datasets were used to set up different federated environments to prove the effectiveness of BOFRF in challenging scenarios of feder-

ated learning. In this regard, federated environments consisting of different numbers of sites (i.e., 2, 3, or 4) were set up by splitting the datasets into several datasets with different characteristics, i.e., datasets producing well-performing local models or datasets producing unsuccessful local models. The setup and evaluation procedures of these experiments are illustrated in Fig. 6.9. In the setup, the dataset is split into n sites constituting the federated environment. In the evaluation, data at each site are split as training and test, and local Random Forest models are generated. After federated BOFRF model is built on top of the local models, it is evaluated on test data of each site.

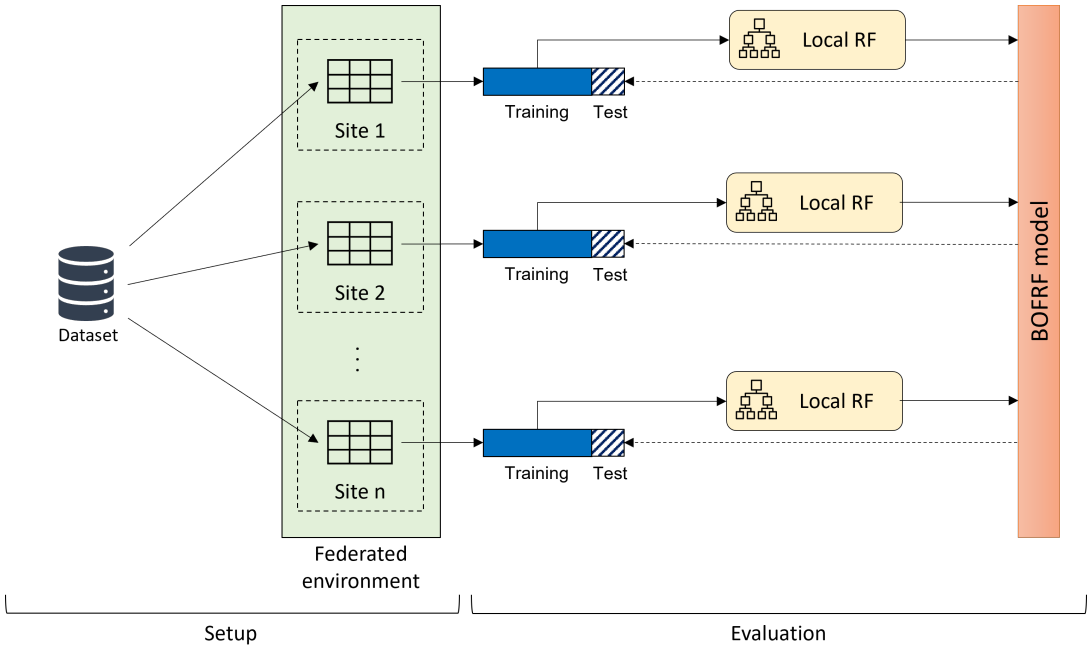

Figure 6.9: The setup and evaluation procedure of the experiments.

In the second type of experiments, which were conducted on the SHELTER dataset, the same setup procedure was not followed because the dataset was already split according to origin countries. Instead, a total of 247 federated environments consisting of two, three, four, five, six, seven, and eight sites were built with all possible combinations of the eight countries involved in the dataset. In these experiments, statistical testing of BOFRF was performed to observe how well BOFRF performs in real federated environments that were not emulated, as well as how effective it is on larger datasets and increasing number of participants. Tables 6.1, 6.2, 6.3 and 6.4 show the

details of the environments, including the total number of records at each site and the number of positive and negative labelled instances.

Table 6.1: Number of sites, total number of records, and the number of positive and negative labelled instances in the federated environments built with the Pima Indians Diabetes dataset in the experiments.

Env. #	Dataset	Site	Records	Positive labels	Negative labels
Env. 1	Diabetes	Site 1	400	152	248
		Site 2	368	116	252
Env. 2	Diabetes	Site 1	300	114	186
		Site 2	200	68	132
		Site 3	268	86	182
Env. 3	Diabetes	Site 1	300	107	193
		Site 2	200	73	127
		Site 3	268	88	180
Env. 4	Diabetes	Site 1	100	33	67
		Site 2	250	94	156
		Site 3	280	57	223
		Site 4	138	84	54
Env. 5	Diabetes	Site 1	200	64	136
		Site 2	200	74	126
		Site 3	200	68	132
		Site 4	168	62	106
Env. 6	Diabetes	Site 1	100	37	63
		Site 2	250	97	153
		Site 3	300	89	211
		Site 4	118	45	73

Table 6.2: Number of sites, total number of records, and the number of positive and negative labelled instances in the federated environments built with the Diabetic Retinopathy dataset in the experiments.

Env. #	Dataset	Site	Records	Positive labels	Negative labels
Env. 7	Diabetic Retinopathy	Site 1	651	340	311
		Site 2	500	271	229
Env. 8	Diabetic Retinopathy	Site 1	400	218	182
		Site 2	300	158	142
		Site 3	451	235	216
Env. 9	Diabetic Retinopathy	Site 1	651	340	311
		Site 2	300	157	143
		Site 3	200	114	86

Table 6.3: Number of sites, total number of records, and the number of positive and negative labelled instances in the federated environments built with the South African Heart Disease dataset in the experiments.

Env. #	Dataset	Site	Records	Positive labels	Negative labels
Env. 10	Heart Disease	Site 1	90	36	54
		Site 2	372	124	248
Env. 11	Heart Disease	Site 1	200	80	120
		Site 2	150	55	95
		Site 3	112	25	87
Env. 12	Heart Disease	Site 1	150	57	93
		Site 2	150	53	97
		Site 3	162	50	112

Table 6.4: Total number of records and the number of positive and negative labelled instances in each site of the SHELTER dataset.

Env. #	Dataset	Site	Records	Positive labels	Negative labels
Environments on SHELTER		Czechia	500	170	330
		Germany	490	173	317
		England	507	179	328
		Finland	448	299	149
		France	490	209	281
		Israel	579	131	448
		Italy	540	89	451
		Netherlands	548	157	391

6.3 Evaluation Procedure

In each experiment, first, the results of the proposed BOFRF algorithm were compared with the baseline, which is the local random forest algorithm of each participating site, to show how BOFRF can improve the predictive power of the baseline local model. While building the local random forest models, 10-fold cross-validation was applied and the grid search method was used to avoid overfitting and determine the best model with the best hyperparameter combination. To ensure privacy of sensitive data and enable the calculation of the AUC, some restrictions were put on the hyperparameter values that are explored by the search grid, such as the minimum leaf size. For instance, a suspicious site may attempt to identify a subject through a decision tree that has a leaf node with only one subject. Therefore, a value of 1 for the minimum leaf size was not allowed so that none of the sites could generate such decision trees. This also makes the calculation of prediction probabilities possible for each decision tree since no leaf node contains a single class estimate. The prediction probabilities of decision trees are required to calculate the AUC values of the random forest models and the BOFRF model. Then, to evaluate the success of the proposed algorithm against existing solutions, the results of BOFRF were compared with those

of the federated model generated by one of the most successful existing solutions in the field, namely BOPPID. For a fair comparison, the BOPPID algorithm proposed by Li *et al.* in their paper [18] was implemented in Python and both BOPPID and local AdaBoost, which is the baseline of BOPPID, were ran in each experiment.

CHAPTER 7

RESULTS

This chapter presents the results of the observational and statistical experiments that were conducted on the datasets and environments explained in Chapter 6. In both types of experiments, first, the base BOFRF algorithm is compared with local RF and BOPPID. Then, for each scenario, the improvement that personalized BOFRF provided over the base BOFRF is presented. Finally, the experimental results obtained for the clustered BOFRF are shown.

7.1 Results of Observational Experiments

7.1.1 Comparison of BOFRF with Local Random Forest

The Pima Indian Diabetes dataset was utilized to set up six different environments with two different characteristics: (i) all sites had quite good models with relatively close AUC values (Environments 1 and 3), and (ii) one site had a very good model with a high AUC value, and at least one site had a poor model with a low AUC value (Environments 2, 4, 5 and 6). The results are presented in Table 7.1. In this table and the tables shown hereinafter in this section, for better visualization, environments are abbreviated as E1, E2, etc., whereas sites are abbreviated as S1, S2, etc. The (*) near the values in the "Change" columns highlights the improvements that BOFRF provided significantly high.

Table 7.1: Comparison of algorithms based on the AUC values in federated environments built on the Pima Indian Diabetes dataset.

Env. #	Site #	Local Ada	BOPPID	Change (%)	Local RF	**BOFRF**	Change (%)
E1	S1	0.792	0.811	+ 2.45	0.763	**0.841**	+10.29 (*)
	S2	0.805	0.816	+ 1.41	0.828	**0.843**	+ 1.76
E2	S1	0.776	**0.809**	+ 4.37	0.738	**0.802**	+ 8.59
	S2	0.772	0.823	+ 6.52	0.752	**0.856**	+13.86 (*)
	S3	0.883	0.906	+ 2.65	0.903	**0.923**	+ 2.24
E3	S1	0.832	0.834	+ 0.90	0.805	**0.840**	+ 4.29
	S2	0.857	0.859	+ 0.23	0.841	**0.865**	+ 2.83
	S3	0.868	**0.845**	- 2.70	0.777	**0.803**	+ 2.81
E4	S1	0.840	0.841	+ 0.10	0.784	**0.861**	+ 9.74
	S2	0.769	0.819	+ 6.46	0.763	**0.832**	+ 9.03
	S3	0.757	0.843	+11.37	0.734	**0.882**	+20.17 (*)
	S4	0.896	0.911	+ 1.64	0.926	**0.937**	+ 1.18
E5	S1	0.839	0.885	+ 5.48	0.791	**0.903**	+12.31
	S2	0.819	0.833	+ 1.57	0.735	**0.833**	+11.7 (*)
	S3	0.895	0.885	- 1.12	0.881	**0.888**	+ 0.76
	S4	0.656	0.745	+13.55	0.714	**0.849**	+15.86 (*)
E6	S1	0.718	0.818	+13.39	0.686	**0.904**	+24.29 (*)
	S2	0.812	**0.816**	+ 0.53	0.757	**0.801**	+ 5.47
	S3	0.927	0.936	+ 0.85	0.945	**0.950**	+ 0.46
	S4	0.755	0.750	- 0.66	0.750	**0.780**	+ 3.85

In all cases, BOFRF successfully improved the performance of all local RF models. In cases of (i), the percentage of improvements was usually around the same level

at all sites, that is between 1-6%. In cases of (ii), a significant improvement was observed in sites having poor models with low AUC values. BOFRF increased the local AUC value from 0.752 to 0.856 in Environment 2 (13.86% increase), from 0.734 to 0.882 in Environment 4 (20.17% increase), from 0.735 to 0.833 and from 0.714 to 0.849 in Environment 5 (11.7% and 15.86% increase, respectively), and from 0.686 to 0.904 in Environment 6 (24.29% increase). In these settings, the improvement on the best-performing model was usually around 0-2%, but this is something expected because these models can only be improved up to a certain level as they already have high AUC values. Fig. 7.1 shows the comparison of the local RF with BOFRF on the Pima Indian Diabetes dataset. In the figure, orange bars represent the performance of local RF, whereas green bars represent the performance of BOFRF.

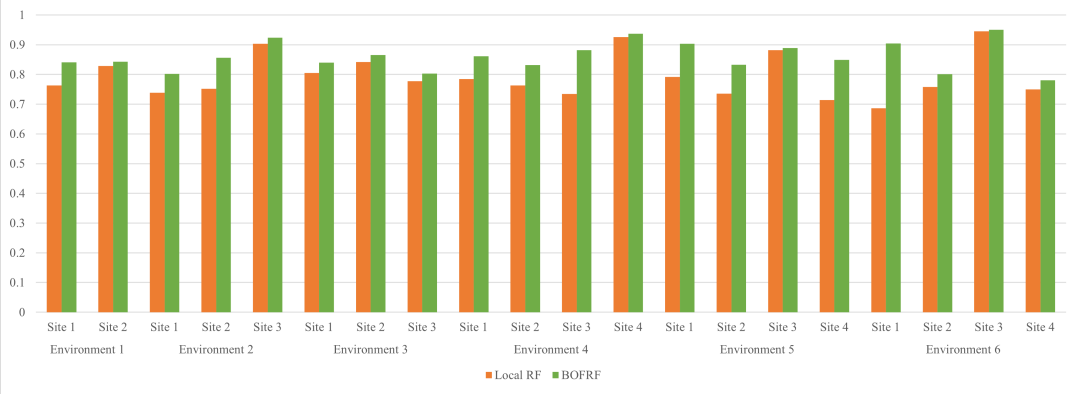

Figure 7.1: Comparison of the local RF with BOFRF in the Pima Indian Diabetes dataset.

On the Diabetic Retinopathy dataset, three environments were set up, i.e., Environments 7, 8, and 9 as shown in Table 7.2. In Environment 8, where the participating sites had AUC values close to each other, the same level of improvement was observed as in the Pima Indian Diabetes dataset. In Environment 7, there were two sites, Site 1 and Site 2, with AUC values of 0.666 and 0.742, respectively. BOFRF improved the AUC values by 9.8% and 2.03%, respectively. In Environment 9, Site 1 was kept as it is and two new sites having AUC values as low as Site 2 in Environment 7 were introduced. In this case, BOFRF significantly improved the AUC values by 20.44% and 30.72%.

Table 7.2: Comparison of algorithms based on the AUC values in federated environments built on the Diabetic Retinopathy dataset.

Env. #	Site #	Local Ada	BOPPID	Change (%)	Local RF	**BOFRF**	Change (%)
E7	S1	0.692	0.718	+ 3.76	0.742	**0.757**	+ 2.03
	S2	0.693	0.695	+ 0.32	0.666	**0.732**	+ 9.80 (*)
E8	S1	0.710	0.720	+ 1.49	0.708	**0.727**	+ 2.69
	S2	0.703	0.768	+ 9.18	0.774	**0.788**	+ 1.85
	S3	0.684	0.739	+ 2.98	0.704	**0.740**	+ 5.12
E9	S1	0.692	0.705	+ 1.80	0.742	**0.745**	+ 0.31
	S2	0.729	0.747	+ 2.41	0.658	**0.793**	+20.44 (*)
	S3	0.707	0.823	+16.51	0.638	**0.833**	+30.72 (*)

Figure 7.2: Comparison of the local RF and BOFRF in the Diabetic Retinopathy dataset.

On the South African Heart Disease dataset, sites having the lowest AUC values in all experiments were introduced and the performance improvement was evaluated. The results for Environments 10, 11, and 12 are shown in Table 7.3. In Environment 10, Site 1 had a local RF model with an AUC value of 0.580, and Site 2 had a local RF model with an AUC value of 0.758. After applying BOFRF, the AUC values increased

by 30.77% and 4.71%, and became 0.759 and 0.793, respectively. In Environment 11, Site 3 had a very low AUC value of 0.573 due to its imbalanced data with only 22% positive. BOFRF provided 21.21% of improvement on this site and increased its AUC value to 0.696. Similarly, it increased the AUC of Site 1 in Environment 11 from 0.618 to 0.709 by 14.7%, and the AUC of Site 2 in Environment 12 from 0.599 to 0.685 by 14.22%. The overall comparison of local RF and BOFRF in the Diabetic Retinopathy and South African Heart Disease datasets is displayed in Fig. 7.2 and Fig. 7.3, respectively.

As a result of these experiments, it was observed that for all sites, the proposed BOFRF algorithm successfully improved the prediction performance of the local RF. In particular, BOFRF significantly improved the prediction performance of sites whose local model was poor (i.e., having an AUC value less than 0.75). Such cases were highlighted in Tables 7.1, 7.2 and 7.3 with "(*)" near the percentage of change values. For instance, in Environments 6, 9, and 10, it was observed that BOFRF increased the local AUC value from 0.686 to 0.904 (24.29% increase), 0.638 to 0.833 (30.72% increase), and 0.580 to 0.759 (30.77% increase), respectively, with the help of other well-performing models. Furthermore, it was also observed that even a site with a poor local model can provide a remarkable contribution to the other sites in BOFRF as presented in Environments 7 and 9.

Table 7.3: Comparison of algorithms based on the AUC values in federated environments built on the South African Heart Disease dataset.

Env. #	Site #	Local Ada	BOPPID	Change (%)	Local RF	**BOFRF**	Change (%)
E10	S1	0.509	0.607	+ 19.30	0.580	**0.759**	+30.77 (*)
	S2	0.708	0.708	0.00	0.758	**0.793**	+ 4.71
E11	S1	0.613	0.653	+ 6.52	0.618	**0.709**	+14.70 (*)
	S2	0.726	0.751	+ 3.39	0.771	**0.773**	+ 0.19
	S3	0.582	0.669	+14.92	0.573	**0.696**	+21.21 (*)
E12	S1	0.728	0.683	- 6.17	0.692	**0.718**	+ 3.07
	S2	0.597	0.679	+13.81	0.599	**0.685**	+14.22 (*)
	S3	0.629	0.687	+ 9.13	0.724	**0.741**	+ 2.38

Figure 7.3: Comparison of the local RF and BOFRF in the South African Heart Disease dataset.

7.1.2 Comparison of BOFRF with BOPPID

In the observational experiments, it was observed that BOFRF provided better AUC results than BOPPID in 91.6% of the cases. In addition, the percentage of improvement provided by BOFRF was higher than that provided by BOPPID in 77.7% of the

cases as shown in Figures 7.4, 7.5 and 7.6, where blue bars represent the percentage of improvement provided by BOPPID, whereas green bars represent the percentage of improvement provided by BOFRF. The average change percentages of BOFRF and BOPPID were 9.04% and 4.67% respectively, which shows that BOFRF can improve the performance of the local models better than BOPPID.

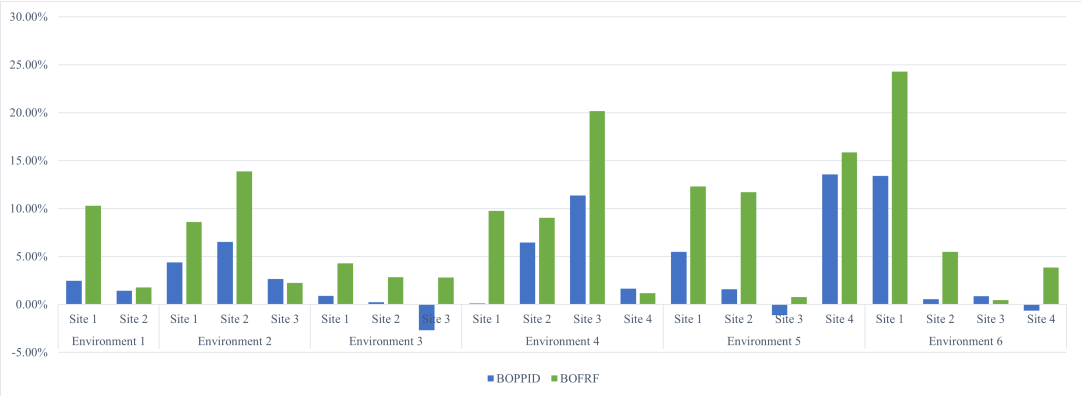

Figure 7.4: Comparison of the percentage of improvement provided by BOPPID and BOFRF on their baseline local models in observational experiments conducted on the Pima Indian Diabetes dataset.

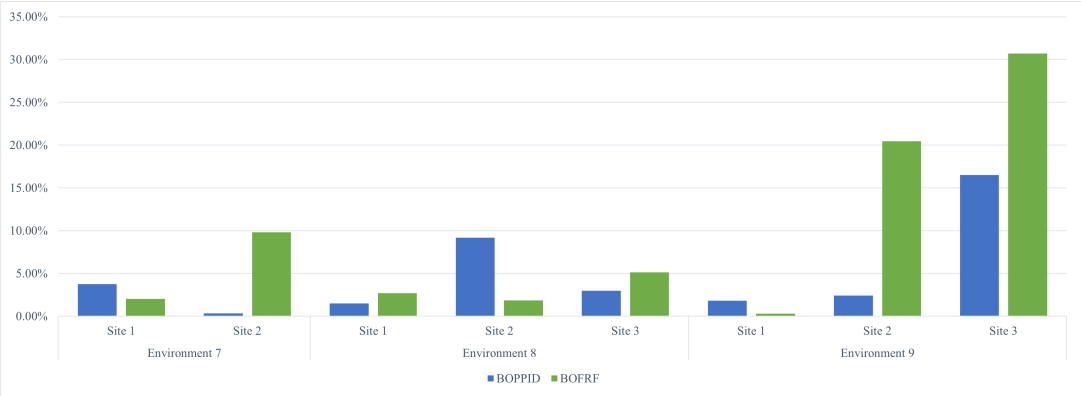

Figure 7.5: Comparison of the percentage of improvement provided by BOPPID and BOFRF on their baseline local models in observational experiments conducted on the Diabetic Retinopathy dataset.

Figure 7.6: Comparison of the percentage of improvement provided by BOPPID and BOFRF on their baseline local models in observational experiments conducted on the South African Heart Disease dataset.

7.1.3 BOFRF vs Personalized BOFRF

In the integration step of BOFRF, weak classifiers having an MCC value below a certain threshold value, are removed to improve the prediction performance of the final federated ensemble classifier. The results in the observational experiments presented in the previous sections were obtained with a threshold value of 0.2. Then, secondly in this study, a personalized version of BOFRF was implemented, where each participant fine-tunes the hyperparameters of BOFRF locally, including threshold, ensemble strategy and parameters of random forest, to come up with a better-performing federated model on their own datasets.

In the second type of observational experiments, the personalized BOFRF was run in the same environments where the base BOFRF was run and the results obtained by both were compared. The AUC values that were obtained by the base BOFRF and personalized BOFRF in federated environments built on the Pima Indian Diabetes, Diabetic Retinopathy and South African Heart Disease datasets and the threshold value (τ) giving the corresponding AUC value are presented in Tables 7.4, 7.5 and 7.6, respectively. The values of all hyperparameters are provided in Appendix A.

As shown in the tables, using a different threshold value than 0.2 gave better AUC result in 86.1% of the cases, while the threshold value of 0.2 gave the best result in

%13.9 of the cases. Among 36 cases in total, the highest improvements that personalized BOFRF provided over BOFRF were observed in Site 2 in Environment 3 (3.93%), Site 3 in Environment 9 (4.08%), Site 3 in Environment 11 (4.02%), and Site 3 in Environment 12 (5.13%). The corresponding threshold values for these are 0.3, 0.3, 0.4 and 0.1, respectively. This shows that in some cases using a larger threshold value, which eliminates a greater number of weak classifiers, helps producing better-performing federated models; while in some other cases, it is better to use a smaller threshold value, which allows to include weaker classifiers in the federated model. That's why it is a good idea to use a fine-tuning approach to personalize of BOFRF for specific sites.

Table 7.4: AUC comparison of base BOFRF and personalized BOFRF in federated environments built on the Pima Indian Diabetes dataset.

Env. #	Site #	Local RF	BOFRF	**Personalized BOFRF**	Improvement (%)	τ
E1	S1	0.763	0.841	0.841	0.00	0.2
	S2	0.828	0.843	**0.860**	+ 2.02	0.3
E2	S1	0.738	0.802	**0.805**	+ 0.37	0.15
	S2	0.752	0.856	**0.881**	+ 2.92	0.25
	S3	0.903	0.923	**0.950**	+ 2.93	0.25
E3	S1	0.805	0.840	**0.853**	+ 1.55	0.1
	S2	0.841	0.865	**0.899**	+ 3.93	0.3
	S3	0.777	0.803	**0.824**	+ 2.62	0.3
E4	S1	0.784	0.861	**0.869**	+ 0.93	0
	S2	0.763	0.832	**0.841**	+ 1.05	0
	S3	0.734	0.882	**0.888**	+ 0.68	0.1
	S4	0.926	0.937	0.937	0.00	0.2
E5	S1	0.791	0.903	0.903	0.00	0.2
	S2	0.735	0.833	**0.846**	+ 1.56	0.3
	S3	0.881	0.888	0.888	0.00	0.2
	S4	0.714	0.849	**0.856**	+ 0.82	0.3
E6	S1	0.686	0.904	**0.911**	+ 0.77	0.3
	S2	0.757	0.801	**0.805**	+ 0.50	0.4
	S3	0.945	0.950	**0.953**	+ 0.32	0.25
	S4	0.750	0.780	**0.790**	+ 0.78	0.4

Table 7.5: AUC comparison of base BOFRF and personalized BOFRF in federated environments built on the Diabetic Retinopathy dataset.

Env. #	Site #	Local RF	BOFRF	**Personalized BOFRF**	Improvement (%)	τ
E7	S1	0.742	0.757	**0.758**	+ 0.13	0
	S2	0.666	0.732	**0.742**	+ 1.37	0.4
E8	S1	0.708	0.727	0.727	0.00	0.2
	S2	0.774	0.788	**0.790**	+ 0.25	0
	S3	0.704	0.740	**0.741**	+ 0.14	0
E9	S1	0.742	0.745	**0.746**	+ 0.13	0.3
	S2	0.658	0.793	**0.795**	+ 0.25	0.3
	S3	0.638	0.833	**0.867**	+ 4.08	0.3

Table 7.6: AUC comparison of base BOFRF and personalized BOFRF in federated environments built on the South African Heart Disease dataset.

Env. #	Site #	Local RF	BOFRF	**Personalized BOFRF**	Improvement (%)	τ
E10	S1	0.580	0.759	**0.768**	+ 1.19	0.3
	S2	0.758	0.793	**0.799**	+ 0.76	0
E11	S1	0.618	0.709	**0.714**	+ 0.71	0.4
	S2	0.771	0.773	**0.783**	+ 1.29	0.4
	S3	0.573	0.696	**0.724**	+ 4.02	0.4
E12	S1	0.692	0.718	0.718	0.00	0.2
	S2	0.599	0.685	0.685	0.00	0.2
	S3	0.724	0.741	**0.779**	+ 5.13	0.1

7.2 Results of Statistical Experiments

The statistical experiments were performed on the SHELTER dataset. In these experiments, 247 environments were built in total, consisting of every possible combination of countries in federated settings with two, three, four, five, six, seven, and eight sites. For example, Czechia (CZ) is involved in 7 settings with two sites (Czechia and another country), 21 settings with three sites (Czechia and two more countries), 35 settings with four sites, 35 settings with five sites, 21 settings with six sites, 7 settings with seven site, and 1 setting with eight sites. In each environment, first, the local random forest model, which is the baseline of BOFRF, was run at each site (country) without using the federated learning approach. Then, the BOPPID and BOFRF algorithms were run in each of the 247 environments, and the mean AUC and accuracy values were calculated for each country. In Tables 7.7 and 7.8, these values are compared along with their standard deviations and *p-value*s.

Table 7.7: Achieved AUC values of countries along with their standard deviations in real federated environments built on the SHELTER dataset.

Site	Non-federated	Federated		
	Local RF (Baseline)	BOPPID	**BOFRF**	*p-value*
CZ	0.621 ± 0.039	0.630 ± 0.016	$\mathbf{0.659 \pm 0.026}$	0.000
DE	0.677 ± 0.039	0.645 ± 0.021	$\mathbf{0.688 \pm 0.005}$	0.000
EN	0.645 ± 0.023	0.681 ± 0.011	$\mathbf{0.692 \pm 0.012}$	0.001
FI	0.699 ± 0.032	0.687 ± 0.011	$\mathbf{0.715 \pm 0.019}$	0.000
FR	0.702 ± 0.007	0.685 ± 0.004	$\mathbf{0.714 \pm 0.005}$	0.000
IL	0.743 ± 0.026	0.751 ± 0.005	$\mathbf{0.777 \pm 0.003}$	0.000
IT	0.703 ± 0.046	0.687 ± 0.013	$\mathbf{0.705 \pm 0.015}$	0.000
NL	0.767 ± 0.015	$\mathbf{0.794 \pm 0.008}$	0.781 ± 0.003	1.000

Table 7.8: Achieved accuracy values of countries along with their standard deviations in real federated environments built on the SHELTER dataset.

Site	Non-federated	Federated		
	Local RF (Baseline)	BOPPID	**BOFRF**	*p-value*
CZ	0.680 ± 0.035	0.692 ± 0.010	$\mathbf{0.717 \pm 0.017}$	0.000
DE	0.658 ± 0.020	0.641 ± 0.014	$\mathbf{0.663 \pm 0.010}$	0.000
EN	0.661 ± 0.042	0.686 ± 0.013	$\mathbf{0.697 \pm 0.010}$	0.001
FI	0.623 ± 0.011	0.631 ± 0.048	$\mathbf{0.702 \pm 0.005}$	0.000
FR	0.609 ± 0.015	0.603 ± 0.013	$\mathbf{0.646 \pm 0.014}$	0.000
IL	0.831 ± 0.043	0.834 ± 0.005	$\mathbf{0.848 \pm 0.003}$	0.000
IT	0.849 ± 0.018	0.861 ± 0.005	$\mathbf{0.862 \pm 0.008}$	0.228
NL	0.744 ± 0.005	0.763 ± 0.016	$\mathbf{0.773 \pm 0.008}$	0.001

The statistical experiments confirmed the findings obtained in observational experiments; that is, BOFRF improved the prediction power of the baseline local random forest model in all cases. In particular, in Table 7.7, the best improvement was observed in Czechia (CZ) and England (EN), which had the lowest AUC value in their baseline random forest models, that is, the two most unsuccessful local classifiers. On the other hand, BOPPID provided the best improvement on the site having the best performing local model, the Netherlands (NL), which is also the only case in which BOPPID produced better result than BOFRF. This is understandable because BOPPID is designed to give more importance to the site's own local model than the others in the federated model. However, in all the other cases, BOFRF produced better results than BOPPID.

In statistics, the hypothesis testing can be performed as either *one-tailed* or *two-tailed*, where a tail refers to the side at either end of a distribution curve [122]. If the hypothesis is about testing if "X is greater than Y" or "X is smaller than Y", then one-tailed test is applied. If it is about testing if "X is not equal to Y", then two-tailed test

is applied. The right-tailed test and left-tailed test are two types of one-tailed tests. Right-tailed tests are used when the hypothesis contains a greater than symbol, and left-tailed are used when the hypothesis contains a smaller than symbol.

To statistically verify that BOFRF outperforms BOPPID, right-tailed Z-test was applied. The alternative hypothesis H_1 was that the mean AUC value of BOFRF (μ) is greater than the mean AUC value of BOPPID (μ_0) at a given level of significance. The null hypothesis was just the opposite as formulated in (7.1).

$$\begin{aligned} H_0 &: \mu \leq \mu_0 \\ H_1 &: \mu > \mu_0 \end{aligned} \tag{7.1}$$

After the hypothesis statements were written, the *Z-score*s were calculated using the formula

$$Z = \frac{\overline{x} - \mu_0}{\sigma - \sqrt{n}} \tag{7.2}$$

where $\overline{x}$ is the mean AUC value of BOFRF, σ is the standard deviation, and n is the sample size. After the Z-tests were run, the corresponding *p-value* of each test were calculated based on the output *Z-score*s. The level of significance was determined as $\alpha = 0.05$. If the *p-value* ≤ 0.05, the null hypothesis should be rejected, otherwise it cannot be rejected. The null hypothesis can also be rejected if the *Z-score* ≥ 1.645, which is the critical value for the significance level of 0.05. As shown in Table 7.7, the *p-value* of all sites except the Netherlands were either 0.000 or 0.001, which means that the null hypothesis is rejected, hence the alternative hypothesis is true. The reason that *p-value*s of the sites had such low values was because their corresponding *Z-score*s had extremely high values, ranging from 7 to 52. Appendix B contains the complete list of *Z-score*s. The results obtained with the accuracy values were in line with those obtained with the AUC values. As shown in Table 7.8, BOFRF gave better accuracy results than BOPPID in all cases. The only *p-value* that was not below 0.05, hence not rejecting the null hypothesis, was that of Italy. Consequently, in seven out of eight sites in both AUC and accuracy, statistically significant evidences were obtained to confirm that BOFRF outperforms BOPPID.

Fig. 7.7 and Fig. 7.8 elaborate the comparison of AUC and accuracy values shown in Table 7.7 and Table 7.8 respectively, and provide line charts to depict the influence of the increasing number of sites on the performance of the algorithms. In the figures, the blue lines represent the mean values of BOFRF, whereas the orange lines represent the mean values of BOPPID. The AUC/accuracy value of the baseline local RF model is represented by a straight green line. To illustrate, in Fig. 7.7, the mean AUC values of BOFRF for Czechia in federated settings with two, three, four, five, six, and seven sites were calculated as 0.643, 0.649, 0.658, 0.661, 0.665, and 0.678, respectively. The blue-painted areas in the figure show the range in which the values of BOFRF change, that is, the area between the minimum and the maximum values, and the blue lines indicate the average values for each setting. In Czechia, England, Finland, Italy, and the Netherlands, the blue line followed an ascending path, implying that for an increasing number of sites, the performance of the BOFRF increased as well. In the case of France, increasing number of sites did not significantly affect the result; hence, the line followed a straight path, indicating that the same level of improvement was observed in all scenarios. A decrease in performance for an increasing number of sites was only observed in Germany and Israel, but in both cases, the rate of decrease was limited and BOFRF performed better than local RF and BOPPID in all scenarios.

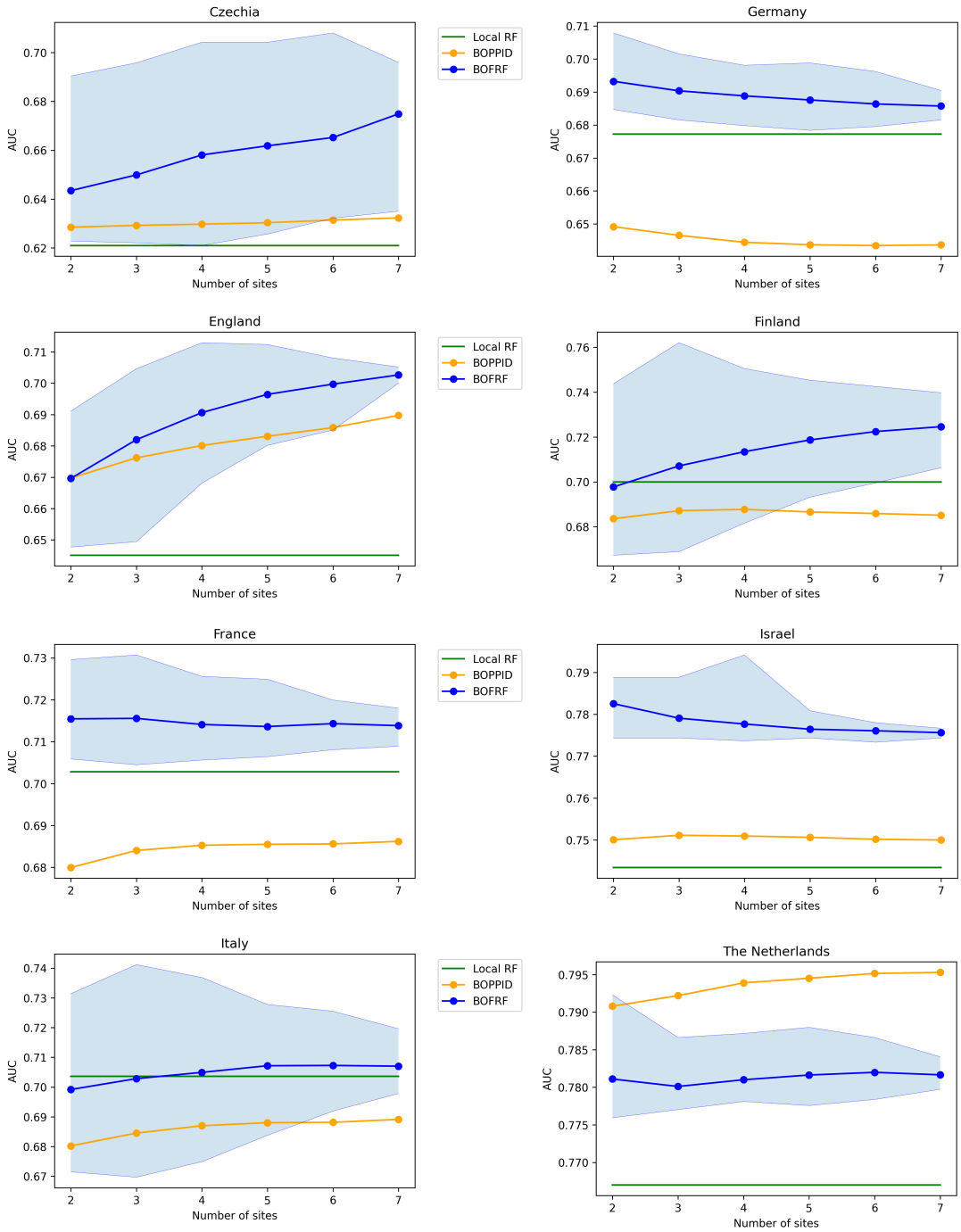

Figure 7.7: Overall AUC comparison of BOFRF with the baseline local RF model and BOPPID.

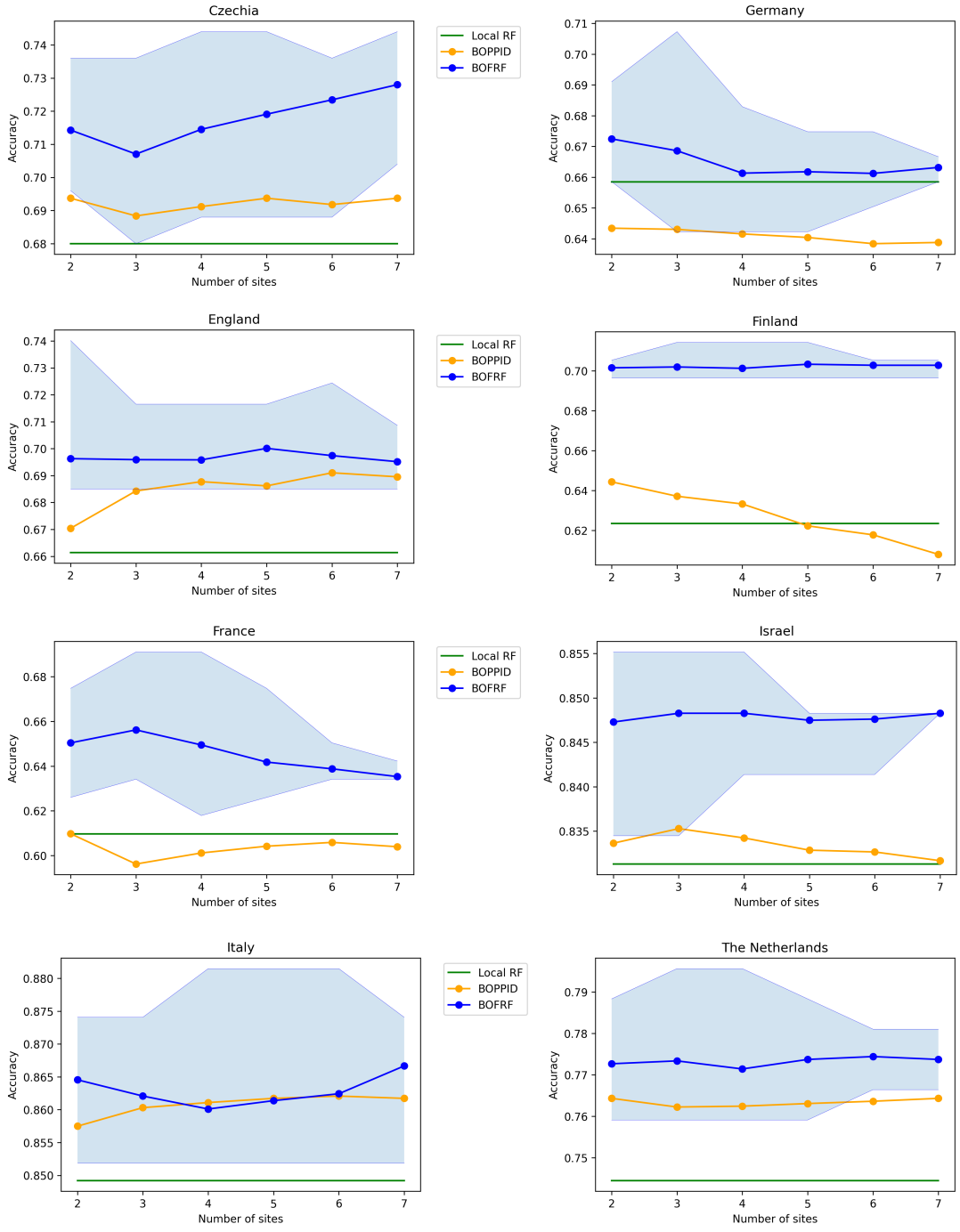

Figure 7.8: Overall accuracy comparison of BOFRF with the baseline local RF model and BOPPID.

7.3 Results of the Experiments for Clustered BOFRF

The experiments for evaluating the performance of the clustered BOFRF algorithm were conducted on the SHELTER dataset. Following the steps of the algorithm explained in Sec. 5.2, first, a characteristic vector was created for each country. These vectors are presented in Appendix C. Then, the clustered BOFRF algorithm was executed with different number of clusters and by using different distance metrics. In this section, the results of these experiments are presented. For the sake of traceability, the distance between each country calculated with the Manhattan and Eucledian distance metrics is listed in Tables 7.9 and 7.10, respectively.

Table 7.9: Manhattan Distance matrix on the SHELTER dataset.

	CZ	DE	EN	FI	FR	IL	IT	NL
CZ	-	1.64	1.81	2.23	2.10	1.72	1.83	1.67
DE	1.64	-	1.74	1.60	1.47	1.62	1.51	0.94
EN	1.81	1.74	-	2.17	1.13	1.47	1.76	1.61
FI	2.23	1.60	2.17	-	2.26	2.16	2.12	1.57
FR	2.10	1.47	1.13	2.26	-	1.80	1.64	1.64
IL	1.72	1.62	1.47	2.16	1.80	-	0.94	1.70
IT	1.83	1.51	1.76	2.12	1.64	0.94	-	1.84
NL	1.67	0.94	1.61	1.57	1.64	1.70	1.84	-

Table 7.10: Eucledian Distance matrix on the SHELTER dataset.

	CZ	DE	EN	FI	FR	IL	IT	NL
CZ	-	0.63	0.76	0.77	0.80	0.64	0.60	0.71
DE	0.63	-	0.64	0.59	0.54	0.51	0.48	0.30
EN	0.76	0.64	-	0.74	0.43	0.45	0.56	0.58
FI	0.77	0.59	0.74	-	0.72	0.83	0.78	0.56
FR	0.80	0.54	0.43	0.72	-	0.55	0.55	0.52
IL	0.64	0.51	0.45	0.83	0.55	-	0.28	0.57
IT	0.60	0.48	0.56	0.78	0.55	0.28	-	0.54
NL	0.71	0.30	0.58	0.56	0.52	0.57	0.54	-

7.3.1 Experiment with 2 clusters and Manhattan distance

In the first experiment covered in this section, the clustered BOFRF algorithm was executed with 2 clusters and by using Manhattan distance as the distance calculation metric. The algorithm randomly selected Czechia (CZ) as the initial centroid, hence the second centroid became Finland (FI), which is the farthest site to Czechia with a distance of 2.23 as shown in Table 7.9. Then, the remaining countries were assigned to the nearest centroid. As a result, the first cluster was formed as Czechia, England, France, Israel and Italy, and the second cluster was formed as Finland, Germany and the Netherlands. Later, the personalized BOFRF algorithm was run on these clusters separately, and the AUC values presented in Table 7.11 were obtained. As can be seen in the table, the AUC value of the clustered BOFRF was better than the AUC value of the base local RF model in all cases. In addition, when the AUC value was compared with the mean AUC value of the 247 environments set up in Sec. 7.2, it was observed that the AUC value of the clustered BOFRF was better than the mean AUC value in all cases.

Table 7.11: Achieved AUC values of countries with 2 clusters formed with Manhattan distance.

Site		Local RF (Baseline)	BOFRF (Mean)	**Clustered BOFRF**
Cluster 1	CZ	0.621	0.659	**0.676**
	EN	0.645	0.692	**0.703**
	FR	0.702	0.714	**0.719**
	IL	0.743	0.777	**0.777**
	IT	0.703	0.705	**0.720**
Cluster 2	DE	0.677	0.688	**0.697**
	FI	0.699	0.715	**0.762**
	NL	0.767	0.781	**0.781**

7.3.2 Experiment with 2 clusters and Eucledian distance

In the second experiment, the clustered BOFRF algorithm was again executed with 2 clusters, but this time Eucledian distance was used as the distance calculation metric. In this experiment, the randomly selected site, hence the first centroid, was Germany (DE), and the second centroid was England (EN) with a distance of 0.64 as shown in Table 7.10. After running the algorithm, the clusters presented in Table 7.12 were formed. As shown in the table, the clustered BOFRF model once again outperformed the local RF model in all cases. The AUC values obtained with the clustered BOFRF were no better than the mean AUC value in two of the cases, which are Czechia and Italy, but the result was still acceptable as it was an improvement over the local model.

Table 7.12: Achieved AUC values of countries with 2 clusters formed with Eucledian distance.

Site		Local RF (Baseline)	BOFRF (Mean)	**Clustered BOFRF**
Cluster 1	CZ	0.621	0.659	**0.647**
	DE	0.677	0.688	**0.689**
	FI	0.699	0.715	**0.743**
	IT	0.703	0.705	**0.703**
	NL	0.767	0.781	**0.783**
Cluster 2	EN	0.645	0.692	**0.688**
	FR	0.702	0.714	**0.722**
	IL	0.743	0.777	**0.779**

7.3.3 Experiment with 3 clusters and Eucledian distance

In the third experiment, the number of clusters was increased to 3 and the algorithm was forced to perform the same initialization presented in the previous section. The aim was to analyze whether adding one more cluster would be beneficial in further improving the prediction power of the sites. After repeating the clustering process of the algorithm once more, the cluster consisting of England, France and Israel (Cluster 2 in Table 7.12) remained the same, but the other cluster was split into two as shown in Table 7.13. The results showed that although the clustered BOFRF outperformed the local RF in 4 out of 5 cases, splitting the relevant cluster once again did not improve the prediction power of the sites in that cluster when compared to not splitting it, such that Czechia's AUC dropped from 0.647 to 0.633, Finland's AUC dropped from 0.743 to 0.732, Italy's AUC dropped from 0.703 to 0.689, and the Netherland's AUC dropped from 0.783 to 0.780. The only improvement was for Germany, where the AUC value increased from 0.689 to 0.690.

Table 7.13: Achieved AUC values of countries with 3 clusters formed with Eucledian distance.

Site		Local RF (Baseline)	BOFRF (Mean)	**Clustered BOFRF**
Cluster 1	CZ	0.621	0.659	**0.633**
	DE	0.677	0.688	**0.690**
	IT	0.703	0.705	**0.689**
Cluster 2	FI	0.699	0.715	**0.732**
	NL	0.767	0.781	**0.780**
Cluster 3	EN	0.645	0.692	**0.688**
	FR	0.702	0.714	**0.722**
	IL	0.743	0.777	**0.779**

CHAPTER 8

CONCLUSION

In this study, first, a novel federated ensemble classification algorithm for horizontally partitioned data called Boosting-based Federated Random Forest (BOFRF) is proposed, which not only increases the predictive power of all participating sites, but also provides significantly high improvements on the predictive power of sites having unsuccessful local models. The novelty of BOFRF lies in adapting the idea of boosting to random forest in a federated manner and introducing a new aggregation and weight calculation methodology in the integration phase. Second, to further improve the classification performance of participating sites, a personalized version of BOFRF is presented. The proposed BOFRF algorithm provides an adequate level of privacy in its base and personalized form as the actual data is not shared among the participants. However, in federated environments, a sneaky site may attempt to identify an individual in any of the datasets. To prevent this from happening, two privacy-preserving implementations of the proposed algorithm are provided. Lastly, an extension of BOFRF, namely clustered BOFRF, is proposed to cluster the participating sites according to their data distribution similarities or differences prior to running the algorithm.

To evaluate the effectiveness of BOFRF, a number of observational and statistical experiments are conducted on four healthcare datasets. The experiments show that BOFRF can successfully generate a powerful global model for all sites participating in the federated setting, regardless of whether they have well-performing or underperforming local models. The empirical results show that BOFRF consistently improves the prediction performance of local RF model, hence achieves the main objective of federated learning because BOFRF considers the performance of each deci-

sion tree across all sites with equal emphasis. When compared to one of the best-performing existing solutions, namely BOPPID, the most important advantage of BOFRF is that it provides significantly high prediction improvement for the sites whose local model is not performing well, suggesting that the algorithm also attained the expected impact. The particularly important feature of BOFRF that enables it to achieve this is its novel MCC-based weight calculation methodology, which takes both target classes (0 and 1) into account when calculating the weights of each decision tree. In the experiments, it was observed that the closer the AUC value of the local site is to 0.5, which is regarded as the failure limit, the more improvement BOFRF can provide given successful local models from other sites. This is important because in federated environments, enhancing the prediction capability of unsuccessful sites is much more important than the others in many cases. Thanks to BOFRF, these sites could benefit from the advantages of federated machine learning, that is, the ability to make accurate predictions despite the insufficient data they have in their repositories. Therefore, unless the usage of AdaBoost models is more convenient than RF models for some datasets (e.g., datasets producing high bias and low variance models), BOFRF would be a better choice than BOPPID because of the better prediction improvement it can provide for the sites, especially those with relatively unsuccessful local models.

In the literature, there are numerous solutions utilizing deep learning methods for federated learning [123, 124]. Although federated deep-learning approaches can produce better results than traditional federated learning methods in certain settings, they are computationally expensive, bring a communication overhead, and require a large amount of training data. Therefore, the implementation of federated solutions on traditional machine learning methods, including random forest, is still an important topic that has been widely studied by researchers. In this regard, the proposed solution is considered to be an essential contribution to state-of-the-art federated solutions.

Using decision trees as weak classifiers is a useful approach to follow in federated settings because decision trees have many advantages over other machine learning methods, such as the ability to handle categorical data and the ability to deal with outliers and noisy or missing data. Furthermore, although the participating sites mostly share the same set of features, they may still have some local features that are not present

in others. This is not an issue in random forest because the standard random forest algorithm already uses the idea of taking a random subset of features to build each decision tree to prevent overfitting; hence, the local features are not present in every decision tree generated at sites. However, this is valid up to a point because the more local features differ from each other, the more vertical they become, which may prevent the BOFRF algorithm from producing successful results. Therefore, it should be noted that this work is currently limited to horizontally partitioned data; hence, it may not be suitable for applications in vertical settings. Furthermore, the work presented in this study does not focus on improving the robustness of the proposed algorithm. Future enhancements could include the addition of a validation step, where the sites approve the retrieval of decision trees sent by other sites. In addition, the effect of applying feature selection techniques [125, 126] prior to running the actual algorithm could also be analyzed to handle heterogeneous data.

REFERENCES

[1] M. B. Malik, M. A. Ghazi, and R. Ali, "Privacy preserving data mining techniques: current scenario and future prospects," in *2012 third international conference on computer and communication technology*, pp. 26–32, IEEE, 2012.

[2] X.-D. Zhang, *A matrix algebra approach to artificial intelligence*. Springer, 2020.

[3] R. Nosowsky and T. J. Giordano, "The health insurance portability and accountability act of 1996 (HIPAA) privacy rule: implications for clinical research," *Annu. Rev. Med.*, vol. 57, pp. 575–590, 2006.

[4] "Regulation (EU) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation) (text with eea relevance)." https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679, Last accessed: 26.08.2022.

[5] V. Koutkias, "From data silos to standardized, linked, and fair data for pharmacovigilance: current advances and challenges with observational healthcare data," *Drug Safety*, vol. 42, no. 5, pp. 583–586, 2019.

[6] C. C. Aggarwal and P. S. Yu, "A general survey of privacy-preserving data mining models and algorithms," in *Privacy-preserving data mining*, pp. 11–52, Springer, 2008.

[7] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *2007 IEEE 23rd international conference on data engineering*, pp. 106–115, IEEE, 2006.

[8] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning,"

Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 13, no. 3, pp. 1–207, 2019.

[9] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[10] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint arXiv:1602.05629*, vol. 2, 2016.

[11] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pp. 1–11, 2019.

[12] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020.

[13] W. Zhang, T. Zhou, Q. Lu, X. Wang, C. Zhu, H. Sun, Z. Wang, S. K. Lo, and F.-Y. Wang, "Dynamic-fusion-based federated learning for COVID-19 detection," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15884–15891, 2021.

[14] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440*, 2020.

[15] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[16] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[17] I. Palit and C. K. Reddy, "Scalable and parallel boosting with mapreduce," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 10, pp. 1904–1916, 2011.

[18] Y. Li, C. Bai, and C. K. Reddy, "A distributed ensemble approach for mining healthcare data under privacy constraints," *Information sciences*, vol. 330, pp. 245–259, 2016.

[19] Z.-H. Zhou, "Ensemble learning," in *Machine learning*, pp. 181–210, Springer, 2021.

[20] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979.

[21] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[22] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, pp. 1–34, Springer, 2012.

[23] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[24] S. Zhang, Y. Chen, W. Zhang, and R. Feng, "A novel ensemble deep learning model with dynamic error correction and multi-objective ensemble pruning for time series forecasting," *Information Sciences*, vol. 544, pp. 427–445, 2021.

[25] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 31–40, 2016.

[26] M. Abdar, M. Zomorodi-Moghadam, X. Zhou, R. Gururajan, X. Tao, P. D. Barua, and R. Gururajan, "A new nested ensemble technique for automated diagnosis of breast cancer," *Pattern Recognition Letters*, vol. 132, pp. 123–131, 2020.

[27] T. Zhou, H. Lu, Z. Yang, S. Qiu, B. Huo, and Y. Dong, "The ensemble deep learning model for novel COVID-19 on CT images," *Applied soft computing*, vol. 98, p. 106885, 2021.

[28] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," tech. rep., Brown Univ Providence Ri Inst for Brain and Neural Systems, 1992.

[29] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[30] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[31] R. Hornung and M. N. Wright, "Block forests: random forests for blocks of clinical and omics covariate data," *BMC bioinformatics*, vol. 20, no. 1, pp. 1–17, 2019.

[32] A. Hajjem, F. Bellavance, and D. Larocque, "Mixed-effects random forest for clustered data," *Journal of Statistical Computation and Simulation*, vol. 84, no. 6, pp. 1313–1328, 2014.

[33] C. A. Field and A. H. Welsh, "Bootstrapping clustered data," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 3, pp. 369–390, 2007.

[34] M. Samanta and A. H. Welsh, "Bootstrapping for highly unbalanced clustered data," *Computational Statistics & Data Analysis*, vol. 59, pp. 70–81, 2013.

[35] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[36] P. Bühlmann and B. Yu, "Boosting," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 1, pp. 69–74, 2010.

[37] "What is the difference between bagging and boosting?." https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/, Last accessed: 08.10.2022.

[38] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[39] C. Ying, M. Qi-Guang, L. Jia-Chen, and G. Lin, "Advance and prospects of adaboost algorithm," *Acta Automatica Sinica*, vol. 39, no. 6, pp. 745–758, 2013.

[40] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[41] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[42] T. Chen and C. Guestrin, "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.

[43] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[44] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[45] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[46] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[47] L. T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[48] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "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.

[49] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, "Secure multi-party computation: theory, practice and applications," *Information Sciences*, vol. 476, pp. 357–372, 2019.

[50] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences*, vol. 552, pp. 183–200, 2021.

[51] Y. Liu, Z. Ma, Z. Yan, Z. Wang, X. Liu, and J. Ma, "Privacy-preserving federated k-means for proactive caching in next generation cellular networks," *Information Sciences*, vol. 521, pp. 14–31, 2020.

[52] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-IID data.," in *AAAI*, pp. 7865–7873, 2021.

[53] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE international conference on communications (ICC)*, pp. 1–7, IEEE, 2019.

[54] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[55] S. Gambs, B. Kégl, and E. Aïmeur, "Privacy-preserving boosting," *Data Mining and Knowledge Discovery*, vol. 14, no. 1, pp. 131–170, 2007.

[56] J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, "Privacy-preserving decision trees over vertically partitioned data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 2, no. 3, pp. 1–27, 2008.

[57] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data.," *Proc. Priv. Enhancing Technol.*, vol. 2017, no. 4, pp. 345–364, 2017.

[58] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption," *arXiv preprint arXiv:1711.10677*, 2017.

[59] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems*, vol. 36, no. 6, pp. 87–98, 2021.

[60] Y. Liu, Y. Liu, Z. Liu, Y. Liang, C. Meng, J. Zhang, and Y. Zheng, "Federated forest," *IEEE Transactions on Big Data*, 2020.

[61] N. Ge, G. Li, L. Zhang, and Y. Liu, "Failure prediction in production line based on federated learning: an empirical study," *Journal of Intelligent Manufacturing*, pp. 1–18, 2021.

[62] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[63] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[64] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1–19, 2021.

[65] Q. Wang and Y. Zhou, "FedSPL: federated self-paced learning for privacy-preserving disease diagnosis," *Briefings in Bioinformatics*, vol. 23, no. 1, p. bbab498, 2022.

[66] O. Choudhury, Y. Park, T. Salonidis, A. Gkoulalas-Divanis, I. Sylla, *et al.*, "Predicting adverse drug reactions on distributed health data using federated learning," in *AMIA Annual symposium proceedings*, vol. 2019, p. 313, American Medical Informatics Association, 2019.

[67] Q. Dou, T. Y. So, M. Jiang, Q. Liu, V. Vardhanabhuti, G. Kaissis, Z. Li, W. Si, H. H. Lee, K. Yu, *et al.*, "Federated deep learning for detecting COVID-19 lung abnormalities in CT: a privacy-preserving multinational validation study," *NPJ digital medicine*, vol. 4, no. 1, pp. 1–11, 2021.

[68] M. Abdul Salam, S. Taha, and M. Ramadan, "COVID-19 detection using federated machine learning," *PLoS One*, vol. 16, no. 6, p. e0252573, 2021.

[69] A. Vaid, S. K. Jaladanki, J. Xu, S. Teng, A. Kumar, S. Lee, S. Somani, I. Paranjpe, J. K. De Freitas, T. Wanyan, *et al.*, "Federated learning of electronic health records improves mortality prediction in patients hospitalized with covid-19," *medRxiv*, 2020.

[70] C. Alvarez-Romero, A. Martínez-García, A. A. Sinaci, M. Gencturk, E. Méndez, T. Hernández-Pérez, R. Liperoti, C. Angioletti, M. Löbe, N. Ganapathy, *et al.*, "Fair4health: Findable, accessible, interoperable and reusable data to foster health research," *Open Research Europe*, vol. 2, no. 34, p. 34, 2022.

[71] C. Alvarez-Romero, A. Martinez-Garcia, J. T. Vega, P. Díaz-Jimènez, C. Jimènez-Juan, M. D. Nieto-Martín, E. R. Villarán, T. Kovacevic, D. Bokan, S. Hromis, *et al.*, "Predicting 30-day readmission risk for patients with chronic obstructive pulmonary disease through a federated machine learning architecture on findable, accessible, interoperable, and reusable (FAIR) data: Development and validation study," *JMIR Medical Informatics*, vol. 10, no. 6, p. e35307, 2022.

[72] J. Carmona-Pírez, B. Poblador-Plou, A. Poncel-Falcó, J. Rochat, C. Alvarez-Romero, A. Martínez-García, C. Angioletti, M. Almada, M. Gencturk, A. A. Sinaci, *et al.*, "Applying the FAIR4Health solution to identify multimorbidity patterns and their association with mortality through a frequent pattern growth association algorithm," *International Journal of Environmental Research and Public Health*, vol. 19, no. 4, p. 2040, 2022.

[73] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[74] Y. Liu, J. James, J. Kang, D. Niyato, and S. Zhang, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7751–7763, 2020.

[75] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad, and M. Bennis, "Federated learning in the sky: Joint power allocation and scheduling with uav swarms," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2020.

[76] H. Cao, S. Liu, R. Zhao, and X. Xiong, "Ifed: A novel federated learning framework for local differential privacy in power internet of things," *International Journal of Distributed Sensor Networks*, vol. 16, no. 5, p. 1550147720919698, 2020.

[77] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 4, pp. 2964–2973, 2020.

[78] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[79] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[80] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[81] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *arXiv preprint arXiv:2003.13461*, 2020.

[82] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning: A meta-learning approach," *arXiv preprint arXiv:2002.07948*, 2020.

[83] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, pp. 1126–1135, PMLR, 2017.

[84] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, and D. Ramage, "Federated evaluation of on-device personalization," *arXiv preprint arXiv:1910.10252*, 2019.

[85] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[86] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," *Advances in neural information processing systems*, vol. 30, 2017.

[87] F. Hanzely and P. Richtárik, "Federated learning of a mixture of global and local models," *arXiv preprint arXiv:2002.05516*, 2020.

[88] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21394–21405, 2020.

[89] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized federated learning: An attentive collaboration approach," 2020.

[90] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[91] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 8, pp. 3710–3722, 2020.

[92] M. Duan, D. Liu, X. Ji, Y. Wu, L. Liang, X. Chen, Y. Tan, and A. Ren, "Flexible clustered federated learning for client-level data distribution shift," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[93] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2020.

[94] L. Yu, W. Nie, L. Xin, and M. Guo, "Clustered federated learning based on data distribution," in *2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*, pp. 1–5, 2021.

[95] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19586–19597, 2020.

[96] M. Gencturk, A. A. Sinaci, and N. K. Cicekli, "BOFRF: A novel boosting-based federated random forest algorithm on horizontally partitioned data," *IEEE Access*, vol. 10, pp. 89835–89851, 2022.

[97] T. Bruckhaus, "The business impact of predictive analytics," in *Knowledge discovery and data mining: Challenges and realities*, pp. 114–138, Igi Global, 2007.

[98] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning*, pp. 233–240, 2006.

[99] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[100] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[101] S. Boughorbel, F. Jarray, and M. El-Anbari, "Optimal classifier for imbalanced data using matthews correlation coefficient metric," *PloS one*, vol. 12, no. 6, p. e0177678, 2017.

[102] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[103] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," *Machine learning*, vol. 37, no. 3, pp. 297–336, 1999.

[104] C. Adams, "Trusted third party," in *Encyclopedia of Cryptography and Security*, pp. 1335–1335, Springer, 2011.

[105] H.-Y. Chien, "Dynamic public key certificates with forward secrecy," *Electronics*, vol. 10, no. 16, p. 2009, 2021.

[106] A. Gurevich and E. Gudes, "Privacy preserving data mining algorithms without the use of secure computation or perturbation," in *2006 10th International*

Database Engineering and Applications Symposium (IDEAS'06), pp. 121–128, IEEE, 2006.

[107] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM Sigkdd Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[108] S. Kumar, "Understanding k-means, k-means++ and, k-medoids clustering algorithms," 2020. https://towardsdatascience.com/understanding-k-means-k-means-and-k-medoids-clustering-algorithms-ad9c9fbf47ca.

[109] M. Ali, "Implementing k-means clustering with k-means++ initialization in python," 2021. https://medium.com/geekculture/implementing-k-means-clustering-with-k-means-initialization-in-python-7ca5a859d63a.

[110] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," 2006.

[111] J. Brownlee, "Distance measures for machine learning," 2020. https://machinelearningmastery.com/distance-measures-for-machine-learning.

[112] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*, pp. 420–434, Springer, 2001.

[113] J. W. Smith, J. E. Everhart, W. Dickson, W. C. Knowler, and R. S. Johannes, "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus," in *Proceedings of the annual symposium on computer application in medical care*, p. 261, American Medical Informatics Association, 1988.

[114] "Pima indians diabetes database." https://www.kaggle.com/uciml/pima-indians-diabetes-database, Last accessed: 26.08.2022.

[115] B. Antal and A. Hajdu, "An ensemble-based system for automatic screening of diabetic retinopathy," *Knowledge-based systems*, vol. 60, pp. 20–27, 2014.

[116] D. Dua and C. Graff, "UCI machine learning repository," 2019. http://archive.ics.uci.edu/ml.

[117] T. Taveira-Gomes, "Machine learning on the diabetic retinopathy debrecen data set data set," 2016. https://rpubs.com/tiagotaveira/debrecen.

[118] C. Bartley, "Replication data for: South african heart disease," 2016.

[119] "SHELTER: Services and health for elderly in long term care." https://cordis.europa.eu/project/id/223115, Last accessed: 20.10.2022.

[120] G. Onder, I. Carpenter, H. Finne-Soveri, J. Gindin, D. Frijters, J. C. Henrard, T. Nikolaus, E. Topinkova, M. Tosato, R. Liperoti, *et al.*, "Assessment of nursing home residents in europe: the services and health for elderly in long term care (SHELTER) study," *BMC health services research*, vol. 12, no. 1, pp. 1–10, 2012.

[121] G. Onder, R. Liperoti, D. Fialova, E. Topinkova, M. Tosato, P. Danese, P. F. Gallo, I. Carpenter, H. Finne-Soveri, J. Gindin, *et al.*, "Polypharmacy in nursing home in europe: results from the SHELTER study," *Journals of Gerontology Series A: Biomedical Sciences and Medical Sciences*, vol. 67, no. 6, pp. 698–704, 2012.

[122] O. C. Ibe, "Chapter 9 - introduction to inferential statistics," in *Fundamentals of applied probability and random processes*, pp. 275–305, Boston: Academic Press, second edition ed., 2014.

[123] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[124] M. Hao, H. Li, G. Xu, S. Liu, and H. Yang, "Towards efficient and privacy-preserving federated deep learning," in *ICC 2019-2019 IEEE international conference on communications (ICC)*, pp. 1–6, IEEE, 2019.

[125] R. Sheikhpour, M. A. Sarram, S. Gharaghani, and M. A. Z. Chahooki, "A robust graph-based semi-supervised sparse feature selection method," *Information Sciences*, vol. 531, pp. 13–30, 2020.

[126] L. Chamakura and G. Saha, "An instance voting approach to feature selection," *Information Sciences*, vol. 504, pp. 449–469, 2019.

APPENDIX A

HYPERPARAMETER VALUES IN THE OBSERVATIONAL EXPERIMENTS

Table A.1: Hyperparameter values giving the best result in the observational experiments.

Environment #	Site	n_estimators	min_samples_split	min_samples_leaf	max_features	max_depth	criterion	threshold	ensemble_strategy
E1	S1	100	8	2	7	14	gini	0.2	global
	S2	50	4	2	7	14	IG	0.2	global
E2	S1	100	8	2	7	14	gini	0.15	global
	S2	50	4	2	3	14	gini	0.25	global
	S3	150	6	3	7	15	IG	0.25	global
E3	S1	50	6	2	5	14	IG	0.1	global
	S2	50	8	2	5	14	IG	0.3	global
	S3	150	8	3	5	13	IG	0.3	global
E4	S1	150	6	3	7	15	IG	0	global
	S2	50	4	2	3	14	gini	0	global
	S3	50	4	2	7	14	IG	0.1	global
	S4	150	6	3	7	15	IG	0.2	global

Continued on next page

Table A.1: Hyperparameter values giving the best result in the observational experiments. (Continued)

E5	S1	50	6	2	5	14	IG	0.2	global
	S2	150	6	3	7	15	IG	0.3	global
	S3	50	8	2	3	13	gini	0.2	local
	S4	50	6	2	5	15	IG	0.3	global
E6	S1	50	8	2	3	13	gini	0.3	global
	S2	50	4	2	7	14	IG	0.4	global
	S3	150	6	3	7	15	IG	0.25	local
	S4	50	4	2	3	14	gini	0.4	global
E7	S1	150	7	14	auto	none	IG	0	global
	S2	150	3	13	auto	none	IG	0.4	global
E8	S1	100	3	15	auto	none	IG	0.2	global
	S2	100	3	15	auto	none	IG	0	global
	S3	150	5	14	auto	none	IG	0	global
E9	S1	150	7	14	auto	none	IG	0.3	global
	S2	100	5	14	auto	none	IG	0.3	global
	S3	100	5	14	auto	none	IG	0.3	global
E10	S1	50	6	2	5	5	gini	0.3	global
	S2	50	6	2	5	5	gini	0	global
E11	S1	150	2	3	7	7	IG	0.4	global
	S2	150	3	3	3	9	IG	0.4	local
	S3	100	3	2	9	7	gini	0.4	global
E12	S1	30	6	3	7	7	gini	0.2	global
	S2	30	2	3	3	9	gini	0.2	global
	S3	50	4	2	7	7	IG	0.1	local

APPENDIX B

Z SCORES OF THE SITES IN THE SHELTER DATASET

Table B.1: *Z-score*s of countries for AUC and accuracy in federated environments built on the SHELTER dataset.

Site	*Z-score* for AUC	*Z-score* for accuracy
Czechia (CZ)	10.72	14.11
Germany (DE)	22.44	14.70
England (EN)	7.22	7.51
Finland (FI)	14.51	16.55
France (FR)	50.54	24.63
Israel (IL)	52.50	27.85
Italy (IT)	10.58	0.75
The Netherlands (NL)	-16.82	6.22

APPENDIX C

CHARACTERISTIC VECTORS IN THE SHELTER DATASET

Table C.1: Characteristic vector of the countries in the SHELTER dataset.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
CZ	.72	.68	.59	.57	.82	.53	.82	.60	.31	.87	.92	.72	.63	.86	.66
DE	.79	.45	.75	.70	.81	.61	.83	.64	.79	.70	.91	.81	.75	.90	.65
EN	.72	.74	.38	.64	.87	.62	.87	.93	.88	.93	.93	.75	.86	.88	.65
FI	.75	.41	.72	.29	.82	.54	.90	.75	.74	.87	.96	.91	.85	.93	.33
FR	.76	.68	.53	.71	.58	.58	.84	.94	.88	.70	.92	.80	.89	.83	.57
IL	.71	.66	.50	.77	.78	.84	.96	.75	.75	.87	.95	.76	.70	.93	.77
IT	.73	.59	.57	.72	.62	.79	.92	.68	.67	.91	.90	.80	.81	.94	.84
NL	.67	.47	.77	.53	.78	.56	.81	.74	.91	.75	.94	.72	.79	.87	.71

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Gençtürk, Mert
Nationality: Turkish (TC)

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Computer Engineering Department, METU	2015
B.S.	Computer Engineering Department, METU	2012
High School	Kastamonu Göl Anadolu Öğretmen Lisesi	2007

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2018-Present	SRDC Software Research & Development and Consultancy Corp.	Technical Manager & Researcher
2015-2018	SRDC Software Research & Development and Consultancy Ltd.	Senior Software Engineer & Researcher
2012-2015	SRDC Software Research & Development and Consultancy Ltd..	Software Engineer
2010-2012	SRDC Software Research & Development and Consultancy Ltd.	Part-time Software Developer

PUBLICATIONS

Journal Publications

1. **M. Gencturk**, A. A. Sinaci, N. K. Cicekli, "BOFRF: A novel boosting-based federated random forest algorithm on horizontally partitioned data," *IEEE Access*, vol. 10, pp. 89835-89851, 2022. DOI: 10.1109/ACCESS.2022.3202008

2. C. Alvarez-Romero, A. Martinez-Garcia, J. T. Vega, P. Díaz-Jiménez, C. Jiménez-Juan, M. D. Nieto-Martín, ..., **M. Gencturk**, A. A. Sinaci, M. Ollero-Baturone, C. L. Parra-Calderón, "Predicting 30-day readmission risk for patients with chronic obstructive pulmonary disease through a federated machine learning architecture on findable, accessible, interoperable, and reusable (FAIR) data: Development and validation study," *JMIR Medical Informatics*, vol. 10, no. 6, p. e35307, 2022. DOI: 10.2196/35307

3. J. Carmona-Pírez, B. Poblador-Plou, A. Poncel-Falcó, J. Rochat, C. Alvarez-Romero, A. Martínez-García, C. Angioletti, M. Almada, **M. Gencturk**, A. A. Sinaci, *et al.*, "Applying the FAIR4Health solution to identify multimorbidity patterns and their association with mortality through a frequent pattern growth association algorithm," *International Journal of Environmental Research and Public Health*, vol. 19, no. 4, p. 2040, 2022. DOI: 10.3390/ijerph19042040

4. C. Alvarez-Romero, A. Martínez-García, A. A. Sinaci, **M. Gencturk**, E. Mendéz, T. Hernández-Pérez, R. Liperoti, C. Angioletti, M. Löbe, N. Ganapathy, *et al.*, "FAIR4Health: Findable, Accessible, Interoperable and Reusable data to foster Health Research," *Open Research Europe*, vol. 2, no. 34, p. 34, 2022. DOI: 10.12688/openreseurope.14349.2

5. A. A. Sinaci, F. J. Núñez-Benjumea, **M. Gencturk**, J. Malte-Levin, T. Deserno, C. Chronaki, G. Cangioli, C. Cavero-Barca, J. M. Rodríguez-Pérez, M. M. Pérez-Pérez, *et al.*, "From raw data to FAIR data: the FAIRification workflow for health research," *Methods of Information in Medicine*, vol. 59, no. S 01, pp. e21-e32, 2020. DOI: 10.1055/s-0040-1713684

International Conference Publications

1. **M. Gencturk**, G. B. L. Erturkmen, H. Gappa, W. Schmidt-Barzynski, A. Steinhoff, P. Abizanda, T. Robbins, O. Pournik, B. Ahmad, H. Randeva, *et al.*, "The design of a mobile platform providing personalized assistance to older multimorbid patients with mild dementia or mild cognitive impairment (MCI)," in *10th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2022)*, 2022, pp. 1-7. DOI: 10.1145/3563137.3563138

2. G. B. L. Erturkmen, M. Baskaya, B. Sarigul, M. Yuksel, T. Namli, S. Gonul, G. Yilmaz, **M. Gencturk**, J. Bloemeke, T. N. Arvanitis, *et al.*, "Enabling Patient Adherence via Personalised, Just-in Time Adaptive Interventions in ADLIFE Architecture," in *10th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2022)*, 2022, pp. 1-6. DOI: 10.1145/3563137.3563151

3. O. Pournik, B. Ahmad, G. Despotou, S. N. L. C. Keung, H. Muir, Y. Mohamad, H. Gappa,G. B. L. Erturkmen, M. Yuksel, **M. Gencturk**, *et al.*, "CAREPATH methodology for development of computer interpretable, integrated clinical guidelines," in *10th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2022)*, 2022, pp. 1-6. DOI: 10.1145/3563137.3563155

4. H. Gappa, Y. Mohamad, M. Breidenbach, ..., **M. Gencturk**, M. Yuksel, J. Ayadi, L. Gilardi, A. Consoli, L. Ferrazzini, C. A. Velasco, "Making Person-Centred Health Care Beneficial for People with Mild Cognitive Impairment (MCI) or Mild Dementia–Results of Interviews with Patients and Their Informal Caregivers," in *International Conference on Computers Helping People with Special Needs*, 2022, pp. 468-474. DOI: 10.1007/978-3-031-08648-9_54

5. M. Perbix, M. Löbe, S. Stäubert, A. A. Sinaci, **M. Gencturk**, M. Quintero, A. Martinez-Garcia, C.a Alvarez-Romero, C. L. Parra-Calderon, A. Winter, "A Formal Model for the FAIR4Health Information Architecture," in *Advances in Informatics, Management and Technology in Healthcare*, 2022, pp. 446-449. DOI: 10.3233/SHTI220761

6. O. Pournik, B. Ahmad, S. N. L. C. Keung, O. Khan, G. Despotou, A. Consoli, J. Ayadi, L. Gilardi, G. B. L. Erturkmen, M. Yuksel, **M. Gencturk**, *et al.*, "CAREPATH: Developing Digital Integrated Care Solutions for Multimorbid Patients with Dementia," in *Advances in Informatics, Management and Technology in Healthcare*, 2022, pp. 487-490. DOI: 10.3233/SHTI220771

7. **M. Gencturk**, A. Teoman, C. Alvarez-Romero, A. Martinez-Garcia, C. L. Parra-Calderon, B. Poblador-Plou, N. Löbe, A. A. Sinaci, "End User Evaluation of the FAIR4Health Data Curation Tool," in *Public Health and Informatics*, 2021, pp. 8-12. DOI: 10.3233/SHTI210110

8. C. Alvarez-Romero, A. Martinez-Garcia, **M. Gencturk**, A. A. Sinaci, C. L. Parra-Calderon, "FAIR4Health FAIRification Tools," in *16th International Digital Curation Conference*, Apr. 2021.

9. A. A. Sinaci, T. Namli, S. Postaci, **M. Gencturk**, G. B. L. Erturkmen, "Transforming Health Data into Timeseries for Predictive Analytics," in *11th Congress of Medical Informatics (TurkMIA)*, Nov. 2018.

10. **M. Gencturk**, M. Sahin, E. Simsek, Y. Kabak, "Certification in Electronic Emergency Management," in *Recent Trends in Control and Sensor Systems in Emergency Management*, 2018, pp. 84-94. DOI: 10.1007/978-3-319-70452-4_8

11. G. Schimak, D. Havlik, P. Kutschera, R. Duro, **M. Gencturk**, "Emergency Maps Tool as a Collaborative Instrument for Decision Makers in a Command and Control Environment," in *Recent Trends in Control and Sensor Systems in Emergency Management*, 2018, pp. 1-13. DOI: 10.1007/978-3-319-70452-4_1

12. R. Duro, **M. Gencturk**, G. Schimak, P. Kutschera, D. Havlik, K. Kutschera, "Framework for Enabling Technical and Organizational Interoperability in the Management of Environmental Crises and Disasters," in *International Symposium on Environmental Software Systems*, 2017, pp. 290-301. DOI: 10.1007/978-3-319-89935-0_24

13. **M. Gencturk**, E. Evci, A. Guney, Y. Kabak, G. B. L. Erturkmen, "Achieving Semantic Interoperability in Emergency Management Domain," in *Inter-*

national Symposium on Environmental Software Systems, 2017, pp. 279-289. DOI: 10.1007/978-3-319-89935-0_23

14. **M. Gencturk**, M. Redaelli, Y. Kabak, G. B. L. Erturkmen, R. Arisi, L. Toscano, "Automating the Development and Implementation of Interoperability Profiles," in *8th International Conference on Interoperability for Enterprise Systems and Applications* , Mar. 2016.

15. **M. Gencturk**, R. Duro, Y. Kabak, B. Bozic, K. Kahveci, B. Yilmaz, "Interoperability Profiles for Disaster Management and Maritime Surveillance," in *eChallenges e-2015 Conference*, 2015, pp. 1-9.
DOI: 10.1109/eCHALLENGES.2015.7441073

16. **M. Gencturk**, O. Eren, M. Yuksel, "Evaluation Study for Self Management of Patients with Ankylosing Spondylitis (AS) through a Personal Health System," in *The European League Against Rheumatism (EULAR) Congress*, Jun. 2015.

17. **M. Gencturk**, R. Arisi, L. Toscano, Y. Kabak, M. Di Ciano, A. Palmitessa, "Profiling Approach for the Interoperability of Command & Control Systems with Sensing Systems in Emergency Management," in *6th International IFIP Working Conference on Enterprise Interoperability*, May 2015.

18. B. Bozic, **M. Gencturk**, R. Duro, Y. Kabak, G. Schimak, "Requirements Engineering for Semantic Sensors in Crisis and Disaster Management," in *International Symposium on Environmental Software Systems*, 2015, pp. 397-406. DOI: 10.1007/978-3-319-15994-2_40

19. **M. Gencturk**, E. Alpay, G. B. L. Erturkmen, A. Dogac, H. Aydin, "Self-Management of Patients with Severe Arthritis through a Personal Health System: the Turkish Case Study in the PALANTE Project," in *eChallenges e-2014 Conference*, 2014, pp. 1-7.

20. T. Namli, S. Postaci, **M. Gencturk**, A. Dogac, A. Yalcinkaya, C. Taskin, "Addressing the Adoptability Challenges of the PHR Systems: SharingCare," in *eChallenges e-2013 Conference*, Oct. 2013.

21. T. Namli, S. Postaci, **M. Gencturk**, A. Dogac, A. Yalcinkaya, I. Kiremitci, C. Taskin, E. Erkel, "A Personal Health Ecosystem: SharingCare," in *Med-e-Tel*

Conference, Apr. 2013.

22. S. Postaci, **M. Gencturk**, R. V. Basar, A. Yilmaz, G. B. L. Erturkmen, M. Yuksel, A. Dogac, A. Yalcinkaya, T. Namli, C. O. Etemoglu, "mePHR: A Mobile Personal Health System (PHS) Framework," in *eChallenges e-2012 Conference*, Oct. 2012.