Federated Learning with Adaptive Deep Rule-Based Classifier

By

Dipu Saha
ID: 012211072

Submitted in partial fulfilment of the requirements
of the degree of Master of Science in Computer Science and Engineering

July 14, 2024

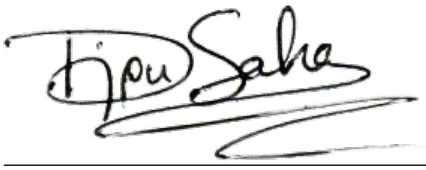
Department of Computer Science and Engineering
United International University

Declaration

I, **Dipu Saha**, declare that this thesis titled **Federated Learning With Adaptive Deep Rule-Based Classifier** and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a M.Sc. degree at United International University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all the main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made it clear exactly what was done by others and what I have contributed myself.

Signed by:



(Dipu Saha)

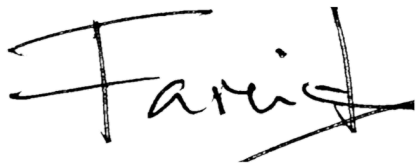
Date: July 14, 2024

Certificate

I do hereby declare that the research work embodied in this thesis, titled **Federated Learning With Adaptive Deep Rule-Based Classifier**, is the outcome of an original work carried out by **Dipu Saha** under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of M.Sc. in Computer Science & Engineering.

Signed by:



(Prof. Dr. Dewan Md. Farid)
Department of Computer Science and Engineering,
United International University,
United City, Madani Ave, Dhaka 1212, Bangladesh.

Date: July 14, 2024

Abstract

In the past, data processing required the transportation of data to computational resources. However, the current paradigm entails the deployment of computational resources to the data's location. Federated Learning, a specialized domain within machine learning, employs a decentralized methodology for model training. Rather than centralizing the training process on a single server, this method allows for the local training of classifiers on individual devices, with model adjustments or parameter updates being disseminated to a central repository. This manuscript outlines two distinctive methodologies that are intended for implementation within Federated Learning frameworks. The first methodology integrates a scalable Decision Tree algorithm within a Federated Learning context to introduce a privacy-conscious mining strategy for large personalized data. This approach employs a Decision Tree classifier within the RainForest framework, guaranteeing that the central server does not receive any personal information. Each participating device trains the Decision Tree classifier locally and transmits their derived probability values as parameters to the central server. Using five benchmark datasets, we assessed the efficacy of this methodology, which demonstrated exceptional performance. The following methodology involves the integration of traditional rule-based classifiers into a Federated Learning environment. This method, which is based on supervised learning, employs a set of IF-THEN rules that are derived by a Decision Tree classifier. This method generates classification rules by utilizing local datasets on client devices. The rules are then transmitted to the central server for examination and integration. The iterative process enables the continuous refinement and adaptation of classification rules in response to the evolution of concepts. The efficacy of this method was validated through experimentation with ten benchmark datasets, which illustrated remarkable performance.

Acknowledgements

I would like to express my sincere gratitude to all those who have contributed to the completion of this thesis. First and foremost, I am deeply indebted to my supervisor, Professor Dr. Dewan Md. Farid, for his invaluable guidance, support, and encouragement throughout the entirety of this research endeavor. His expertise, patience, and unwavering commitment have been instrumental in shaping this work. I extend my heartfelt appreciation to the members of my thesis committee, Professor Dr. Chowdhury Mofizur Rahman, Professor Dr. Swakkhar Shatabda, and Professor Dr. Mohammad Nurul Huda, for their insightful feedback and constructive criticism, which significantly enriched the quality of this thesis. I am grateful to the United International University for fostering an intellectually stimulating environment conducive to academic growth. My deepest appreciation goes to my family for their unconditional love, support, and understanding throughout this journey. Their encouragement has been a constant source of motivation, inspiring me to persevere even in the face of challenges. Last but not least, I am indebted to Mr. Mainul Karim and Mr. Niloy Kumar Kundu for their encouragement, camaraderie, and assistance, which have made this academic pursuit all the more rewarding. To all those mentioned above and to the countless others who have contributed in various ways, I extend my sincerest gratitude. Your support has been invaluable, and this thesis would not have been possible without you.

Thank you.

Publication List

The main contributions of this research are either published or accepted or in preparation in journals and conferences as mentioned in the following list:

Conference Papers

1. Dipu Saha, Mainul Karim, Suriya Phongmoo, and Dewan Md Farid. A privacy-preserving approach for big data mining using rainforest with federated learning. In 2023 IEEE Region 10 Technical Conference (TENCON), pages 59–64. IEEE, 2023.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Machine learning (ML), as a subset of artificial intelligence (AI), is playing a pivotal role in driving significant advancements across diverse technological domains by empowering systems to autonomously learn, adapt, and optimize without the need for explicit programming. This inherent capacity for self-evolution is founded on sophisticated algorithms capable of meticulously analyzing vast datasets, discerning patterns, and making accurate predictions. Whether it is recognizing speech, enhancing image quality, optimizing logistics, or personalizing user experiences, machine learning models reduce the necessity for human intervention and manual adjustments. Moreover, machine learning exhibits remarkable efficacy and precision in diverse applications, including optimizing strategies, early detection of diseases, identification of potential fraudulent transactions, and personalized user experiences [1]. Continuously refining its outputs through adaptation to new data, machine learning catalyzes innovation and boosts efficiency across a myriad of domains. The synergistic blend of scalability and adaptability offered by machine learning has not only revolutionized existing technologies but has also paved the way for novel paradigms and comprehensive solutions, thus providing the groundwork for the development of sophisticated and integrated technological systems. Nevertheless, the advancement of this technology has been accompanied by a series of challenges that impede its further expansion [2].

In the conventional machine learning process, identifying the problem to be addressed is followed by the task of curating the necessary data for model training. Ideally, the dataset should encompass a comprehensive and diverse representation of real-world data to ensure the model's efficacy. However, the current era of big data presents a complex landscape, wherein data volumes continue to experience exponential growth while the diversity and intricacy of data types expand concomitantly. The centralized storage of large-scale datasets, which can reach magnitudes of terabytes, petabytes, or even exabytes, gives rise to a myriad of significant challenges and limitations [3]. Primarily, managing such colossal volumes of data becomes a formidable task for a singular system as it surpasses the

bounds of efficient storage and processing capabilities [4]. The endeavor of accommodating such vast data quantities necessitates considerable investment in expensive hardware and infrastructure to bear the substantial workload. Even with these measures in place, when clients routinely contact the server to complete the forecasting process, the user experience may be adversely affected due to issues concerning accessibility, network latency, power consumption, and other unforeseen concerns [5]. Furthermore, the collection and transmission of user data to a centralized server for processing using conventional learning algorithms gives rise to various critical privacy and security concerns. The concentration of user data in a central repository renders the server susceptible to potential hacking attempts. Should such a breach be successful, it could lead to the exposure of sensitive information, consequently resulting in identity theft, financial fraud, or other malicious activities [5]. Moreover, the vast availability of user data raises the risk of constructing detailed user profiles without their explicit consent, leading to exploitative uses such as targeted advertising or discriminatory practices. In certain instances, the central server might share data with third-party entities without obtaining the necessary user consent, leading to unintended and undisclosed purposes for the data. High-profile data breaches and incidents of data misuse, like the Cambridge Analytics scandal on Facebook [6], exposed the vulnerabilities of personal data and highlighted the need for more robust data protection measures. To address this issue, numerous governments have enacted specific data protection regulations, such as GDPR [7], HIPAA [8], CCPA [9], PDPA [10], PIPEDA [11], LGPD [12], and PDPL [13], tailored to their respective countries or regions and subject to periodic updates. These regulations are meticulously designed to establish a comprehensive framework governing the collection, processing, storage, and sharing of personal information while simultaneously safeguarding individuals' rights and privacy. Consequently, organizations are mandated to demonstrate compliance with these data protection regulations, maintain comprehensive records of their data processing activities, and establish procedures to effectively respond to data breaches and privacy violations. Nonetheless, adhering to these regulations has resulted in certain organizations grappling with data scarcity, ultimately hindering their ability to ensure the desired accuracy of their models.

Federated Learning (FL) emerged as a novel concept delineated in a research paper [14] unveiled by Google researchers. This cutting-edge technology within the domain of machine learning was heralded with the ambition to eradicate pervasive issues concerning data protection and governance traditionally encountered within conventional machine learning paradigms. The cornerstone of this innovation lay in the principle of decentralized learning, thereby precluding the need to share client data with a centralized server. Rather than a transfer of raw data, the server dispatches an initial model to individual client devices, thus allowing for localized training with data retained solely on the clients' terminals. The nuanced approach involves the clients transmitting solely the encrypted gradients of the locally trained model to the server, thereby engendering a new global model through meticulous processes of aggregation or pruning, as the circumstances dic-

tate. This process iterates repeatedly until the achievement of either a predetermined performance threshold or the exhaustion of the allocated communication rounds.

In a further refinement of this privacy-centered approach, a secure aggregation mechanism was crafted. It functions through a buddy system where the server couples clients and introduces random values to the data prior to forwarding it to the central server. This enhancement in anonymity is maintained by the server, as it is precisely aware of the values transmitted to the paired clients and can nullify them to extract the actual content. The effect of this strategy is to mask the various parameters, such as weights and biases, during their conveyance to the server. The inherent decentralization within Federated Learning's framework has not only revolutionized data protection but has also conferred a plethora of indirect advantages. Corporate entities are liberated from anxieties concerning data warehousing, given the obviation of the necessity for centralized storage. The reduced dependency on costly centralized hardware infrastructure contributes to significant savings, avoiding the burdens of managing extensive server farms by utilizing the limited resources of local devices [15].

Furthermore, Federated Learning's design, which prevents the need to send the whole data or complete model to the server [16], empowers the client to function within limited network bandwidth. Even in the unfortunate event of the main server being compromised, the confidentiality of client data remains largely impervious due to the limited information transmitted. In critical scenarios, such as fraud detection requiring real-time updates [17], local devices can easily facilitate this process through Federated Learning implementation. Moreover, Federated Learning exhibits resilience to network failures. When network connectivity falters, training can continue on local devices, and updates can be sent once the network is restored, making it particularly valuable in environments with generally unstable network connections. Federated Learning can additionally serve the purpose of tailoring the model to individual users or distinct user cohorts through the development of discrete models informed by their respective specific datasets and characteristics [18]. This approach has the potential to enhance the precision and relevance of the model across a spectrum of users or diverse user segments, simultaneously diminishing the necessity for superfluous data dissemination. In the complex landscape of real-world data, where information may be distributed in an uneven manner, Federated Learning's adaptive nature ensures that data from diverse sources and environments coalesce, potentially giving rise to a more versatile and generalized model proficient across an extensive range of data distributions [19].

The motivation of this thesis work is to propose two new approaches in the context of federated Learning, which will not only be computationally efficient and enhanced in quality but also enable a broader vision of empowering data-driven decision-making through big data analysis across various domains. By addressing the core challenges that limit the application of traditional algorithms to large datasets, this thesis contributes to the field of data mining and machine learning in a manner that is both profound and

practical.

1.2 Objective

The primary aim of this thesis paper is to extend the capabilities of the Decision Tree algorithms in the Federated Learning environment to address the challenges posed by the analysis of large-scale datasets. To accomplish this, the following objectives have been outlined:

- A privacy-preserving approach is proposed for mining big data employing the RainForest framework within Federated Learning environment. In this method, each local device trains a Decision Tree classifier using big data and shares only the probability values as an AVC group with the central server.
- Another novel approach is proposed by using traditional rule-based classifiers in the Federated Learning environment. This method proposes to generate classification rules from local data on each client device, sharing only these rules with the central server. The server amalgamates, refines, and sends them to each client device in an iterative process. So, the rules adapt to the new concepts over time.

1.3 Organization of the Thesis

The content of this thesis paper is organized as follows: Chapter 2 presents a comprehensive exposition of extant knowledge on Federated Learning with a detailed overview of pertinent research papers in the realm of Federated Learning. Chapter 3 introduces the proposed methodologies of this work. Chapter 4 explains the implementation part of the proposed methodologies and provides an overview of the achieved results. Chapter 5 presents some limitations of both the proposed works and concludes the study with future directions.

Chapter 2

Background

2.1 Federated Learning

Since its introduction, more advanced ideas have been added to Federated Learning technology. More and more devices, no matter how tiny, now have the processing power to run Federated Learning. It can be implemented either in a centralized [20] or decentralized [21] manner, depending on the desired preferences.

2.1.1 Centralized Federated Learning

In the Centralized Federated Learning (CFL) setting, a central server is responsible for orchestrating the different steps of the algorithms and coordinating all the participating nodes during the learning process. Gboard, Google's own keyboard application, uses CFL to improve its predictive typing model. Also, Apple's Health application uses CFL to improve its activity tracking model. Fig. 2.1 shows the basic structure of the CFL.

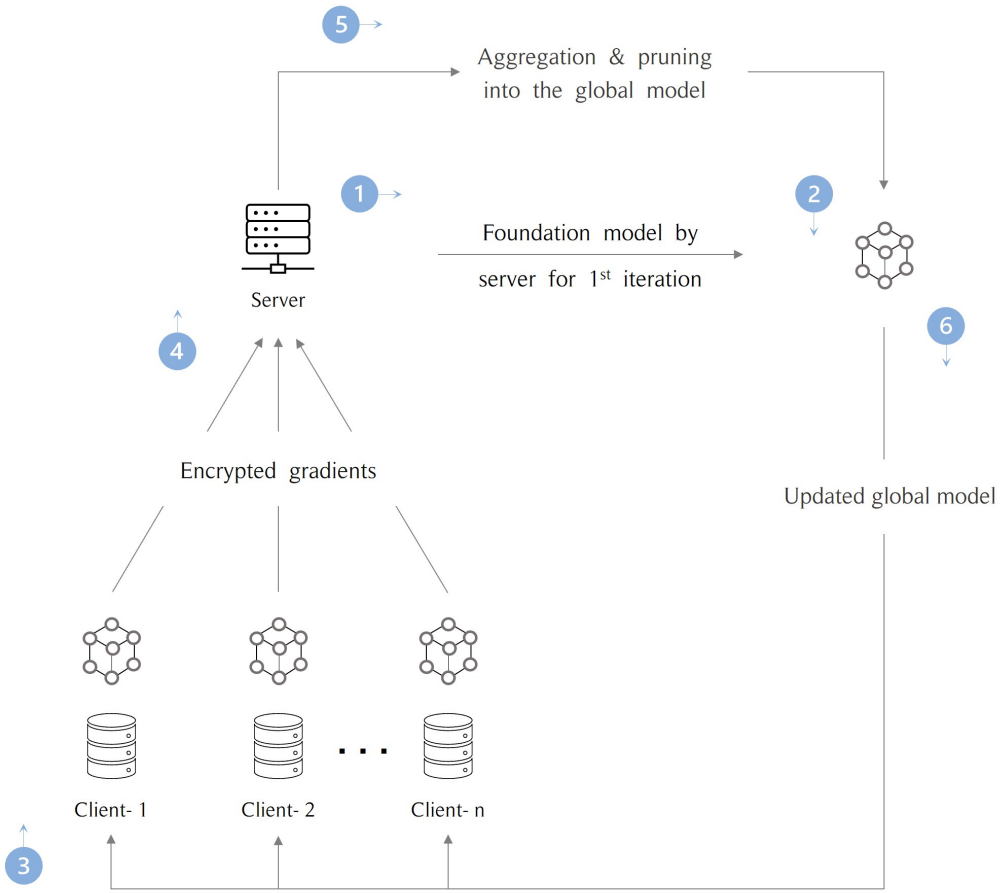


Figure 2.1: Centralized Federated Learning.

Assuming a federated round composed of one iteration of the learning process, the learning procedure of CFL can be summarized as follows:

1. Client selection: The server engages in a process of client selection from a pool of eligible candidates. Eligibility is determined by specific criteria, such as mobile phones being required to be plugged in, connected to a wi-fi network, and in an idle state. This selection strategy aims to ensure minimal disruption to the device user.

2. Broadcast: Upon selection, the chosen clients proceed to download the latest model weights and a training program (e.g., a TensorFlow graph [22]) from the server.

3. Client computation: Each client device, once selected, independently computes a model update by executing the provided training program. This update process might involve employing techniques like Stochastic Gradient Descent (SGD) on its local data, as seen in the Federated Averaging approach.

4. Aggregation: The server then gathers an aggregate of the model updates received from the participating devices. To enhance efficiency, instances of slower or delayed devices, known as stragglers, may be excluded from this point onward, once a

sufficient number of results have been collected. This stage also serves as an integration point for various privacy-enhancing methodologies, which may include secure aggregation to bolster privacy measures, the use of lossy compression to optimize communication efficiency, and the application of noise addition and update clipping to enforce differential privacy.

5. Model update: The current round's aggregated update is calculated by all participating clients, and the server applies that update to the shared model.

The complete pseudocode of CFL is given in Algorithm 1.

Algorithm 1 Centralized Federated Learning Algorithm

Input: Initial global model M, number of communication rounds T, set of client devices $\{C_1, C_2, \ldots, C_k\}$
Output: Final global model $M^{(T)}$
Method:

for each communication round $t = 1$ **to** T **do**
  for each client $C_i \in \{C_1, C_2, \ldots, C_k\}$ **in parallel do**
    Retrieves $M^{(t-1)}$ from the central server
    Local model, $M_i^{(t)} \leftarrow M^{(t-1)}(C_i.data)$
    Transmits $M_i^{(t)}$ to the central server
  end for
  Aggregation of local models at the central server to update the global model: $M^{(t)}$
end for
$M^{(T)} \leftarrow \sum M^{(t)}$

Data flow. In the context of a CFL configuration, the manner in which data is exchanged between the server and the edge devices can manifest as either synchronous or asynchronous. Synchronous data flow is the traditional approach to CFL. In synchronous data flow, the server waits for updates from all of the clients before aggregating them and updating the shared model [14]. Asynchronous data flow allows the server to aggregate updates from clients as they become available [23]. This means that the server can start training the model as soon as it receives an update from a client, even if other clients have not yet sent their updates. The best method of data flow in CFL depends on the specific application and the requirements of the user, as mentioned below.

- Accuracy: Synchronous data flow could be used for applications where accuracy is critical, such as medical diagnosis or financial fraud detection. Because it ensures that all of the clients are using the same version of the model. In an asynchronous environment, clients may have different versions of the model, which leads to less accuracy in the updated model.

- Training speed: Asynchronous data flow could be used for applications where speed is more important, such as personalized recommendations or fraud detection in real

time. Synchronous data flow is comparatively slower as it waits for updates from all clients.

- Number of clients: If there are a large number of clients, then asynchronous data flow may be a better choice, as it can help reduce the delays caused by waiting for updates from all of the clients.
- Device heterogeneity: If the clients have different computational capabilities, then asynchronous data flow may be a better choice, as it can help ensure that all of the clients are able to participate in the training process.

2.1.2 Decentralized Federated Learning

Within the framework of Decentralized Federated Learning (DFL), the necessity for a central server is obviated, as this architectural paradigm embraces a distributed arrangement. Instead, each client device maintains its own copy of the model and updates its parameters locally. The clients then exchange their updated parameters with their neighbors in a decentralized manner, and the parameters are aggregated to update the global model [24]. Fig. 2.2 shows the basic structure of the DFL.

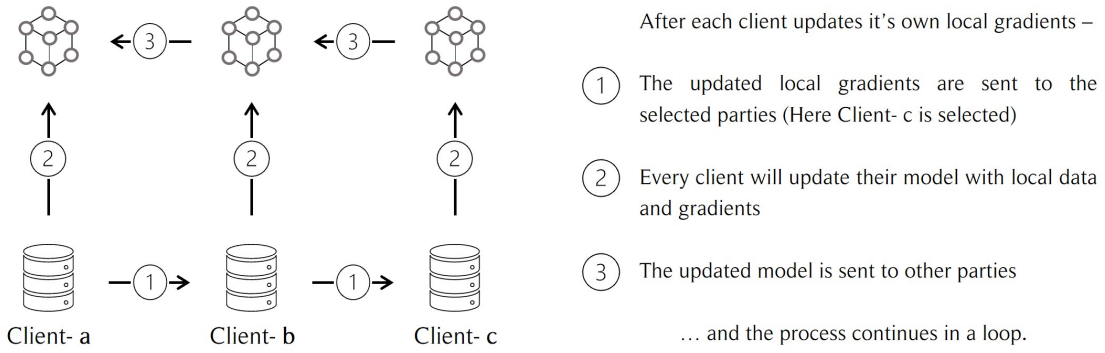


Figure 2.2: Decentralized Federated Learning.

Updating the exchange can be done in different ways. Examples include peer-to-peer approaches, the gossip algorithm, hierarchical clustering, etc. In a peer-to-peer approach, clients communicate directly with each other to exchange model updates [25]. This is the most decentralized approach to DFL, but it can be inefficient as it requires a lot of communication between clients. In gossip-based DFL, clients communicate with a randomly selected subset of other clients to exchange model updates [26]. This is a more efficient approach to DFL than peer-to-peer DFL, but it is less decentralized. In hierarchical clustering, devices are organized into clusters, and updates are aggregated within each cluster before being sent to other clusters [27]. The complete pseudocode of DFL is given in Algorithm 2.

Algorithm 2 Decentralized Federated Learning Algorithm

Input: Initial global model M, number of communication rounds T, set of decentralized nodes $\{N_1, N_2, \ldots, N_k\}$
Output: Final global model $M^{(T)}$
Method:

for each communication round $t = 1$ **to** T **do**
  All nodes select a neighbor node, $N_{neighbour}$, for decentralized model sharing
  for each client $N_i \in \{N_1, N_2, \ldots, N_k\}$ **in parallel do**
    Retrieves $M^{(t-1)}$ from $N_{neighbour}$
    Local model, $M_i^{(t)} \leftarrow M^{(t-1)}(N_i.data)$
    Transmits $M_i^{(t)}$ to the $N_{neighbour}$
  end for
  Aggregation of local models at $N_{neighbour}$ to update the global model: $M^{(t)}$
end for
$M^{(T)} \leftarrow \sum M^{(t)}$

Limitations. The main advantage of the DFL is its resilience. It is more resilient to failures than CFL, as it does not rely on a central server. This makes it a good choice for applications where the data or the network may be unreliable. However, it has some demerits as well.

- Complexity: DFL is more complex to implement than CFL, as it requires a distributed architecture. This can be a challenge for organizations that do not have the expertise or resources to implement it. Additionally, as the number of connected devices increases, the complexity of the communication process also grows [28].
- Convergence: A DFL can take longer to converge than a CFL. This is because the clients have to communicate with each other to share their local updates, which can slow down the learning process. Also, there is a possibility that clients may not be able to agree on a global model.
- Security: DFL can be more difficult to secure than CFL. This is because the clients are communicating with each other directly, which means that there is more opportunity for data to be intercepted or tampered with.
- Communication overhead: DFL can have a high communication overhead because the devices need to communicate with each other to share their updates.
- Heterogeneity: DFL can be challenging to implement in heterogeneous environments where the devices have different computing power and communication capabilities.

Data partition mode. Data is often stored in several entities using a feature matrix as the primary data storage format. Typically, the data will have numerous instances, and

the clients will be on the horizontal axis with their characteristics on the vertical. Federated Learning is usually grouped into three types based on data partition mode: Horizontal Federated Learning, Vertical Federated Learning, and Federated transfer learning [29].

1. Horizontal Federated Learning: The applicability of Horizontal Federated Learning (HFL) is evident when the feature sets of multiple distinct datasets exhibit significant congruence, yet the individual users within these datasets display minimal overlap [30]. This approach to machine learning segregates datasets along the user dimension, subsequently training on data segments where user features coincide but users themselves diverge. This method, thus, amplifies the scope of the user sample size. As an illustration, consider the scenario of two hospitals operating in distinct geographical regions. Each has a unique patient base confined to its respective region, hence displaying minimal intermingling. Despite this, due to the similar nature of their services, the patient attributes recorded are mostly identical, including patient vitals, medical histories, diagnoses, treatments, and outcomes. In such a situation, both hospitals can collaboratively train a machine learning model to predict disease outcomes or treatment effectiveness through this HFL process and significantly improve their predictive models' performance as it escalates the aggregate count of training samples. Fig. 2.3 gives a clear idea of this process. One example of a HFL framework is BlockFL [31], in which each mobile device uses the block chain network to update the local learning model.

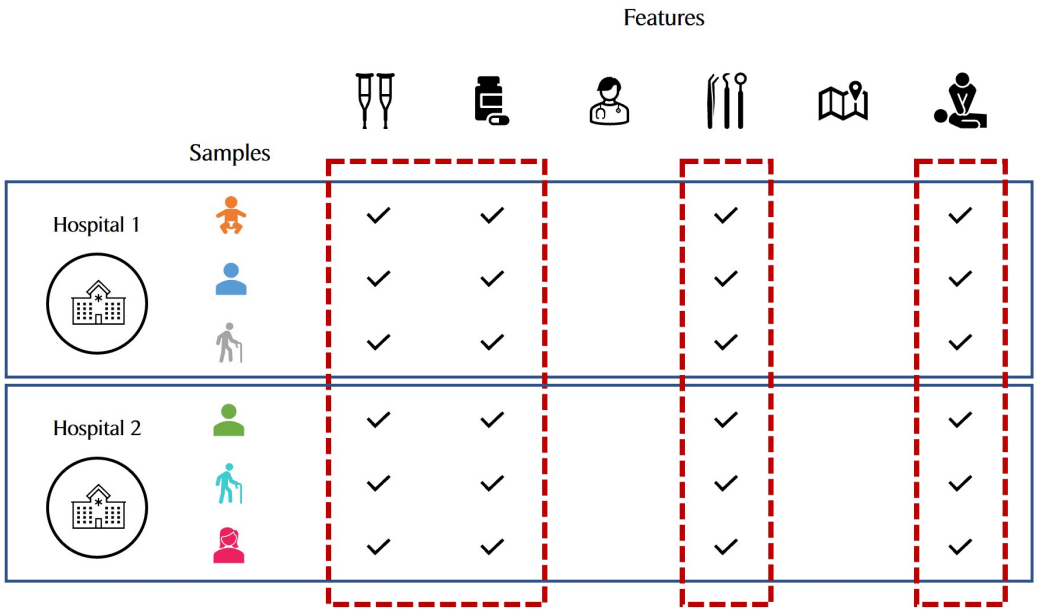


Figure 2.3: Example of Horizontal Federated Learning.

2. Vertical Federated Learning: In instances where multiple datasets possess little overlap in user features yet exhibit substantial overlap in users themselves, the applica-

tion of Vertical Federated Learning (VFL) is appropriate. This approach involves segregating the datasets vertically by the dimensions of user features and then selectively utilizing the data sections where users are identical but feature characteristics differ for the purpose of training [30]. As a consequence, VFL can augment the feature dimension of the training data. Consider an illustrative example involving two separate institutions located in the same vicinity, such as a bank and an e-commerce company. It is highly probable that their user base comprises the majority of the area's residents, resulting in a significant user overlap. However, the intersection of user features is negligible, as banks typically document users' financial behaviors and credit ratings, while e-commerce platforms maintain records of users' browsing and purchasing histories. As exemplified in Fig. 2.4, the VFL process showcases the seamless integration of their data resources to facilitate the creation of AI-driven personalized loan recommendations based on their online shopping behaviors.

Figure 2.4: Example of Vertical Federated Learning.

SecureBoost [32] is a prime example of a VFL system. In this context, all involved parties amalgamate user features to facilitate collective training, thereby enhancing the precision of decision-making. This represents a lossless training approach.

3. Federated Transfer Learning: On occasion, the intersection of users and user features across two datasets is quite limited. Consider, for instance, Automobile Manufacturer A and Ride-Sharing Service B. Manufacturer A, based in Germany, has developed an advanced predictive maintenance model from its vast dataset of global vehicle performance, fuel efficiency, and driving habits. However, Service B, a US company, needs a model for route and fare optimization based on its collected data, including user travel patterns, popular routes, ride timings, and trip fares. Now,

company B wants to use the data from company A. To perform Federated Learning effectively in such circumstances, it becomes imperative to use transfer learning. In 2018, Liu et al. [33] devised Federated Transfer Learning (FTL) to handle these types of scenarios. This strategy facilitates the conveyance of knowledge from one domain, referred to as the source domain, to a different domain, called the target domain, which subsequently yields improved learning outcomes [30]. In this case, company B could use FTL to initialize its machine learning model with the model from company A. This would give company B's model a 'head start' by using company A's insights about vehicle behavior. While company B's model would not directly use company A's specific vehicle metrics, the transferred model could apply its understanding of how different factors influence vehicle usage and performance to company B's data. Company B's model then further trains on its own dataset to learn the specifics of its task, like optimizing routes and ride fares. In this way, company B can benefit from company A's large dataset and modeling experience without directly accessing its data, thus preserving data privacy. Fig. 2.5 shows a rough idea of this process.

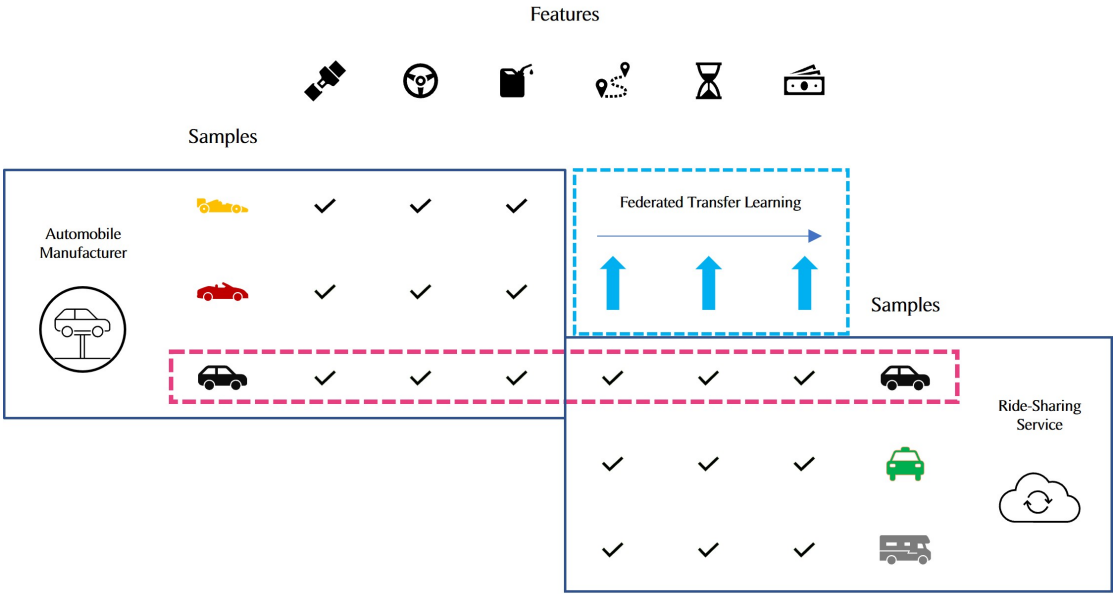


Figure 2.5: Example of Federated Transfer Learning.

2.2 Literature Review

In a Federated Learning environment, where model training is distributed across many devices and then aggregated on a central server, dealing with large models or their huge gradients presents significant challenges, especially in terms of communication cost and efficiency. Experts have already suggested and implemented various methods to transfer sizable models with substantial gradients or parameters gathered from client devices to a central server in a more manageable format. Table 2.1 provides a brief overview of those

works, noting some of their limitations as well.

Table 2.1: Overview and gap analysis of existing methodologies.

No.	Title	Approach	Limitation
1	Deep compression: Compressing deep neural networks with pruning, trained quantization, and huffman coding (2015) [34]	The authors pruned the network by removing connections with weights below a set threshold, significantly reducing its connections. Following pruning, weights were quantized and shared by clustering and fine-tuning centroids, compressing the network further. This reduced the number of bits needed to represent each connection from 32 to 5 bits. Lastly, Huffman coding leveraged the biased distribution of effective weights, offering additional compression.	This method can face limitations, including hardware compatibility issues, as not all platforms may directly support the required quantization and Huffman coding steps. The complexity of implementing this multi-stage compression process could also pose challenges, especially for those with limited resources or technical expertise. Moreover, the approach might not suit dynamically changing environments or very large, complex models where aggressive compression could inadvertently eliminate critical connections, potentially affecting model robustness.
2	Gossip algorithms: Design, analysis, and applications (2005) [35]	It reduced the load on a central server by aggregating updates in a decentralized manner before a global update is performed.	Its practical application faces limitations, including the necessity for accurate network topology knowledge, potentially slow convergence in large or sparsely connected networks, computational overhead associated with SDP, accuracy concerns in distributed gradient computation, scalability challenges in extremely large networks, and the feasibility of energy-constrained environments like sensor networks.

Table 2.1: Overview and gap analysis of existing methodologies.

No.	Title	Approach	Limitation
3	Distilling the knowledge in a neural network (2015) [36]	The authors implemented the knowledge Distillation process that trained a smaller (student) model to mimic the behavior of a larger (teacher) model or ensemble of models using soft targets, which are the outputs of the larger model softened by a high-temperature softmax function. Only the parameters of the smaller model were then sent to the server.	Distillation relies on tuning a parameter to adjust output probabilities, with optimal values varying across tasks and model sizes, necessitating experimentation. While it can enhance smaller models' generalization with less data, data quality and quantity remain crucial. Also, the representatives and diversity of the transfer set used affect the distilled model's performance. Moreover, training an initial model for distillation can be computationally intensive, posing challenges with limited resources. Also, generalization to unseen classes presents additional hurdles. An experiment highlights distillation's capabilities and limitations in such scenarios.
4	QSGD: Communication efficient SGD via randomized quantization and encoding (2018) [37]	The authors implemented the gradient Quantization process that reduced the bit representation of gradients. Also, the implementation of the gradient Sparsification process involved only sending gradients that were larger than a predefined threshold, thereby ignoring smaller gradients that might have had a minimal impact on model updates.	While QSGD significantly enhances communication efficiency in Federated Learning by allowing a balance between compression and accuracy through gradient quantization, it introduces challenges such as increased variance that may impact convergence rates. The effectiveness of QSGD can vary with different neural network architectures, requiring careful hyper-parameter tuning to achieve optimal results. Furthermore, the approach's focus on gradient quantization without fully exploiting gradient sparsity and its primary design for SGD necessitate additional considerations for broader optimization algorithms.

Table 2.1: Overview and gap analysis of existing methodologies.

No.	Title	Approach	Limitation
5	Scalable distributed DNN training using commodity GPU cloud computing (2015) [38]	This work explores the gradient sparsification process, which involves only sending gradients that are larger than a predefined threshold, thereby ignoring smaller gradients that might have a minimal impact on model updates.	The gradient sparsification process exhibits comparatively lower accuracy as it reduces the number of gradients.
6	Scaling distributed machine learning with the parameter server (2014) [39]	The authors introduced a parameter server framework that addresses the challenges of managing large models and gradients by enabling asynchronous communication, employing flexible consistency models, and supporting elastic scalability and fault tolerance. This framework significantly enhances communication efficiency, allowing for dynamic adjustment of computational resources without interrupting ongoing computations and ensuring system robustness through mechanisms for rapid recovery from failures.	The proposed model faces multiple limitations such as data staleness due to asynchronous communication and flexible consistency models, complexity in tuning for optimal balance between efficiency and algorithm performance, increased system overhead for maintaining fault tolerance and scalability, reliance on robust network infrastructure, and the necessity for task-specific customization. These challenges underscore the trade-offs between operational efficiency and model accuracy, and the need for careful system configuration to navigate the varying demands of different learning tasks and network conditions.

Table 2.1: Overview and gap analysis of existing methodologies.

No.	Title	Approach	Limitation
7	Redundancy techniques for straggler mitigation in distributed optimization and learning (2019) [40]	By encoding the dataset to incorporate built-in redundancy, the framework effectively manages straggling nodes as 'erasures', ensuring that their absence does not hinder the learning process. This approach allows for the convergence of various optimization algorithms to approximate solutions without requiring updates from all participating nodes.	This approach comes with several limitations, including increased computational overhead for encoding and decoding, added system complexity, and a trade-off between redundancy and efficiency. It may also exhibit dependency on specific loss functions, potentially affecting learning performance for non-quadratic or non-convex objectives. Moreover, while redundancy aids in mitigating the impact of stragglers, it introduces scalability challenges in highly distributed environments, necessitating careful balance and potential system adaptations to ensure efficient learning without compromising model accuracy or increasing resource demands significantly.

To address the challenges presented by these existing issues, we have formulated two distinct methods for parameter exchange. Both methodologies aim to streamline the communication process and reduce the computational burden on the central server, directly confronting the challenges of managing large models and ensuring robust, efficient learning in distributed environments. These methods not only address specific limitations of existing strategies but also introduce a structured approach to maintaining high model accuracy and reliability without incurring excessive communication costs.

Chapter 3

Proposed Methodology

This chapter provides an overview of the Decision Tree algorithm and the RainForest framework, as well as exemplifies both of the proposed methodologies that utilize Federated Learning to tackle privacy concerns and oversee large-scale data management.

3.1 Decision Tree

A Decision Tree (DT) can be defined as a classifier that partitions the instance space in a recursive manner. The tree structure consists of nodes, forming a directed tree where one node, called the Root, has no incoming edges while all other nodes have exactly one incoming edge. Nodes with outgoing edges are referred to as Internal or Test nodes, whereas the remaining nodes are known as Leaf Nodes, also called Terminal or Decision nodes. Within a Decision Tree, each internal node divides the instance space into two or more sub-spaces based on a discrete function of the input attribute values [41]. Typically, each test in the simplest and most common scenario examines a single attribute, resulting in the partitioning of the instance space according to the attribute's value. In the case of numeric attributes, the condition refers to a specific range. Each leaf node is associated with a particular class, representing the most suitable target value. Alternatively, a leaf node may contain a probability vector indicating the likelihood of the target attribute having a specific value. Instances are classified by traversing the tree from the root to the leaf, following the outcome of the tests along the path. In the case of numeric attributes, Decision Trees can be geometrically interpreted as a set of hyperplanes, each orthogonal to one of the axes [41]. Decision-makers generally prefer less complex Decision Trees, as they tend to be more comprehensible. Additionally, according to [42], the complexity of the tree significantly impacts its accuracy. A well-known Decision Tree algorithm is depicted in Algorithm 3.

Here is a step-by-step explanation of how the Decision Tree algorithm works [43]:

1. Data preparation: The initial step involves preparing the data for implementation in the Decision Tree algorithm. This process necessitates the selection of suitable

Algorithm 3 Decision Tree Algorithm

FUNCTION BuildDecisionTree ($Data$)

if $AllSamplesAreSameClass(Data)$ **then**
  return $CreateLeaf(NodeClass = TheClass)$
end if
if $NoFeaturesLeftToSplitOn(Data)$ **then**
  return $CreateLeaf(NodeClass = MajorityClass(Data))$
end if
$BestFeature \leftarrow$ ***BestFeatureToSplit***$(Data)$
$Node \leftarrow CreateNode(Feature = BestFeature)$
$Data_{left}, Data_{right} \leftarrow SplitData(Data, BestFeature)$
if $Data_{left} \neq \emptyset$ **then**
  $Node.LeftChild \leftarrow BuildDecisionTree(Data_{left})$
else
  $Node.LeftChild \leftarrow CreateLeaf(NodeClass = MajorityClass(Data))$
end if
if $Data_{right} \neq \emptyset$ **then**
  $Node.RightChild \leftarrow BuildDecisionTree(Data_{right})$
else
  $Node.RightChild \leftarrow CreateLeaf(NodeClass = MajorityClass(Data))$
end if
return $Node$

FUNCTION BestFeatureToSplit ($Data$)

$BestFeature \leftarrow NULL$
$BestImpurityDecrease \leftarrow 0$
for $Feature \in Features(Data)$ **do**
  $ImpurityDecrease \leftarrow$ ***CalculateImpurityDecrease***$(Data, Feature)$
  if $ImpurityDecrease > BestImpurityDecrease$ **then**
    $BestFeature \leftarrow Feature$
    $BestImpurityDecrease \leftarrow ImpurityDecrease$
  end if
end for
return $BestFeature$

FUNCTION CalculateImpurityDecrease ($Data, Feature$)

$ImpurityBeforeSplit \leftarrow CalculateImpurity(Data)$
$Data_{left}, Data_{right} \leftarrow SplitData(Data, Feature)$
$ImpurityAfterSplit \leftarrow WeightedAverageImpurity(Data_{left}, Data_{right})$
return $ImpurityBeforeSplit - ImpurityAfterSplit$

features (independent variables) and the target variable (dependent variable) in a format compatible with the algorithm.

2. Tree construction: The Decision Tree algorithm commences by constructing the root node of the tree. At this stage, the algorithm assesses various features to determine the most effective means of dividing the data into homogeneous subsets. This procedure is commonly referred to as attribute selection or the splitting criterion.

3. Splitting the data: Following the determination of the optimal splitting criterion, the algorithm partitions the data into multiple subsets based on the selected feature. Each subset corresponds to a distinct value of the feature. This recursive process is repeated for every subset, thereby generating child nodes and further dividing the data until a predetermined stopping condition is met. Commonly encountered stopping conditions comprise: (a) all instances within a subset belonging to the same class; (b) attainment of a maximum depth or tree size; (c) instances within a subset falling below a predefined threshold; (d) accomplishment of a predefined level of impurity.

4. Leaf node creation: When a stopping condition is satisfied, a leaf node is created to represent the predicted class or value for the corresponding subset. In a classification problem, the majority class is frequently designated as the prediction, while in a regression problem, the average value is often utilized.

5. Pruning (Optional): Pruning constitutes an optional step employed to mitigate overfitting, a phenomenon characterized by a Decision Tree becoming excessively intricate and tailored to the training data, leading to inadequate generalization on new, unseen data. Pruning techniques involve the removal or consolidation of nodes to simplify the decision tree while preserving its predictive accuracy.

6. Prediction: Once the Decision Tree has been constructed, it can be utilized to make predictions regarding new, unseen instances. To classify an instance, the tree traverses from the root node to a leaf node, adhering to the decision rules at each internal node based on the values of the instance's features.

There are various Decision Tree algorithms such as ID3 [44], ID4 [45], ID5 [46], C4.5 [47], C5.0 [48], CART [49], QUEST [50], FACT [51], CHAID [52], FIRM [53], CRUISE [54], etc. The invention and use of those algorithms stem from several factors. Here are some reasons for their existence and application:

- Problem-specific considerations: Various Decision Tree algorithms have been specifically developed to address distinct problem types and data characteristics. In the case of classification tasks, certain algorithms exhibit greater suitability, whereas others excel at regression or the handling of categorical variables. Dissimilarities may also arise in their capacity to manage imbalanced data, missing values, or noisy

data. For instance, when confronted with purely categorical data, the simplicity of ID3 makes it an appealing option [44]. Conversely, if the dataset contains a combination of categorical and continuous variables, along with missing values, the preferences lean towards C4.5 [47] or CART [49]. In situations where the data indicates the potential benefits of multi-level splits, CHAID [52] could be the preferable choice. Conversely, when the problem involves regression, CART [49] emerges as a fitting selection. The concern of overfitting can be mitigated by opting for Random Forests. For extensive datasets, the employment of XGBoost [55], LightGBM [56], or CatBoost [57] is recommended, as these algorithms have been designed to deliver high performance in such contexts.

- Algorithmic improvements: Researchers and practitioners continually strive to develop more accurate, efficient, and scalable decision tree algorithms. New algorithms often incorporate innovative techniques or modifications into existing approaches to enhance performance. These improvements may include addressing bias-variance trade-offs, reducing overfitting, handling large datasets, or improving interpretability.

- Ensemble methods: Ensemble methods combine multiple Decision Trees to form more powerful models. Algorithms like Random Forests [58], Gradient Boosting [59], and AdaBoost [60] use different strategies for constructing ensembles, such as combining multiple weak learners or utilizing different sampling techniques. Each ensemble method has its own unique advantages and trade-offs, making it suitable for different scenarios.

- Interpretability and explainability: Decision Trees are often preferred for their inherent interpretability, as they generate transparent and easily understandable decision rules [61]. However, different algorithms can offer varying levels of interpretability. Some algorithms prioritize interpretability over predictive accuracy by favoring simpler tree structures, while others focus on optimizing predictive performance at the expense of interpretability.

- Scalability and efficiency: As datasets grow in size and complexity, scalability and efficiency become important considerations. Different algorithms employ diverse strategies to handle large-scale datasets, such as parallelization, sampling techniques, or optimized data structures. Some algorithms are better suited for big data scenarios, while others excel on small to medium-sized datasets.

- Research and experimentation: The field of machine learning is continually evolving, and researchers often propose new Decision Tree algorithms to explore novel ideas, validate theoretical concepts, or compare them against existing methods. This constant experimentation drives progress and fosters the development of more effective and specialized algorithms.

Splitting criteria. In the context of a dataset containing multiple attributes, the task of determining the optimal hierarchical arrangement of these attributes can be notably intricate. Simply resorting to a random selection approach may result in imprecise and unfavorable outcomes. In response to this challenge, researchers have diligently addressed the matter, culminating in the proposal of diverse solutions. Several criteria have been advanced, encompassing methods such as Entropy, Information Gain, Gini Index, Gain Ratio, Chi-Square Test, Reduction in Variance, Squared Error, Mean Squared Error, Mean Absolute Error, etc.

- Entropy: Entropy is primarily used in classification problems. It is a measure to determine where to split the data. This concept is borrowed from information theory, which measures the impurity, uncertainty, or disorder within a set of data. Mathematically, it can be represented as:

$$E(S) = -\sum_{i=1}^{k} (p_i \log_2 p_i) \tag{3.1}$$

  Here, S is the total sample space, and p_i represents the proportion of instances of class i in the data set [62]. Entropy will be zero when all instances belong to a single class, implying that there is no uncertainty or disorder (in other words, all the data is purely of one class). Conversely, the entropy will be highest when instances are evenly split among classes, implying high uncertainty or disorder (as it is unsure which class a random instance will belong to) [63].

- Information Gain: Information Gain (IG) is the reduction in entropy achieved by partitioning the samples according to a given feature. It measures how much 'information' a feature gives us about the final outcome after being split [64]. It is used for classification cases. Mathematically, IG can be calculated as:

$$InfoGain(S, A) = Entropy(S) - \sum_{i=1}^{A} \left\{ \frac{S_V}{S} \times Entropy(S_V) \right\} \tag{3.2}$$

  Here, S is the total sample space, A is the feature or attribute, and S_V is the subset of S for which attribute A has value V. When constructing a DT, the feature with the highest IG is typically chosen as the attribute to split on at each node. Among all the attributes, the higher the IG, the higher it will go in the tree. This method results in smaller trees and reduces computational complexity. However, one caveat is that the method is biased towards attributes with many outcomes; it tends to select them because they can result in more partitions and hence more potential for reducing Entropy. The concept of IG forms the basis of the ID3 algorithm as well [44].

- Gain Ratio: The Gain Ratio is an extension of the Information Gain that takes into account the number and size of branches when a node is split. Despite its usefulness, IG is biased towards attributes with more levels (more unique values that are not always important). Such an attribute may lead to many splits, each explaining a small portion of the data, which can cause overfitting. Therefore, it might not be the best attribute to focus on, even though it offers the most information. To overcome the shortcoming of IG, the Gain Ratio was introduced [65]. It incorporates a term known as 'Split Information' that is analogous to Entropy but for the potential splits. Split Information considers the potential splits of a node and provides a measure for their uniformity. The Gain Ratio is then defined as the IG divided by the Split Information. Mathematically:

$$GainRatio\,(S, A) = \frac{InfoGain\,(S, A)}{SplitInfo\,(S, A)} \tag{3.3}$$

where A is the attribute on which to split.

$$SplitInfo\,(S, A) = -\sum_{i=1}^{|A|} \left(\frac{S_V}{S} \times \log_2 \frac{S_V}{S} \right) \tag{3.4}$$

The Split Information term in the denominator penalizes high-cardinality attributes (i.e., attributes with many unique values), reducing their impact on the decision to split. This makes the Gain Ratio a more balanced measure, favoring less complex Decision Trees. When selecting the attribute with the highest Gain Ratio, the Decision Tree algorithm can effectively balance the information provided by the attribute with the potential complexity of the resulting branches. Attributes with a higher Gain Ratio are considered more informative for splitting a node. However, it's essential to know that the Gain Ratio is not always superior to IG or other splitting criteria. Depending on the data, the Gain Ratio can sometimes favor attributes with few unique values. Therefore, in practice, it can be beneficial to experiment with different splitting criteria to find the one that works best for a given dataset. The C4.5 algorithm primarily uses the Gain Ratio to determine the best attribute to split the data at each node of the Decision Tree [63].

- Gini Index: The Gini Index, also known as the Gini impurity, is a measure of how often a randomly chosen element from a set would be incorrectly labeled if it were randomly labeled according to the distribution of labels in the subset. In simpler terms, it quantifies the purity of an attribute or a node in a Decision Tree. It is a popular splitting criterion used in Decision Tree algorithms, such as CART, when dealing with classification problems [63]. The Gini Index varies between values 0 and 1, where 0 expresses the purity of classification, i.e., only one class is present, while 1 denotes a random distribution of elements across various classes. The lower

the value of the Gini Index, the better the split, and it will sit higher, followed by the rest in order. For multi-class classification, the Gini Index formula is:

$$Gini\,(S) = 1 - \sum_{i=1}^{k} \left({p_i}^2\right) \tag{3.5}$$

where p_i means the probability of class 'i'.

3.2 RainForest

Traditional Decision Tree algorithms have been widely used for classification tasks due to their intuitive representation and ability to handle complex datasets. However, these algorithms can encounter limitations when dealing with large-scale datasets or when the available memory is insufficient. The scalability problem arises when the size of the dataset or the number of predictor attributes exceeds the capacity of main memory. Traditional algorithms may struggle to construct the Decision Tree efficiently under such circumstances, leading to performance degradation or even out-of-memory errors. In the realm of data mining, the classification of large datasets emerges as a pivotal challenge, necessitating robust and efficient algorithms. The RainForest framework stands out as a paradigm-shifting framework for the construction of Decision Trees, especially when handling extensive datasets. It provides a solution to this problem by separating the scalability aspects of Decision Tree construction from the central features that determine the quality of the tree. This separation enables the framework to be instantiated with most Decision Tree algorithms and provides a scalable version of these algorithms without modifying their core features or sacrificing the quality of the resulting Decision Tree.

3.2.1 Mechanism

At its core, the RainForest framework distinguishes itself through a unique approach to data management and scalability. It operates on the principle of segregating the dataset into AVC-group (Attribute-Value Class-label group), which essentially compacts the data into a more manageable form, significantly reducing memory requirements without compromising the Decision Tree's quality. This ingenious method allows for scalable versions of various classification algorithms, including C4.5, CART, and ID3, among others, to be instantiated within the RainForest framework seamlessly. In understanding the RainForest, it is useful to keep in mind that the following steps are carried out for each tree node n [66]:

1. The algorithm generates an AVC set for each predictor attribute at a node n by scanning the dataset. All AVC sets of a node n together form the AVC group of that node n.

2. For each predictor attribute, the algorithm finds the best partitioning by examining the AVC sets within the AVC group of node n. This decision is based on split selection methods like Information Gain or Gini Index, which measure how well an attribute splits the data into homogeneous groups.

3. After choosing the best splitting attribute, the algorithm divides the dataset into subsets based on the chosen attribute, creating child nodes for each category of the attribute.

By applying RainForest, memory consumption can be reduced through efficient management of AVC sets in each of these steps. While generating AVC-sets, memory is used to store AVC-sets for each attribute, but only one attribute's AVC-set is held in memory at a time. This implies that the entire dataset does not need to be loaded into memory, thus reducing overall memory usage. While finding the best partition for each predictor attribute, instead of the full dataset, only the AVC sets are employed. This step necessitates less memory. Also, when the algorithm compares this result, it uses aggregated statistics rather than the full data, further minimizing memory use. In the last step, when records are split into new nodes, only the relevant AVC-set information is used, not the entire dataset. This keeps memory requirements low as the tree grows. In essence, the RainForest framework streamlines Decision Tree construction for large datasets by focusing on efficiently summarizing and utilizing data, thereby enhancing scalability without sacrificing the quality of the resulting Decision Tree. The complete pseudocode of the RainForest algorithm is depicted in Algorithm 4.

Algorithm 4 RainForest Algorithm

FUNCTION BuildTree (Training database D, node n, split selection method SS):

```
for each predictor attribute X do
    Call SS.find_best_partitioning (AVC-set of X)
end for
SS.decide_spliting_criterion ()
Let k be the number of children of n
if k > 0 then
    Create k children n_1, n_2, ..., n_k of n
    Use best split to partition D into D_1, D_2, ..., D_k
    for (i = 1; i <= k; i++) do
        BuildTree (n_i, D_i, SS)
    end for
end if
```

3.3 Implementation of the RainForest framework in Federated Learning

In this section, we have explained the methodology of our first proposed model, which incorporates the RainForest framework in a Federated Learning environment. Figure 3.1 shows an overview of this method. In the initial phase of this proposed model, the central server delineates a cohort of local client devices to establish the Federated Learning ecosystem. Then, it divides the big data stored in its repository into an equal number of subsets of local client devices for use in training. Any equitable data distribution technique can be used to subset this. The central server then sends one subset to each local client device. Each local client device combines its own data and received subset to create an AVC group on it. Thus, all local client devices create their own AVC group and send it to the central server, where the AVC sets are combined by matrix addition according to the attributes. The central server then uses a split selection method to evaluate every attribute and choose the one that maximizes purity. The central server then directs the local client devices to make their decisions for that node using that specific attribute. Each local client device divides its data at that specific node according to different classes of that attribute. This process continues in a recursive way for each child node and the corresponding subdataset obtained after partitioning. This iterative process continues until all the possible splits have been performed or the specified criteria are met. For the segmentation of big data in our experiment, a straightforward K-means clustering technique [67] was deployed. Furthermore, the selection of the optimal attribute for splitting at each node was accomplished through the utilization of Information Gain [64]. The pseudocode of this proposed model is mentioned in Algorithm 5.

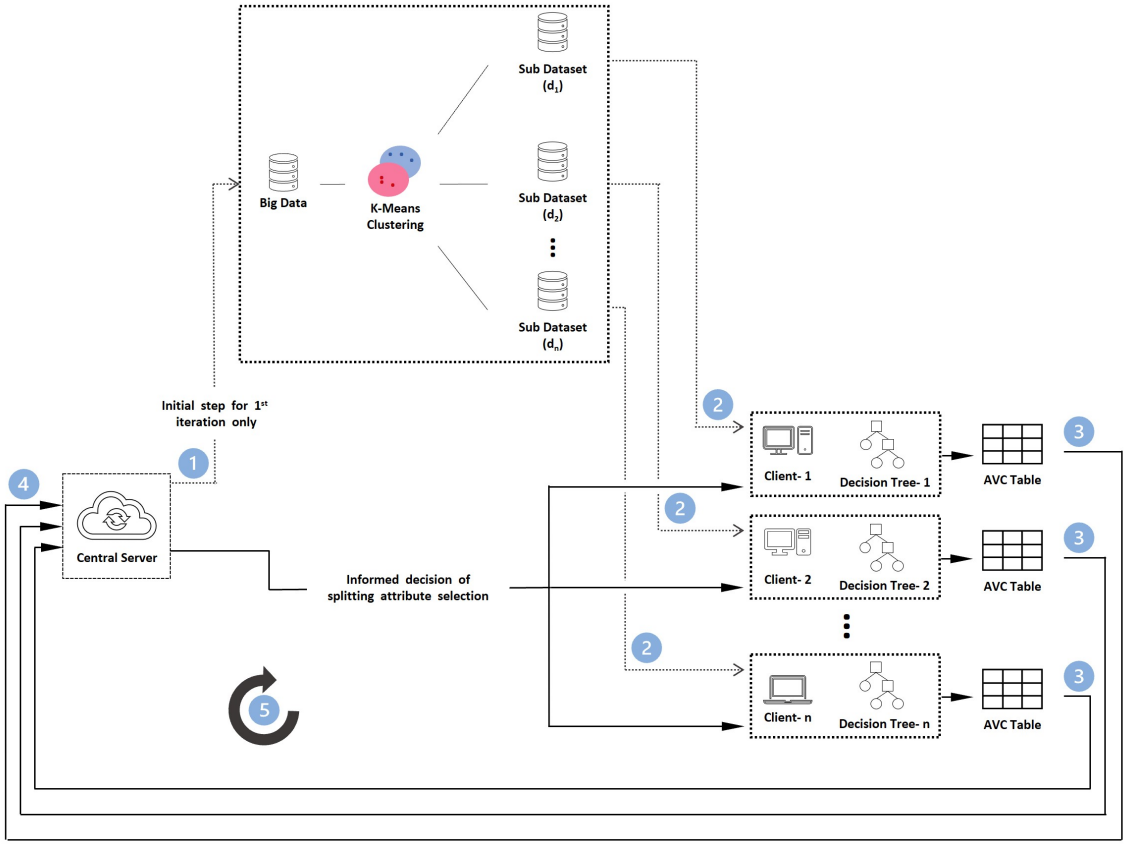


Figure 3.1: First Proposed Method: Implementation of the RainForest framework in Federated Learning.

Algorithm 5 Implementation of the RainForest framework in Federated Learning

Require: Number of local client devices n, big data D, number of iterations w (optional)

Initialization:
Server selects a group of local client devices, $C \in \{C_1, C_2, \ldots, C_n\}$
Server partitions D into n subsets $\{D_1, D_2, \ldots, D_n\}$
for $i = 1$ to n **in parallel do**
  Subset D_i is dispatched to client C_i
  Client C_i amalgamates D_i with its local data
end for
while Decision Tree is not fully built in each $C_i \in C$ **do**
  for each $C_i \in C$ **in parallel do**
    Formulates an AVC group for a specific node N_i
    AVC group is relayed back to the central server
  end for
  Server aggregates AVC sets from all clients by applying matrix addition, focusing on attribute-based integration
  Server employs a split selection method to evaluate every attribute and chooses the one that maximizes purity
  Server instructs C to stratify their datasets at node N_i, predicated on the classes of the selected attribute
  C partition their dataset at node N_i according to different classes of that attribute
end while

3.4 Federated Learning with Adaptive Deep Rule-Based Classifier

In this section, we have explained the methodology of our second proposed model, which incorporates an adaptive rule-based classifier in a Federated Learning environment. In the realm of supervised learning, a rule-based classifier leverages a predefined collection of IF-THEN rules to effectively address classification tasks. This framework incorporates the Decision Tree classifier as the foundational model for generating intricate rule sets conducive to classification. Fig. 3.2 provides a visual representation of the complete process inherent to the proposed method. According to this methodology, individual local clients embark on the construction of a Decision Tree algorithm, leveraging their distinct local datasets. This endeavor culminates in the formulation of classification rules, which are subsequently relayed to a central server, where they undergo a process of merging. Also, any redundancies are eliminated when there are numerous occurrences of the same classification rule. The resulting classification rules are then disseminated back to the clients, serving as the global classification rules for the ensuing iteration. From the second iteration, only the new unique rules are shared as part of the global rules by the server, which are merged with existing rules on all local devices. This cyclical procedure persists until predetermined criteria are satisfied. So the proposed model updates the classification rules and adapts the new concepts over time.

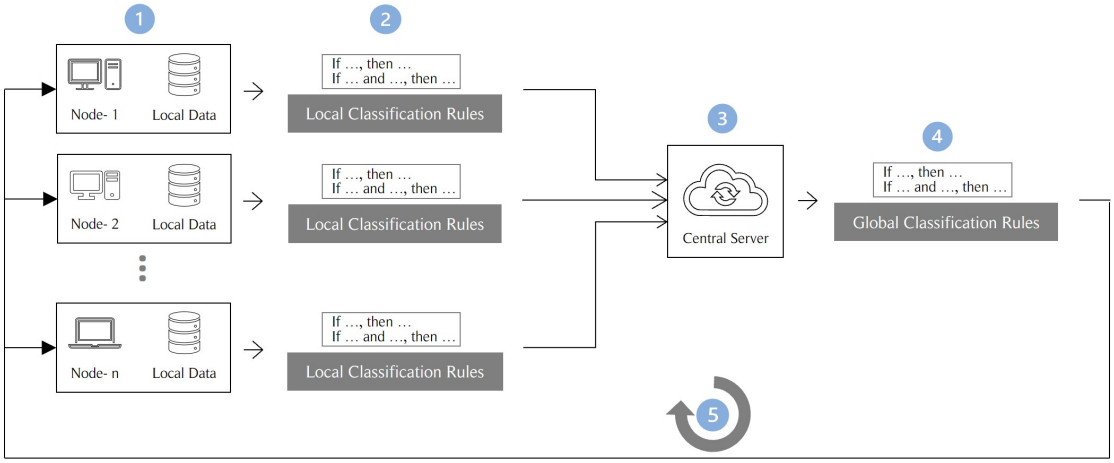


Figure 3.2: Second Proposed Method: Federated Learning with Adaptive Deep Rule-Based Classifier.

3.4.1 Federated Learning Process

The algorithm delineated in the pseudocode 6 orchestrates the federated learning process designed to employ in our proposed method. The procedure commences by setting the number of communication rounds (t) and specifying a set of participating client devices, $C_1, C_2, \ldots, C_k$. Then, two empty sets are initialized at the central server, M and $UniqueRules$, where $UniqueRules$ is an empty set for storing unique classification rules. Then, the process continues in a loop, iterating through T communication rounds. During each round t, every client device C_i operates in parallel. At the first iteration ($t = 1$), an empty set $Rules^{(t)}$ is initialized at the central server. Then, each client C_i extracts its initial set of local classification rules $m_i^{(t)}$ using the Classification Rules Extraction algorithm, applied to its local dataset. These rules are then transmitted to the central server, where they are accumulated into the set $Rules^{(t)}$. For each rule r in the $Rules^{(t)}$ set, the server checks whether r is already in the set $UniqueRules$. If r is not present, it is added both to $UniqueRules$ and M. If r is already in $UniqueRules$, it is considered redundant and discarded. For subsequent iterations ($t > 1$), the $Rules^{(t)}$ set is emptied to prepare for new data at first. Then, each client receives the M set and updates its rules $m_i^{(t)}$ by incorporating new rules from M and refining existing ones. After updating, the clients transmit their updated rules back to the server to be accumulated into the set $Rules^{(t)}$. The server then empties the set M to prepare for the updated compilation of rules. As in the first iteration, for each rule r in the $Rules^{(t)}$ set, the server checks whether r is already in the set $UniqueRules$. If r is not present, the server adds it to both $UniqueRules$ and M. If present, it is discarded. The final outcome of this iterative, dynamic process is a refined set of global classification rules, representing a cumulatively enriched and updated classification model that integrates insights from diverse data environments.

Algorithm 6 Federated Learning with Adaptive Deep Rule-Based Classifier

Federated Learning Process:

Set the number of communication rounds, T
Define a set of client devices, $\{C_1, C_2, \ldots, C_k\}$
Initialize an empty set at the central server, $M \leftarrow \{\}$
Initialize an empty set at the central server, $UniqueRules \leftarrow \{\}$
for each communication round $t = 1$ **to** T **do**
  if communication round $t = 1$ **then**
    Initialize an empty set at the central server, $Rules^{(t)} \leftarrow \{\}$
  else
    Empty the $Rules^{(t)}$ set at the central server, $Rules^{(t)} \leftarrow \{\}$
  end if
  for each client $C_i \in \{C_1, C_2, \ldots, C_k\}$ **in parallel do**
    if communication round $t = 1$ **then**
      $m_i^{(t)} \leftarrow$ ***Classification Rules Extraction***$(C_i.\text{data})$
      Transmit $m_i^{(t)}$ to the central server
    else
      Receive the M set
      Update $m_i^{(t)}$ by incorporating new rules from M and refining existing ones
      Transmit $m_i^{(t)}$ to the central server
    end if
  end for
  $Rules^{(t)} \leftarrow \sum m^{(t)}$
  if communication round $t \neq 1$ **then**
    Empty the M set at the central server, $M \leftarrow \{\}$
  end if
  for each rule r **in** $Rules^{(t)}$ **do**
    if $r \notin UniqueRules$ **then**
      $UniqueRules \leftarrow UniqueRules \cup \{r\}$
      $M \leftarrow M \cup \{r\}$
    else
      r is discarded
    end if
  end for
end for

3.4.2 Rules Extraction

The algorithm presented in 7 describes the procedure for extracting classification rules from a decision tree structure, which is initiated by initialising an empty set *LocalRule* and calling the *ExtractRule* function starting from the root node of the tree. This *ExtractRule* function is recursive and operates by checking whether the current node is a leaf. If it is, the rule accumulated so far (denoted as *CurrentRule*) is added to a set of rules called *LocalRule*. If the node is not a leaf, indicating that further subdivisions of the decision space exist, the algorithm iterates over each child node (branch). For each branch, it constructs a condition that represents the decision criterion at that branch. This condition is appended to the *CurrentRule* to form a new rule *NewRule*, which incorporates

all previous conditions leading to this branch. The function then recursively calls itself with the new branch and the updated rule, thereby traversing the entire decision tree. The accumulation of rules continues until all paths through the tree have been explored, culminating in a comprehensive set of rules stored in $LocalRule$, which represent all the decision paths from the root to each leaf of the tree. This method efficiently encapsulates the decision-making process of the tree into a set of explicit rules, facilitating the interpretation and application of the model in decision scenarios.

Algorithm 7 Classification Rules Extraction

Initialize $LocalRule \leftarrow \{\}$
Call $ExtractRule$ function starting from the root node
return $LocalRule$

FUNCTION ExtractRule ($node, CurrentRule$):

if *node* is a leaf node **then**
  Append $CurrentRule$ to $LocalRule$
else
  for each *branch* in *node.children* **do**
    $condition \leftarrow$ getCondition($branch$)
    $NewRule \leftarrow CurrentRule + [condition]$
    Recursively call $ExtractRule(branch, NewRule)$
  end for
end if

Chapter 4

Implementation and Results

This chapter presents a comprehensive overview of the datasets employed in both of the proposed models, accompanied by concise descriptions elucidating their relevance. Furthermore, it delineates the environmental setup utilized for conducting the experiments, providing a detailed exposition to facilitate reproducibility. The selection and rationale behind the chosen evaluation metric are expounded upon, underscoring its suitability for assessing the efficacy of the proposed methodology. Afterwards, the results of the output are provided in a complete manner.

4.1 Dataset

To evaluate the performance of the first proposed model 3.3, five discrete datasets were procured from the KAGGLE [68] website. Each dataset serves a unique purpose, showcasing the model's versatility across different domains and types of data. The detailed description of the selected datasets is provided in Table 4.1.

Table 4.1: Dataset description for the first proposed methodology 3.3.

Dataset	**No. of attribute**	**Attribute type**	**No. of instance**	**No. of output**
Species selection of Penguin [69]	17	Mixed	344	3
Species selection of Iris flowers [70]	4	Mixed	150	3
Identification of stroke-prone patients [71]	11	Mixed	5110	2
Survival prediction in Titanic accident [72]	27	Real	1309	2
Type diagnosis of Wisconsin breast cancer [73]	31	Real	569	2

Similarly, as with the first model, ten authentic benchmark datasets collected from the KAGGLE website [68], including four datasets collected for the first model, were taken to evaluate the performance of the second proposed model 3.4. A succinct depiction of each dataset is provided in Table 4.2.

Table 4.2: Dataset description for the second proposed methodology 3.4.

Dataset	**No. of attribute**	**Attribute type**	**No. of instance**	**No. of class**
Wine type detection [74]	13	Mixed	178	3
Bill authenticity detection [75]	4	Real	1372	2
Species selection of Iris flowers [70]	4	Mixed	150	3
Classification of Pumpkin seeds [76]	12	Mixed	2500	2
Identification of stroke-prone patients [71]	11	Mixed	5110	2
Survival prediction in Titanic accident [72]	27	Real	1309	2
Potential loan purchaser identification [77]	13	Real	5000	2
Risk level prediction of COVID-19 patients [78]	20	Mixed	1048576	2
Type diagnosis of Wisconsin breast cancer [73]	31	Real	569	2
Gender prediction from voice characteristics [79]	20	Mixed	3168	2

4.2 Dataset Preprocessing

Dataset preprocessing was performed on all the datasets to ensure data quality and compatibility with the Federated Learning model. This process included:

- Data Cleaning: Missing values were identified using Pandas' isnull() function. Depending on the nature of the data (numerical or categorical), different strategies were applied.
- Exploratory Data Analysis: Exploratory data analysis was conducted to gain insights into the datasets' characteristics and underlying patterns. Summary statistics such as mean, median, standard deviation, and percentiles were calculated. Visualizations, including histograms, box plots, scatter plots, and correlation matrices, were created to explore distributions, identify outliers, and assess relationships between variables.
- Feature Engineering: Feature engineering involved identifying relevant features that contributed most to the predictive model's performance. Techniques such as correlation analysis, feature importance ranking, and domain knowledge were employed to select a subset of informative features. Feature engineering encompasses creating new features from existing ones and transforming variables to improve model performance.
- Outlier Detection and Handling: Outliers can significantly affect model performance. For numerical data, the interquartile range (IQR) was used to detect and handle outliers.
- Handling Imbalanced Data: If the dataset exhibited class imbalance, techniques such as oversampling, undersampling, or using specialized algorithms like SMOTE (Synthetic Minority Oversampling Technique) were employed to balance the class

distribution. This ensured that the model was not biased toward the majority class and performed well across all classes.

4.3 Environment Setup

The experiments were executed on a local machine equipped with an Intel(R) Core (TM) i5-6500 CPU @3.20 GHz and 12 GB of RAM, using the Visual Studio Code IDE to implement the solution in the Python [80] programming language. To facilitate the creation of the Federated Learning Environment, the Flower framework [81] was enlisted, renowned for provisioning functionalities conducive to orchestrating Federated Learning setups. These functionalities encompass communication protocols, model aggregation techniques, as well as tools pertinent to the management and coordination of training endeavors across distributed devices. The Federated Learning configuration encompassed the establishment of a principal server alongside four local client nodes.

4.4 Performance Evaluation Metric

The validation of machine learning-driven predictive methodologies necessitates the meticulous selection of appropriate evaluation techniques and metrics. Within the context of the first proposed model 3.3, only the Accuracy [82] metric was used to evaluate the performance. It is a good metric to use when classes are balanced and misclassification errors for different classes have similar importance. A higher accuracy value reflects enhanced performance, signifying an augmented fraction of accurate predictions. Mathematically, it can be represented as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

Here,
TP= No. of accurately predicted positive samples,
TN= No. of accurately predicted negative samples,
FP= No. of negative samples mispredicted as positives,
FN= No. of positive samples mispredicted as negatives.

In the context of the second proposed model 3.4, a comprehensive suite of six established performance evaluation metrics was employed. Using multiple evaluation metrics offers a comprehensive view of performance, accounting for different error types, imbalanced datasets, and trade-offs between metrics. It ensures robust evaluation, meets diverse stakeholder requirements, aids in comparing models, aligns with application specificity, and provides insights into specific misclassifications. This multifaceted approach prevents overoptimization to one metric and ensures a model's performance is holistically assessed. The

utilized metrics encompass Accuracy [82], Precision [83], Sensitivity [84], Specificity [85], F1-score [84], and MCC (Mathews Correlation Coefficient) [86].

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.3}$$

$$Specificity = \frac{TN}{TN + FP} \tag{4.4}$$

$$F1 - score = \frac{2TP}{2TP + FP + FN} \tag{4.5}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{((TP + FP)(TP + FN)(TN + FP)(TN + FN))}} \tag{4.6}$$

4.5 Results and Discussion

4.5.1 Experimental Results of the First Proposed Model

The complexity of the first proposed model 3.3 raises doubts about its accuracy compared to traditional algorithms. Hence, in order to conduct a comprehensive comparison, the experiment was bifurcated into two distinct phases. Initially, the experiment encompassed the evaluation of their accuracy employing two traditional algorithms, specifically the Decision Tree [43] and the Random Forest [87], within a traditional machine learning framework. Subsequently, the second phase ensued, wherein the effectiveness of the proposed model was scrutinized within the context of Federated Learning. Both of the traditional algorithms and the proposed model underwent training until all conceivable partitions were executed. To evaluate the performance, 20% of each selected dataset was used as the test dataset, ensuring a robust evaluation of the model's predictive power. The evaluation was conducted utilizing test datasets on the Decision Trees generated at each local client device. For each dataset, we created separate bar charts to show the comparisons among established algorithms and the proposed model under Fig. 4.1. The values delineated under the 'Proposed Model' bar in the comparison graphs correspond to their respective average accuracy readings.

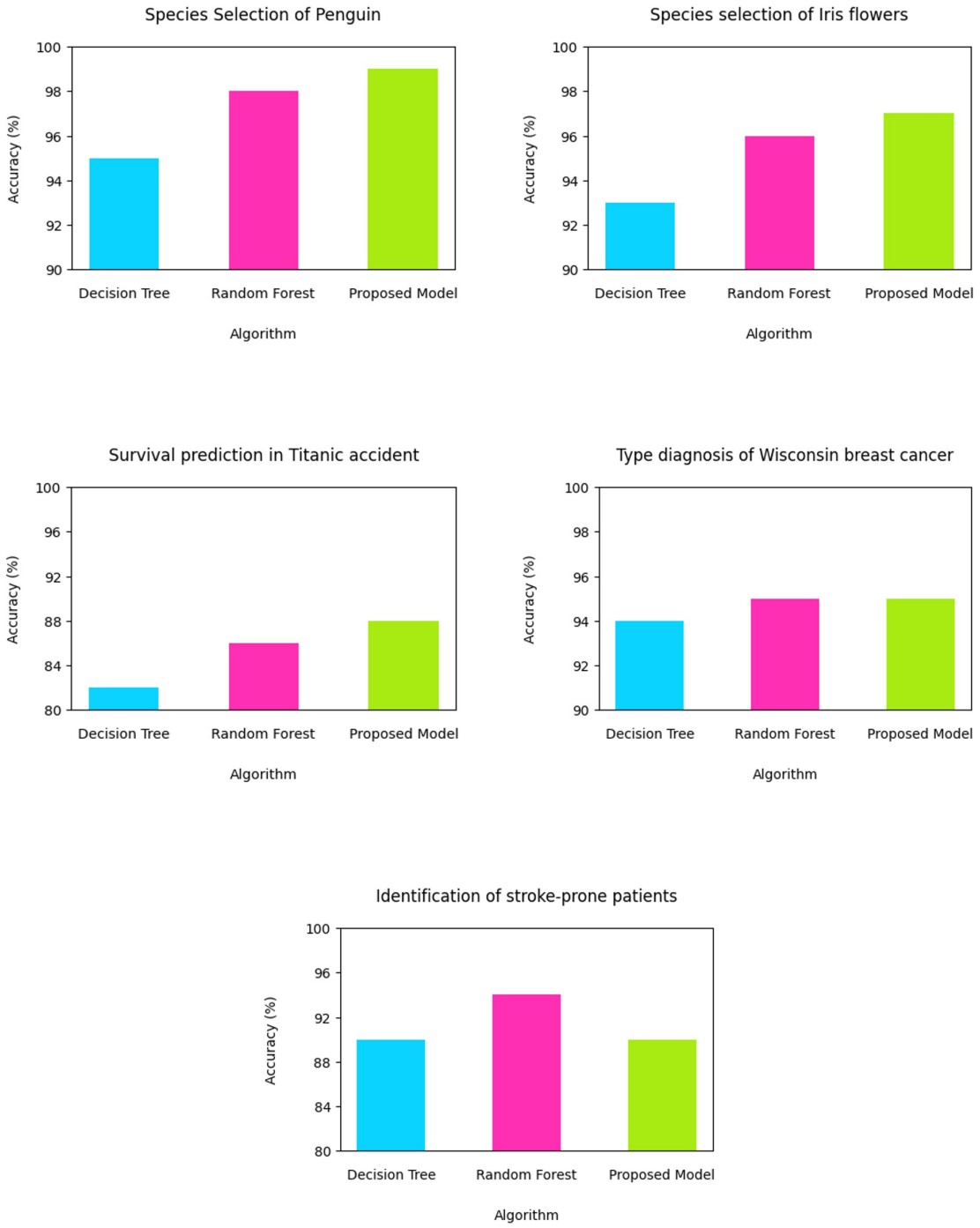


Figure 4.1: Accuracy comparison of the first proposed method 3.3.

Notably, the proposed model demonstrated superior accuracy compared to the other two models for the survival prediction in Titanic accident dataset, species selection of Iris flowers, and species selection of Penguin dataset. For the type diagnosis of Wisconsin breast cancer dataset, the proposed model yielded an enhanced accuracy rate in comparison to the Decision Tree, and it achieved the same accuracy rate as the Random Forest. However, for the identification of stroke-prone patients dataset, both the proposed model

and the Decision Tree produced the same accuracy, whereas the Random Forest achieved 4% higher. These findings indicate that the proposed model exhibits a promising approach for enhancing the accuracy of classification tasks in contrast to conventional models, particularly when dealing with big data. Even if the accuracy is equal or marginally lower, it still holds promise for privacy preservation or personalized modeling purposes that traditional algorithms cannot.

4.5.2 Experimental Results of the Second Proposed Model

To evaluate the second proposed model 3.4, we performed K-fold cross-validation [88, 89, 90, 91] on the selected datasets. K-fold cross-validation involves partitioning the original dataset into K subsets (or folds) of approximately equal size. Common choices for K are 5 or 10, but they can vary based on the size of the dataset and computational resources. In this work, the considered value of K was 5. This means the training and validation processes for every dataset were repeated five times. In each iteration, one of the five subsets was chosen as the validation set, while the remaining four subsets were used as the training set. The proposed Federated Learning model was trained on the training set from each dataset and evaluated on the validation set corresponding to the same dataset using six different performance metrics. The outcome readings of this proposed model for each iteration were documented in Table 4.3.

Table 4.3: Experimental results of the second proposed method 3.4.

Datasets	K-Fold	Accuracy (%)	Precision (WA)	Sensitivity (WA)	Specificity (WA)	F1-Score (WA)	MCC (WA)
Wine type detection [74]	1	84.3	0.718	0.834	1.000	0.776	0.756
	2	85.1	0.865	0.852	0.935	0.853	0.781
	3	85.2	0.865	0.852	0.936	0.854	0.780
	4	85.7	0.893	0.857	1.000	0.844	0.810
	5	86.3	0.922	0.863	0.919	0.873	0.813
Bill authenticity detection [75]	1	89.3	0.900	0.893	0.943	0.897	0.787
	2	91.0	0.906	0.910	0.920	0.915	0.815
	3	90.3	0.904	0.903	0.935	0.903	0.804
	4	92.0	0.932	0.920	0.845	0.921	0.852
	5	93.0	0.927	0.930	0.875	0.935	0.843
Species selection of Iris flowers [70]	1	85.3	0.936	0.853	1.000	0.854	0.802
	2	90.4	0.942	0.938	0.887	0.916	0.845
	3	94.4	0.952	0.939	1.000	0.945	0.920
	4	97.7	0.979	0.977	1.000	0.978	0.969
	5	100.0	1.000	1.000	1.000	1.000	1.000
Classification of Pumpkin seeds [76]	1	85.6	0.856	0.856	0.850	0.856	0.712
	2	87.9	0.880	0.879	0.892	0.879	0.759
	3	88.7	0.890	0.887	0.926	0.886	0.774
	4	91.1	0.911	0.911	0.919	0.912	0.819
	5	93.1	0.931	0.931	0.910	0.931	0.860
Identification of stroke-prone patients [71]	1	95.8	0.953	0.958	0.755	0.956	0.467
	2	94.3	0.897	0.953	0.618	0.927	0.552
	3	96.7	0.927	0.965	0.752	0.947	0.526
	4	96.9	0.920	0.967	0.713	0.949	0.514
	5	96.5	0.924	0.965	0.755	0.942	0.620
Survival prediction in Titanic accident [72]	1	85.4	0.853	0.861	0.677	0.838	0.642
	2	86.9	0.864	0.868	0.812	0.857	0.684
	3	89.2	0.889	0.892	0.775	0.898	0.681
	4	90.0	0.914	0.900	1.000	0.901	0.801
	5	92.1	0.921	0.921	0.910	0.921	0.821
Potential loan purchaser identification [77]	1	88.1	0.888	0.881	0.845	0.879	0.764
	2	88.9	0.910	0.889	0.815	0.890	0.798
	3	89.5	0.903	0.895	0.850	0.894	0.788
	4	89.0	0.900	0.890	0.965	0.890	0.790
	5	90.6	0.915	0.906	0.824	0.905	0.812
Risk level prediction of COVID-19 patients [78]	1	94.7	0.940	0.947	0.782	0.937	0.497
	2	94.6	0.940	0.946	0.784	0.937	0.507
	3	94.7	0.940	0.937	0.800	0.938	0.506
	4	94.7	0.941	0.947	0.787	0.938	0.505
	5	94.8	0.943	0.948	0.804	0.939	0.526
Type diagnosis of Wisconsin breast cancer [73]	1	94.2	0.951	0.942	1.000	0.943	0.880
	2	93.0	0.939	0.930	1.000	0.929	0.870
	3	83.5	0.847	0.835	0.916	0.837	0.619
	4	95.2	0.956	0.953	0.938	0.950	0.893
	5	97.1	0.970	0.971	0.977	0.971	0.937
Gender prediction from voice characteristics [79]	1	94.3	0.956	0.943	0.988	0.898	0.912
	2	95.6	0.957	0.956	0.942	0.957	0.920
	3	96.9	0.969	0.969	0.979	0.969	0.937
	4	97.4	0.975	0.974	0.973	0.973	0.945
	5	98.1	0.989	0.981	0.958	0.979	0.958

After five iterations were completed, the results from each fold were averaged to provide an overall assessment of the model's performance, which is portrayed as a heatmap in Fig. 4.2.

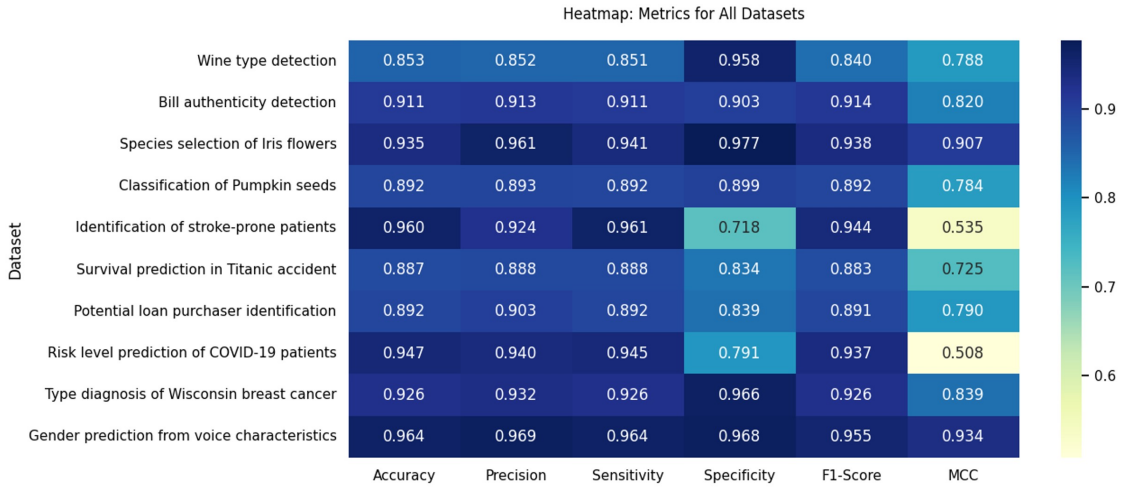


Figure 4.2: Average experimental results of the second proposed method 3.4.

This average performance score gives a more robust indication of how well the proposed model 3.4 is likely to perform on new, unseen data. From the experimental results of Fig. 4.2, it is evident that most datasets achieved strong accuracy, precision, sensitivity, and F1-score results. However, there is a notable divergence in the specificity metric across the various datasets. Datasets such as 'Species selection of Iris flowers', 'Type diagnosis of Wisconsin breast cancer', 'Gender prediction from voice characteristics', and 'Wine type detection' have exhibited notably elevated levels of specificity, indicative of a limited occurrence of false positives. Conversely, datasets like 'Identification of stroke-prone patients' and 'Risk level prediction of COVID-19 patients' have lower specificity values compared to their sensitivity. This discrepancy implies that the models employed in these datasets are less proficient at accurately discerning the risk level of COVID-19 patients or individuals at lower risk of stroke, potentially leading to erroneous positive identifications. Given the observed deficit in specificity relative to sensitivity, the MCC values for these datasets tend to reside on the slightly lower side of the spectrum, signifying the potential presence of class imbalances or other factors that might be impinging upon the overall quality of classification.

Chapter 5

Conclusion and Future Work

This chapter provides a summary of both the proposed models put forth in this study. Additionally, it elucidates certain limitations inherent in both models and underscores potential avenues for refinement in subsequent research endeavors.

5.1 Summary

- The first methodology 3.3 proposed in this study demonstrates the integration of two distinct concepts, Federated Learning and RainForest, which, when utilized in a pair, are more effective in terms of accuracy, communication cost, and privacy. An experiment was conducted with five datasets to test and compare the model's performance with other models implemented in a conventional manner. In general, the proposed method showed higher accuracy in almost every dataset.

- The second methodology 3.4 presented a new concept that primarily helps to handle users' data privacy with other mediating advantages by using classification rules as parameters in the Federated Learning environment. An extensive experiment was conducted with ten different datasets to test and analyze their performance. In general, the performance of the proposed method was exceptional across nearly all evaluation metrics.

5.2 Limitation

- While experimenting with the first proposed method 3.3, the large dataset was divided equally among local devices under the assumption of uniform computational capabilities across all devices. Consequently, apprehensions arise when considering the utilization of local devices with disparate computational capacities in practical scenarios, particularly regarding the performance of those with lesser computational power.

- Within the framework of the second proposed model 3.4, a definitive decision re-

garding the resolution of conflicting rules at the central server remains pending. This unresolved aspect necessitates further consideration to ensure the efficacy and integrity of the model in operational contexts.

5.3 Future Work

- With regard to the first model 3.3, a question arises regarding the optimal allocation of significant data volumes across devices with varying computational capabilities. Determining the ideal distribution ensures that each device can effectively train the model with its allocated data, thus facilitating future work on the trained model.
- As discussed in the limitation section about the potential conflicts arising from inconsistent rules across different local nodes in the second proposed model 3.4, investigation is required for achieving consistency throughout the Federated Learning process.
- A prospective avenue in the second proposed model 3.4 involves the systematic manipulation of classification rules to tailor models to the unique requirements of individual local client nodes, thus enabling more customized model outcomes.

By embracing these directions, subsequent iterations of this work could uncover novel insights and solutions that are even more efficient, effective, and compatible.

References

[1] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[2] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys*, 55(6):1–29, 2022.

[3] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[4] Dewan Md Farid, Mohammad Abdullah Al-Mamun, Bernard Manderick, and Ann Nowe. An adaptive rule-based classifier for mining big biological data. *Expert Systems with Applications*, 64:305–316, 2016.

[5] Sheng Shen, Tianqing Zhu, Di Wu, Wei Wang, and Wanlei Zhou. From distributed machine learning to federated learning: In the view of data privacy and security. *Concurrency and Computation: Practice and Experience*, 34(16):e6002, 2022.

[6] Felipe González-Pizarro, Andrea Figueroa, Claudia López, and Cecilia Aragon. Regional differences in information privacy concerns after the facebook-cambridge analytica data scandal. *Computer Supported Cooperative Work (CSCW)*, 31(1):33–77, 2022.

[7] General data protection regulation, available link:Official Website, last access: 20-08-2023.

[8] Health insurance portability and accountability act of 1996, available link:Official Website, last access: 20-08-2023.

[9] California consumer privacy act, available link:Official Website, last access: 20-08-2023.

[10] Personal data protection act , available link:Official Website, last access: 20-08-2023.

[11] The personal information protection and electronic documents act, available link:Official Website, last access: 20-08-2023.

[12] Lei geral de proteção de dados, available link:Official Website, last access: 20-08-2023.

[13] Personal data protection law, available link:Official Website, last access: 20-08-2023.

[14] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *arXiv e-prints*, pages arXiv–1602, 2016.

[15] Sumudu Samarakoon, Mehdi Bennis, Walid Saad, and Mérouane Debbah. Distributed federated learning for ultra-reliable low-latency vehicular communications. *IEEE Transactions on Communications*, 68(2):1146–1159, 2019.

[16] JianFei Zhang and YuChen Jiang. A data augmentation method for vertical federated learning. *Wireless Communications and Mobile Computing*, 2022, 2022.

[17] Wensi Yang, Yuhang Zhang, Kejiang Ye, Li Li, and Cheng-Zhong Xu. Ffd: A federated learning based method for credit card fraud detection. In *Big Data–BigData 2019: 8th International Congress, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 8*, pages 18–32. Springer, 2019.

[18] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*, 2020.

[19] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[20] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of machine learning and systems*, 1:374–388, 2019.

[21] Caner Korkmaz, Halil Eralp Kocas, Ahmet Uysal, Ahmed Masry, Oznur Ozkasap, and Baris Akgun. Chain fl: Decentralized federated machine learning via blockchain. In *2020 Second international conference on blockchain computing and applications (BCCA)*, pages 140–146. IEEE, 2020.

[22] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[23] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[24] Qinbin Li, Zeyi Wen, and Bingsheng He. Practical federated gradient boosting decision trees. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 4642–4649, 2020.

[25] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[26] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[27] Christopher Briggs, Zhong Fan, and Peter 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)*, pages 1–9. IEEE, 2020.

[28] Madhura Joshi, Ankit Pal, and Malaikannan Sankarasubbu. Federated learning for healthcare domain-pipeline, applications and challenges. *ACM Transactions on Computing for Healthcare*, 3(4):1–36, 2022.

[29] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.

[30] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

[31] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. On-device federated learning via blockchain and its latency analysis. *arXiv preprint arXiv:1808.03949*, 2018.

[32] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98, 2021.

[33] Liu Yang, Chen Tianjian, and Yang Qiang. Secure federated transfer learning. *arXiv preprint arXiv:1812.03337*, 2018.

[34] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[35] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1653–1664. IEEE, 2005.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[37] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via randomized quantization and encoding. *Advances in Neural Information Processing Systems 30*, 3:1710–1721, 2018.

[38] Nikko Ström. Scalable distributed dnn training using commodity gpu cloud computing. 2015.

[39] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on operating systems design and implementation (OSDI 14)*, pages 583–598, 2014.

[40] Can Karakus, Yifan Sun, Suhas Diggavi, and Wotao Yin. Redundancy techniques for straggler mitigation in distributed optimization and learning. *The Journal of Machine Learning Research*, 20(1):2619–2665, 2019.

[41] Lior Rokach and Oded Maimon. Decision trees. *Data mining and knowledge discovery handbook*, pages 165–192, 2005.

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

[43] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[44] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[45] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4:161–186, 1989.

[46] PAUL E. UTGOFF. Id5: An incremental id3. In John Laird, editor, *Machine Learning Proceedings 1988*, pages 107–120. Morgan Kaufmann, San Francisco (CA), 1988.

[47] J Ross Quinlan. *Programs for Machine Learning C4. 5*. Morgan Kaufmann, 1993.

[48] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. A method for classification of network traffic based on c5. 0 machine learning algorithm. In *2012 international conference on computing, networking and communications (ICNC)*, pages 237–241. IEEE, 2012.

[49] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. Cart. *Classification and regression trees*, 1984.

[50] Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997.

[51] Wei-Yin Loh and Nunta Vanichsetakul. Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, 83(403):715–725, 1988.

[52] Gordon V Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 29(2):119–127, 1980.

[53] Douglas M Hawkins, S Stanley Young, and Andrew Rusinko III. Analysis of a large structure-activity data set using recursive partitioning. *Quantitative Structure-Activity Relationships*, 16(4):296–302, 1997.

[54] Hyunjoong Kim and Wei-Yin Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454):589–604, 2001.

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

[56] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[57] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[58] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[59] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[60] Yoav Freund, Robert Schapire, and Naoki Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[61] Dewan Md Farid, Li Zhang, Chowdhury Mofizur Rahman, M Alamgir Hossain, and Rebecca Strachan. Hybrid decision tree and naïve bayes classifiers for multi-class classification tasks. *Expert systems with applications*, 41(4):1937–1946, 2014.

[62] Changxing Shang, Min Li, Shengzhong Feng, Qingshan Jiang, and Jianping Fan. Feature selection via maximizing global information gain for text classification. *Knowledge-Based Systems*, 54:298–309, 2013.

[63] Chowdhury Mofizur Rahman, Dewan Md Farid, Nouria Harbi, Emna Bahri, and Mohammad Zahidur Rahman. Attacks classification in adaptive intrusion detection using decision tree. 2010.

[64] Yue Liu, Lingjie Hu, Fei Yan, and Bofeng Zhang. Information gain with weight based decision tree for the employment forecasting of undergraduates. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 2210–2213. IEEE, 2013.

[65] Heba Ezzat Ibrahim, Sherif M Badr, and Mohamed A Shaheen. Adaptive layered approach using machine learning techniques with gain ratio for intrusion detection systems. *arXiv preprint arXiv:1210.7650*, 2012.

[66] Johannes Gehrke, Raghu Ramakrishnan, and Venkatesh Ganti. Rainforest—a framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery*, 4:127–162, 2000.

[67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[68] Kaggle, available link:dataset source, last access: 20-08-2023.

[69] Palmer archipelago (antarctica) penguin dataset, available link:Dataset5, last access: 20-02-2023.

[70] Iris species dataset, available link:Dataset2, last access: 20-02-2023.

[71] Stroke prediction dataset, available link:Dataset4, last access: 20-02-2023.

[72] Titanic dataset, available link:Dataset1, last access: 20-02-2023.

[73] Breast cancer dataset, available link:Dataset3, last access: 20-02-2023.

[74] Wine dataset, available link:Dataset6, last access: 10-11-2023.

[75] Bill authentication dataset, available link:Dataset7, last access: 10-11-2023.

[76] Pumpkin seeds dataset, available link:Dataset10, last access: 10-11-2023.

[77] Personal loan dataset, available link:Dataset9, last access: 10-11-2023.

[78] Covid-19 dataset, available link:Dataset11, last access: 10-11-2023.

[79] Gender voice prediction dataset, available link:Dataset8, last access: 10-11-2023.

[80] Python programming, available link:Official Website, last access: 25-02-2023.

[81] Flower framework, available link:Official Website, last access: 25-02-2023.

[82] Alif Choyon, Ashiqur Rahman, Md Hasanuzzaman, Dewan Md Farid, and Swakkhar Shatabda. Presa2i: incremental decision trees for prediction of adenosine to inosine rna editing sites. *F1000Research*, 9:262, 2020.

[83] Sajid Ahmed, Farshid Rayhan, Asif Mahbub, Md Rafsan Jani, Swakkhar Shatabda, and Dewan Md Farid. Liuboost: locality informed under-boosting for imbalanced data classification. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2*, pages 133–144. Springer, 2019.

[84] Dewan Md Farid, Ann Nowe, and Bernard Manderick. Ensemble of trees for classifying high-dimensional imbalanced genomic data. In *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 1*, pages 172–187. Springer, 2018.

[85] Farshid Rayhan, Sajid Ahmed, Swakkhar Shatabda, Dewan Md Farid, Zaynab Mousavian, Abdollah Dehzangi, and M Sohel Rahman. idti-esboost: identification of drug target interaction using evolutionary and structural features with boosting. *Scientific reports*, 7(1):17731, 2017.

[86] Sheikh Adilina, Dewan Md Farid, and Swakkhar Shatabda. Effective dna binding protein prediction by using key features via chou's general pseaac. *Journal of theoretical biology*, 460:64–78, 2019.

[87] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[88] Zaynab Mousavian, Sahand Khakabimamaghani, Kaveh Kavousi, and Ali Masoudi-Nejad. Drug–target interaction prediction from pssm based evolutionary information. *Journal of pharmacological and toxicological methods*, 78:42–51, 2016.

[89] Peng-Wei Hu, Keith CC Chan, and Zhu-Hong You. Large-scale prediction of drug-target interactions from deep representations. In *2016 international joint conference on neural networks (IJCNN)*, pages 1236–1243. IEEE, 2016.

[90] Hailin Chen and Zuping Zhang. A semi-supervised method for drug-target interaction prediction with consistency in networks. *PloS one*, 8(5):e62975, 2013.

[91] Dong-Sheng Cao, Shao Liu, Qing-Song Xu, Hong-Mei Lu, Jian-Hua Huang, Qian-Nan Hu, and Yi-Zeng Liang. Large-scale prediction of drug–target interactions using protein sequences and drug topological structures. *Analytica chimica acta*, 752:1–10, 2012.