

Federated Learning for Wearable Sensor Data

by

Adil Akhmetov

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

NAZARBAYEV UNIVERSITY

April 2024

© Nazarbayev University 2024. All rights reserved.

Author
Department of Computer Science
April 30, 2024

Certified by
Adnan Yazici
Professor and Chair
Thesis Supervisor

Certified by
Ben Tyler
Associate Professor
Thesis Supervisor

Accepted by
Yelyzaveta Arkhangelsky
Dean, School of Engineering and Digital Sciences

Federated Learning for Wearable Sensor Data

by

Adil Akhmetov

Submitted to the Department of Computer Science
on April 30, 2024, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

Digitalization has revolutionized healthcare, with technologies like electronic records, apps, and wearables improving patient care. The combination of the Internet of Things (IoT) and Artificial Intelligence (AI) has raised healthcare standards, while the concept of Quantified Self (QS) encourages self-tracking and personal development. Yet, IoT data gathered for QS is not fully utilized due to analysis challenges, with only 7% of data properly used and interoperability issues complicating data exchange. Federated Learning (FL) provides a solution by enabling collective model training across different sources while protecting data privacy, in contrast to Centralized Learning (CL) which centralizes data. Overcoming interoperability is essential for effective data use, but complexities in digital health frameworks make this difficult. Thus, this work aims to investigate the use of FL, leading to the development of the web application that enables to conduct a predictive analysis of interoperable data from wearable devices and sensors in healthcare. The primary objectives of the work include: (1) Achieving data interoperability via FHIR protocol; (2) Evaluating the effectiveness of FL and CL in analyzing interoperable data; (3) Creating a user-friendly application using AutoML to train ML models on interoperable data in FL and CL approaches. In this work multiple datasets were analyzed by ML algorithms using CL and FL, and the results were evaluated using relevant metrics. Subsequently, a solution that leverages data interoperability is presented. The findings reveal that FL learning is more preferable than CL for preserving interoperable data confidentiality. The implementation and evaluation demonstrate the effectiveness of the web application in conducting the predictive analysis of centralized data.

Thesis Supervisor: Adnan Yazici
Title: Professor and Chair

Thesis Supervisor: Ben Tyler
Title: Associate Professor

Acknowledgments

I want to express my sincere gratitude to my supervisors, Professors Adnan Yazici and Ben Tyler, for their help and advice during my project. Throughout the writing process, they worked with me, showing me how to pick a topic when I only had a rough idea, and helping me grow that idea into a real thesis.

I am also very grateful to Professor Latif Zohaib for teaching me how to write a good thesis proposal and for going through so many versions of it with me.

Additionally, I am deeply thankful to Professor Enver Ever for taking on the role of my external committee member and for the time invested in reading my work.

Special thanks go to my family and friends for supporting me while I was writing my thesis.

Contents

1	Introduction	15
2	Related works	19
2.1	Wearable devices and IoT	19
2.2	Quantified-Self	20
2.3	Federated Learning	22
2.4	Electronic Health Records	23
3	Methodology	25
3.1	System Design	25
3.1.1	Centralized vs Federated Learning	25
3.2	System Implementation	27
3.2.1	Principles	27
3.2.2	Programming stack	27
3.2.3	Proposed solution	30
3.3	Dataset	44
3.3.1	Classification	44
3.3.2	Regression	45
4	Results	47
4.1	Data partitioning	48
4.2	Time Metrics Evaluation	49
4.3	Analysis of Real Data	50

4.3.1	Classification	51
4.3.2	Regression	62
4.4	Analysis of Synthetic Data	69
4.4.1	Classification	69
4.4.2	Regression	72
4.5	Discussion	75
5	Conclusion	79
5.1	Future Works	81

List of Figures

3-1	Centralized Learning Design	26
3-2	Federated Learning Design	26
3-3	Hexagon analysis of Flower: Pros and Cons [43]	29
3-4	The diagram of the proposed solution.	30
3-5	Upload page of the GUI with uploaded data from MiFit.	31
3-6	DataSource class diagram with MiBand and CSV implementations. .	33
3-7	Calories burned in the FHIR format.	34
3-8	The example of Classification model training in PyCaret.	35
3-9	Modelling section of the GUI with uploaded data from Apple Watch.	36
3-10	Finalize model section of the GUI.	36
3-11	Sequence of methods call in Flower [54].	38
3-12	The pseudocode for the FedAvg algorithm [55].	39
3-13	The final overview of the application.	41
3-14	Federated learning section of the GUI.	44
4-1	Evaluation of Centralized Learning	48
4-2	Evaluation of Federated Learning	49
4-3	Generation of Synthetic Data	69
4-4	Data Quality of Classification dataset using SMOTE.	70
4-5	Classification: SMOTE vs Real data distribution.	70
4-6	Data Quality of Regression dataset using ADASYN.	73
4-7	Regression: ADASYN vs Real data distribution.	73

List of Tables

2.1	Overview of the proposed methods	23
3.1	Observation resource structure	33
3.2	Evaluation Metrics for Classification	42
3.3	Evaluation Metrics for Regression	43
4.1	MacBook Pro 14 Characteristics	47
4.2	Apple Watch Dataset Statistics	51
4.3	AW Dataset 60%/40% partition, $k = 2$	52
4.4	AW Dataset 60%/40% partition, $k = 4$	52
4.5	AW Dataset 60%/40% partition, $k = 8$	53
4.6	AW Dataset 70%/30% partition, $k = 2$	53
4.7	AW Dataset 70%/30% partition, $k = 4$	54
4.8	AW Dataset 70%/30% partition, $k = 8$	55
4.9	AW Dataset 80%/20% partition, $k = 2$	55
4.10	AW Dataset 80%/20% partition, $k = 4$	56
4.11	AW Dataset 80%/20% partition, $k = 8$	56
4.12	AW Dataset 90%/10% partition, $k = 2$	57
4.13	AW Dataset 90%/10% partition, $k = 4$	57
4.14	AW Dataset 90%/10% partition, $k = 8$	58
4.15	HARTH Dataset Statistics	59
4.16	HARTH Dataset 60%/40% partition, $k = 2$	59
4.17	HARTH Dataset 60%/40% partition, $k = 4$	60
4.18	HARTH Dataset 60%/40% partition, $k = 8$	60

4.19	HARTH Dataset 70%/30% partition, $k = 2$	61
4.20	HARTH Dataset 70%/30% partition, $k = 4$	61
4.21	MiBand Dataset Statistics	62
4.22	MiBand Dataset 60%/40% partition, $k = 2$	63
4.23	MiBand Dataset 60%/40% partition, $k = 4$	63
4.24	MiBand Dataset 60%/40% partition, $k = 8$	64
4.25	MiBand Dataset 70%/30% partition, $k = 2$	65
4.26	MiBand Dataset 70%/30% partition, $k = 4$	65
4.27	MiBand Dataset 70%/30% partition, $k = 8$	66
4.28	DAT263x Dataset Statistics	67
4.29	DAT263x Dataset 60%/40% partition, $k = 2$	67
4.30	DAT263x Dataset 60%/40% partition, $k = 4$	68
4.31	DAT263x Dataset 60%/40% partition, $k = 8$	68
4.32	Apple Watch using SMOTE Dataset Statistics	71
4.33	AW Synthetic Dataset 60%/40% partition, $k = 2$	71
4.34	AW Synthetic Dataset 60%/40% partition, $k = 8$	72
4.35	MiBand Dataset using ADASYN Statistics	74
4.36	MiBand Synthetic Dataset 60%/40% partition, $k = 2$	74
4.37	MiBand Synthetic Dataset 60%/40% partition, $k = 8$	75
4.38	Comparison between Federated and Centralized Learning	77

List of Abbreviations

CL Centralized Learning. 17

EHR Electronic Health Record. 17

FHIR Fast Healthcare Interoperability Resources. 16, 17

FL Federated Learning. 17, 18

IoT Internet of Things. 15–17

ML Machine Learning. 16, 17

QS Quantified-Self. 17

Chapter 1

Introduction

The digitalization of the modern world has led to a big leap in technological advances across diverse aspects of human life. Currently, numerous devices interconnect via the Internet to form an intricate system known as the IoT [1]. These systems effectively address various issues where healthcare emerges as one crucial area requiring attention and innovative solutions [2]. The digitalization of healthcare offers tremendous potential in health quality improvement, facilitated by electronic mobile health apps, medical imaging, medical records, low-cost gene sequencing, and the proliferation of sensors and wearable devices [3] [4]. When combined with cutting-edge technologies such as Artificial Intelligence, big data analytics, and cloud computing, this wealth of digital health data holds promise for improving the lives of countless patients worldwide [3]. One of the ways to achieve one individual's healthcare digitalization is through the use of wearable devices. Wearable devices support users in monitoring their weight, calories, sleep, physical activities, heart rate, body temperature, and other vital signs [5] [6] [7]. Today the wearable technology market is in its golden era, making them highly affordable [5] [6] [8], thus allowing many people interested in digitally monitoring their health and behavior emphasizing personal improvement through self-tracking to form a community of enthusiasts known as Quantified Self (QS) [5] [8] [9]. The term generally refers to anything that can be measured [7]. Recently, the increasing interest of academia towards QS has resulted in the formation of an interdisciplinary research field called personal informatics. This field necessitates

the collaboration of various disciplines, encompassing consumer informatics, health informatics, ubiquitous computing, and Human-Computer Interaction (HCI) [9].

Despite the rapid evolution of wearable devices, the analysis of large quantities of QS data remains a challenge. The QS practice is still primarily focused on data collection, while data analysis is restricted to basic visualization and correlation analysis [9]. Only 7% of IoT devices are used in related applications [2]. Thus, the data retrieved by IoT devices is used ineffectively. More effective use of such an enormous amount of QS data requires ML algorithms [7]. However, a significant portion of medical data lacks interoperability. It is stored in isolated databases, resides in incompatible systems, and is entangled in proprietary software, making it a challenging task to share, analyze, and interpret this data. Current healthcare systems utilize a diverse range of data formats, individualized specifications, and unclear meanings. This negatively affects the progress of technologies that rely on this data, such as ML, AI, and Big Data [3] [10]. Therefore, data interoperability is essential to unlocking the complete capabilities of smart and interconnected healthcare, which has the potential to enhance patient results and reduce expenses [11].

Hence, to fully leverage ML and AI algorithms and capitalize on the potential of big data analysis by providing them with the most comprehensive input, it is vital to process information from various systems and transcend institutional limits. To handle the data and derive valuable insights, both syntactic and semantic interoperability are necessary. Syntactic interoperability defines the data format and structure. One of the protocols that helps to ensure syntactic interoperability is the Electronic Health Record (EHR). It stores patient medical information digitally [4]. OpenEHR enables healthcare practitioners and health IT specialists to delineate clinical content using archetypes, which describe clinical concepts built upon a foundational reference model. In contrast, FHIR is gaining widespread acceptance in the healthcare sector for its compatibility with creating mobile health applications operating across diverse IT platforms. Regrettably, the current digital health framework complicates handling large amounts of data across IT systems needlessly. Furthermore, executing algorithms on unorganized and non-standardized data can introduce inaccuracies in the

analysis. Even though the significance of interoperable health IT systems is gaining recognition, there is still limited awareness of this subject among healthcare practitioners, particularly when compared to other topics such as artificial intelligence, big data, and mobile technologies, which are commonly regarded as the primary catalysts for digital health innovation [3].

This work aims to develop a web application that enables to conduct a predictive analysis of interoperable data from wearable devices and sensors in healthcare. Multiple datasets are analyzed by ML algorithms trained in a Federated and Centralized manner. The results are then evaluated using relevant metrics. Subsequently, a solution that leverages data interoperability is presented. Shortly, the main objectives of this work are:

- To reach data interoperability via the FHIR protocol.
- To compare the use and performance of FL and CL in analyzing interoperable data.
- To design and develop a user-friendly application utilizing AutoML that trains ML models on interoperable data using FL and CL.

Our proposed system using federated learning improves the performance and privacy of IoT wearable sensors by enabling decentralized processes without centralizing sensitive data, thus improving privacy and interoperability, and facilitating real-time applications that use both real and synthetic data to ensure model compliance with FHIR, which results in more dynamic and secure health technology interactions.

This introductory chapter introduced the concept of digitalization in healthcare, highlighting the significance of data interoperability. The following chapter reviews the existing literature on such subtopics as Wearable devices and IoT, Quantified-Self, Electronic Health Record, and FL. The next chapter details the programming stack and the implementation of the proposed solution. Chapter 4 presents the results and compares the evaluation metrics of the models trained by CL and FL. The final chapter summarizes the work and emphasizes the importance of overcoming

interoperability challenges in healthcare by leveraging FL. Additionally, it discusses implications for future research.

Chapter 2

Related works

The chapter will focus on several topics such as Wearable devices and IoT, Quantified-Self, Electronic Health Records, and Federated Learning. These technologies are essential for a deep understanding of the problem and finding a solution.

2.1 Wearable devices and IoT

In the same way as the Internet and smartphones, wearables and IoT are rapidly changing our lives [12]. Nowadays, people share not just photos of their exercise, but also heart rate metrics, step counts, and descriptive maps of their routes, which is made possible by wearable devices. Therefore, it is important to recognize what IoT and wearables are and explore the boundaries of their application.

Wearable biosensors, commonly known as "wearables", are a wide-ranging interdisciplinary health services effort to utilize mHealth (mobile health) for enhancing data collection, diagnosis, treatment monitoring, and health insights. The Global Observatory for eHealth by the World Health Organization describes mHealth as the utilization of mobile devices such as mobile phones, patient monitoring tools, personal digital assistants, and other wireless devices [13] to aid medical and public health activities. These wearable devices are currently easily accessible to purchase as consumer goods and are being more commonly used to gather and analyze basic physiological information like weight, calorie intake, exercise, sleep patterns, body

temperature, and heart rate [5] [6] [7], and even more complex data like energy expenditure, arrhythmia, cardiovascular, continuous glucose monitoring, diabetes mellitus, and many more [13]. Due to the complexity of the field, the topics researchers study vary significantly. Below, some of the applications of wearables are discussed.

Wearable health technology has been the focus of research in recent years. Witt et al. [13] analyzed the algorithms used in popular wearable devices from 2014 to 2017, while in [14] the authors surveyed recent developments in wearable IoT. Another study [15] introduced Medical 4.0 as a significant advancement in healthcare, utilizing IoT in data collection and promoting patient-centered therapy. Pathak et al. [16] developed SemBox, a wireless system that enhances interoperability among diverse wearable health monitoring devices. The authors of another study [17] discussed the opportunities and challenges of the Internet of Medical Things (IoMT), highlighting the benefits of patient empowerment, healthcare collaboration, and data sharing. Fei and Ur-Rehman developed a low-power smart wristband that can monitor heart rate, count steps, and detect abnormal hand movements [18]. A user-end program was also created for data analysis and presentation. In a study involving 46 participants [19], the Xiaomi Mi Band 4 fitness tracker was found to be accurate and precise in tracking step count and heart rate. Another study [20] has shown that consumer sleep tracking devices such as Fitbit Charge 2 and Neuroon eye mask are good for non-clinical use, but not suitable for diagnosing sleep disorders.

2.2 Quantified-Self

Quantified-Self (QS) involves using technology to track various aspects of one's life, such as biology, physical activity, behavior, or environmental factors [21]. The term 'Quantified-Self' was first coined in 2007 by Wired writers Gary Wolf and Kevin Kelly, who founded the QS movement, a global community focused on self-tracking for gaining self-knowledge through data [7] [8]. Self-monitoring, measuring, and recording for self-improvement or reflection have historical roots, but the advent of digital technologies has sparked new interest and expanded the domains and applications of self-

tracking. This subsection discusses the practical applications of the QS community and the potential impact it may have on improving mHealth.

Swan notes the rise of the QS movement, where individuals track various aspects of their well-being [21]. The long-term vision is to optimize personal performance in real-time. QS encompasses quantitative and qualitative data and is evolving into a "qualified self" by tracking qualitative aspects and promoting behavior change. Another paper discusses the rise of self-tracking apps and devices and how they can be used to improve personal productivity [5]. A netnographic study has been conducted on the Fitbit user community to explore wearable tech's impact on the quantified self [8]. The study uncovers social engagement mechanisms, such as empowering motivation and friendly rivalry, that are built on this foundation. Billis et al. [22] explore how seniors have integrated self-tracking, and how it can be a habitual practice. They developed a web application to visualize sensor data collected over six months in real-life settings. In [23], authors introduce the concept of Personal Health Information Self-Quantification Systems (PHI-SQS) and systematically review data management processes in eleven self-quantification tools. The paper suggests that self-quantification for personal health maintenance holds promise, but further research is needed to support its use in this context. Sharon [24] studies how self-tracking devices can be used for individual health management and raises ethical concerns. She suggests a practice-based approach to reveal diverse enactments of values by self-trackers in the QS community. Three novel recommendation approaches for QS applications: Virtual Coach, Virtual Sleep Regulator, and Virtual Nurse have been proposed in [25]. These approaches aim to improve users' health conditions by offering personalized workout plans, considering the medical history and health goals for physical activity recommendations, and addressing sleep quality issues. Lastly, researchers in [6] summarize recent studies on the use of wearable devices for promoting physical activity, mental health, and health consciousness. The review identifies 25 relevant papers covering various aspects of wearable devices, including accuracy, data collection tools, and interventions.

2.3 Federated Learning

Federated learning (FL) is an innovative approach to distributed learning that addresses the growing concern of data security in the era of big data. In this method, multiple users can collaborate to share training results without compromising the privacy of their data [26]. FL was developed as a response to the limitations of centralized deep learning systems and offers a solution that eliminates the need for extensive data collection, allowing for the strict preservation of privacy standards. Unlike traditional machine learning techniques, FL allows for on-device training of machine learning models by coordinating multiple devices, known as FL clients, with a central server, referred to as an FL aggregator, without the need to share actual datasets. This method is especially important for protecting confidential data.

Tyagi et al. [26] give an outline of FL, a decentralized method for cooperative model training that guarantees data privacy. Another study [10] indicates that federated learning could offer a hopeful answer for linking and examining confidential health information from different origins. The method can assist in tackling the issues caused by privacy concerns when sharing electronic health records among various organizations. In their research, Khan et al. [27] propose Fed-Inforce-Fusion, a privacy-preserving FL-based IDS for the IoMT. The model uses reinforcement learning to find relationships in medical data and collaboratively trains a comprehensive IDS model across smart healthcare system (SHS) nodes. The systematic review in [28] on FL in healthcare highlights its promise but also the need for addressing bias risk and increasing transparency through data homogeneity and sharing of metadata and code. An FL approach for diagnosing depression while ensuring patient data privacy has been proposed in [29]. The cluster-based framework enhances standard machine learning models and outperforms other cooperative learning approaches.

Table 2.1: Overview of the proposed methods

Reference	Method	Standard	Tested on	Workflow
[30]	OmniPHR	openEHR	MIMIC-III	Enhances PHR with AI and standard ontology.
[31]	Ontology-based framework for distributed EHR environments	openEHR ADL, XML, RDB, CSV, any	Mansoura University Hospital, Mansoura, Egypt.	2 stages, 3753 axioms, SPARQL evaluation.
[32]	CancerLinQ	openEHR, FHIR	68 oncology sites, 5 EHR vendor products	Vendor commonality aids interoperability, EHR selection insights.
[33]	Hybrid Data Interoperability Method	openEHR, XML	Generated dataset	Hybrid method improves data extraction in healthcare.
[34]	Public Health Framework (PHF)	HL7 FHIR	Health Management Information System (HMIS)	Framework generates alerts, adaptable for healthcare hierarchy.
[35]	Big Data Architecture for EHRs	openEHR, HL7 FHIR	MIMIC-III	Architecture extracts data using deep learning for healthcare.
This work	Data interoperability using Federated Learning	FHIR	Wearable sensors, HARTH, DAT263x	Agents extract data from wearables and compute ML model in federated and centralized manner.

2.4 Electronic Health Records

Electronic Health Records (EHR) have revolutionized the healthcare landscape by setting common standards for health data, fostering seamless data sharing. EHR does not prescribe a specific standard but focuses on unifying health information. This pursuit of standardization is a crucial step in achieving interoperability, defined as the effective communication between different applications without data compromise [36]. EHR combines patient data for efficient sharing, as outlined by ISO standards. To bridge the gap in data exchange interoperability, various application-level standards, like OpenEHR and FHIR [2], play a vital role within IoT architectures in digital healthcare. This subsection focuses on the various applications of EHR and its importance. Table 2.1 summarizes some of the methods.

Roehrs et al. [30] propose a Personal Health Record (PHR) interoperability model named OmniPHR that utilizes a standard ontology and artificial intelligence with natural language processing (NLP) to achieve interoperability. The model was evaluated using a real database of anonymized patient records, demonstrating the feasibility of harmonizing data from various standards into a unified format. Another semantic ontology-based model that achieves interoperability in EHRs has been proposed in [31]. The model unifies data formats, accommodating five healthcare data standards and allowing physicians to interact with diverse systems through a single in-

terface. The integrated ontology facilitates improved patient care. Data from 68 oncology sites using 5 different EHR vendor products have been analyzed in [32] to quantify the interoperability of real-world EHR implementations concerning clinically relevant structured data. The results showed that intra-EHR vendor interoperability was notably higher than intersystem interoperability, highlighting the lack of standardization for clinically relevant data. In [36], Bhartiya et al. discuss the challenges of accessing and sharing EHR in team-based healthcare. They highlight issues related to privacy, data security, and interoperability caused by heterogeneity in EHR systems. The paper identifies different approaches and challenges in achieving interoperability in EHR sharing. Evans analyzes the evolution of EHR from 1992 to 2015 and speculates on their expected state in the next 25 years [37]. EHRs have come a long way, but technical, procedural, social, political, ethical, and security concerns overshadow their usage. Although current EHRs do not entirely meet the evolving needs of the healthcare landscape, emerging EHR technology will help establish international standards for interoperable applications, facilitating precision medicine and a learning health system. Daliya and Ramesh propose a hybrid approach to handle heterogeneous data from various sources in an IoT-based healthcare system [33]. The approach extracts data meaning from diverse sources while ensuring uniformity in data format, contributing to enhanced data interoperability. Another study [34] proposes a framework for managing public healthcare data based on standardized protocols. It digitizes EHR at basic health facilities, consolidates data from multiple sources, and incorporates contextual information using HL7 as an interoperability standard. Researchers from Italy developed a national IT platform for the interoperability of EHR systems [35]. They introduce a Big Data architecture for EHR that offers valuable insights for healthcare professionals, patients, and decision-makers.

This study stands out because it aims to achieve interoperability by employing FL architecture and EHR protocol. The proposed architecture will not just function with refined datasets but also wearable devices.

Chapter 3

Methodology

This chapter discusses the system design and the system implementation which consists of data sources (agents), FHIR server, web user interface, and FL server.

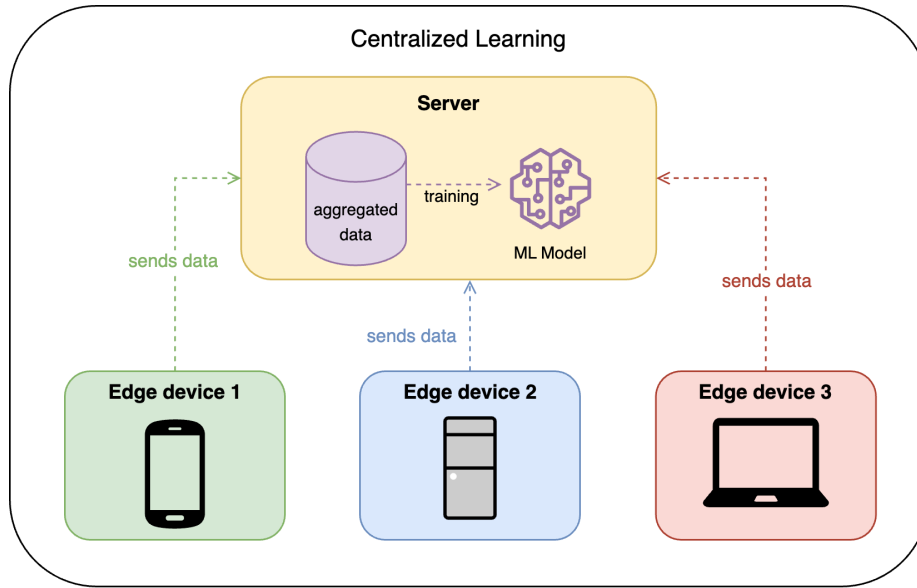
3.1 System Design

This section discusses the design of the proposed solution. The difference between CL and FL is discussed, as well as the design of the proposed solution.

3.1.1 Centralized vs Federated Learning

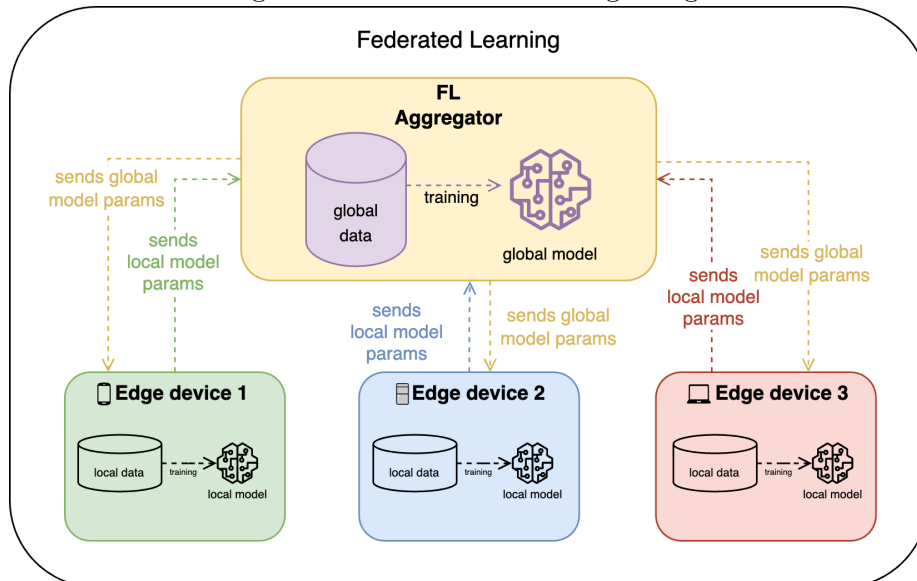
Traditionally, CL requires all edge devices share their data to some centralized node or server, which then trains the model on the aggregated data. This approach is efficient but raises privacy concerns, as the data is centralized. Figure 3-1 shows the design of CL.

Figure 3-1: Centralized Learning Design



FL, on the other hand, allows edge devices to train the model locally and share only the model updates with the FL Aggregator (or FL Server) (See Figure 3-2). The FL Aggregator then takes these model parameters and aggregate them (e.g. by averaging), converging all updates in one global model which then again is distributed among clients. This approach is more privacy-preserving, but it is more complex and may require more communication between devices.

Figure 3-2: Federated Learning Design



3.2 System Implementation

3.2.1 Principles

The philosophy of the application follows the FOSS (Free and Open Source Software) principles [38], thus, doing its best to avoid any non-FOSS dependencies. The architecture of the application is modular, such that each component is replaceable. Due to the scope of research, it has been decided to develop only a proof-of-concept (POC) solution. Thus, POC focuses on agents only for MiBand and CSV files. Since it is a POC, the solution should be low-code and highly configurable. Furthermore, to be user-friendly, POC should provide a graphical user interface (GUI). In addition, POC supports only Supervised learning: Classification and Regression. The source code will be available on GitHub under the MIT license with the following link: <https://gitfront.io/r/weeebdev/gnm1dXbMqiHh/auto-fl-fit>

3.2.2 Programming stack

This subchapter discusses the programming stack of the POC. Specifically, it focuses on open-source, low-code technologies that can be used in the development of the POC.

GUI

The GUI (Graphical User Interface) of the application is based on Streamlit [39]. Following the principles from subsection 3.2.1 "Principles" subsection, it has been decided to use an open-source Python library used for creating web applications with low-code efforts. It has been designed in way that the developer needs to write code in a straightforward manner, similar to writing Python scripts. Its simplicity lies in its reactive programming model, where changes to the code are immediately reflected in the web app, making the development and debugging process efficient. One of the main reasons to use Streamlit is its high integration with the most popular Python libraries like Pandas, Matplotlib, and PyCaret which is discussed later.

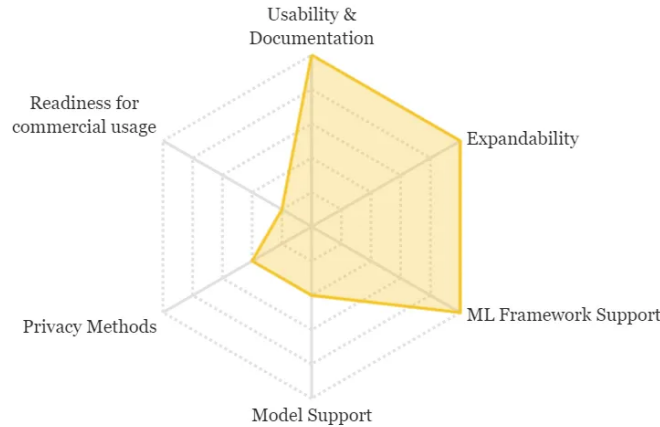
AutoML framework

Compared to other AutoML frameworks, PyCaret is distinguished for being a Python-based, open-source machine learning library with low-code capabilities that simplifies the entire machine learning model development, training, optimization, and deployment process [40]. It is designed for individuals of all skill levels, including beginners and experienced data analysts, to effortlessly utilize machine learning. This is achieved through providing an easily accessible API that simplifies the difficult components of model development. PyCaret accelerates project progress by automating tedious tasks, assisting in selecting and fine-tuning models, and providing insights on model performance and interpretation. Additionally, PyCaret is user-friendly for developers and can be easily personalized because of its simple and efficient design, demanding little effort.

Federated Learning framework

The FL framework that is used is Flower [41]. Flower addresses the challenges of implementing FL realistically regarding scale and systems heterogeneity. It enables edge devices to collaboratively learn a shared prediction model while retaining their training data locally, thus separating machine learning capabilities from the necessity to store data in the cloud [42]. The uniqueness of Flower lies in its ability to conduct large-scale FL experiments and cater to diverse edge device scenarios. It offers novel functionalities that facilitate the execution of scalable FL workloads on heterogeneous edge devices. Flower offers a crucial resource to the research community by offering new features to conduct and analyze FL experiments in various edge device environments, allowing for significant advancements in the field.

Figure 3-3: Hexagon analysis of Flower: Pros and Cons [43]



According to FL frameworks comparison [43] Flower is the best in terms of expandability and ML framework support (Figure 3-3). Additionally, it is well documented, meaning that it is highly developer-friendly. The main reason why Flower has been chosen in this project is its agnostic approach meaning that any ML framework can be utilized. Although Flower lacks when it comes to security, the issue can be solved, which will be discussed in the section 5.1 "Future Works" section.

FHIR

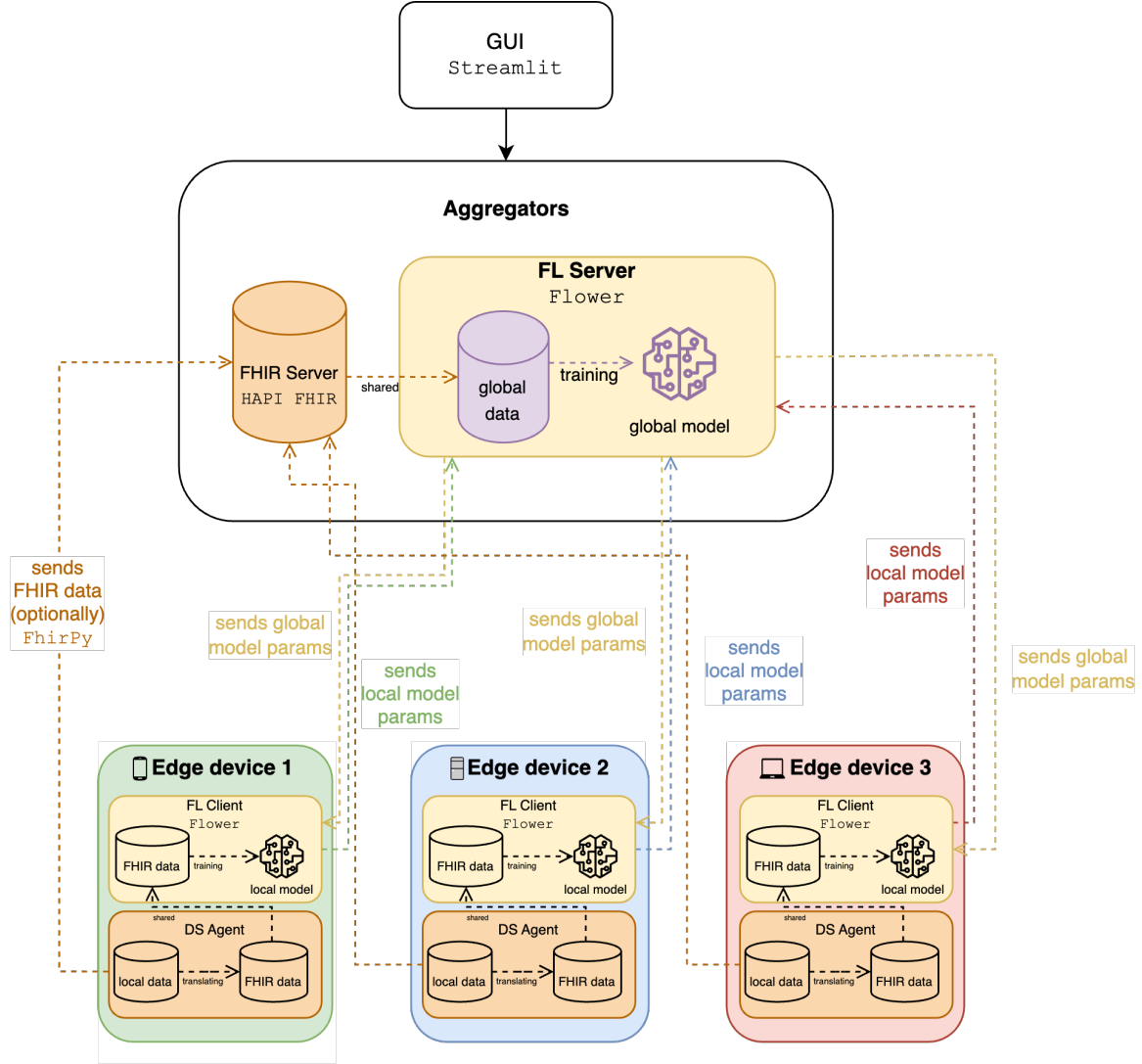
Among the numerous FHIR implementations [44] tested, HAPI FHIR [45] stands out as being highly reliable and comprehensive. HAPI FHIR is a freely available version of the FHIR standard for sharing healthcare information. It offers a comprehensive collection of tools and libraries for developers to generate, utilize, and oversee FHIR-compliant data in their healthcare apps. One major benefit of utilizing HAPI FHIR is its inclusion of a RESTful API for connecting clients through HTTP, as well as a FHIR Resource Model that accommodates various standardized properties and relationships (such as patients and observations). This interoperability in data exchange allows for smooth communication and collaboration between different sources of data, whether it be wearable devices or CSV files. HAPI FHIR acts as a server that holds and serves the data across the project.

To actually utilize the server via RESTful API, the project also needs a client.

FhirPy [46] has been chosen, because it is lightweight and easy to use, providing a declarative way of writing the queries.

3.2.3 Proposed solution

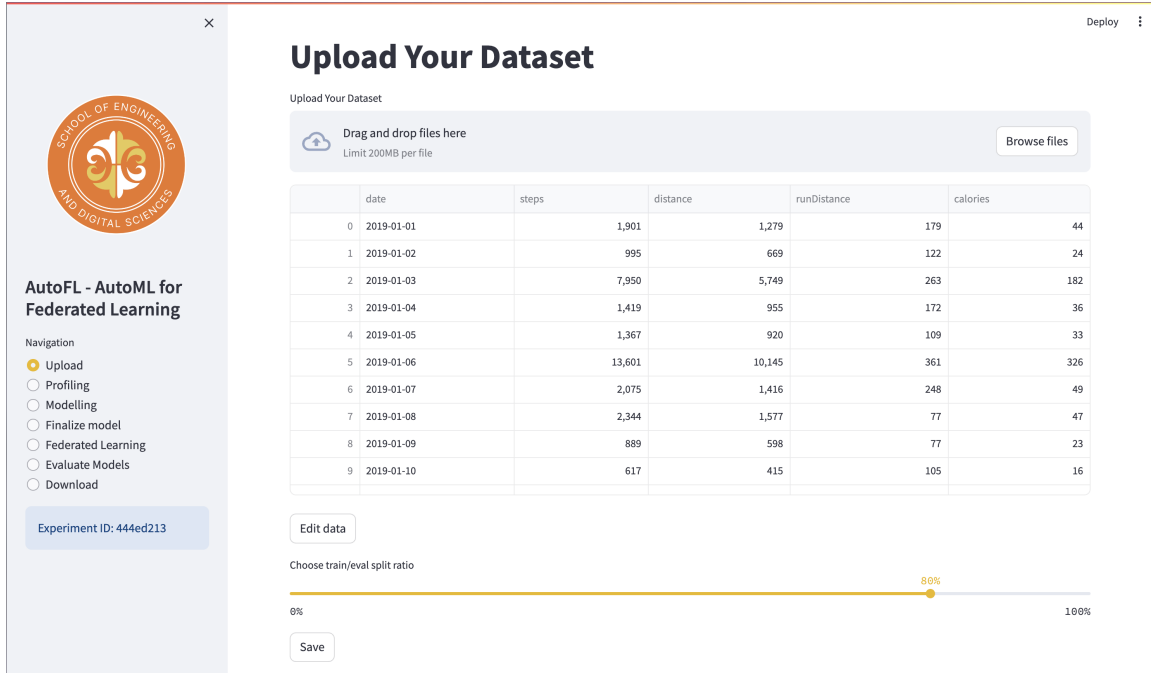
Figure 3-4: The diagram of the proposed solution.



According to the diagram represented in Figure 3-4, the project utilizes all the technologies listed above. The solution proposes data source extensions called agents that are responsible for data collection and processing. Optionally, once all data processing is complete, each node sends the output to a centralized server powered by HAPI FHIR which collects the data and serves it to GUI later. GUI is a web

interface written on Streamlit that acts as a main entry point for the user. The end user only needs to interact with the GUI to operate the process. Also, the user may upload the CSV files directly from the GUI (Figure 3-5).

Figure 3-5: Upload page of the GUI with uploaded data from MiFit.



From now on, the user may manipulate the data: edit, encode labels, and choose the split ratio for training and evaluation of the models. By default, the ratio is 80% to 20%. The reason is that those 20% data partitions are used for the evaluation of all the models, simulating the real-world case where data are continuously generated. For classification models it is mandatory to split the data in a stratified fashion, ensuring that each partition will have the same sample of data. This way the evaluation results will be honest and legit. More about that will be discussed later in chapter 4 "Results".

For each uploaded data, a new experiment ID will be generated using UUID. This ID is assigned to all types of results that the app will produce: centralized and federated models, and logs.

Then a user may perform other types of operations like data profiling, modeling, etc. which will be discussed in section 3.2.3 "Centralized learning".

Subsections below will focus specifically on the implementations of Agents, Cen-

tralized and Federated Learning.

Agents

Agents can actively collect specific types of data and are highly customizable. They are designed in a way that any advanced user can write their own parser. They can be scheduled to parse and retrieve information from sources like wearable devices or files in formats such as CSV, XML (TBD), and JSON (TBD). The FHIR unification format is used, which supports the JSON data format preferable for HTTP interactions.

Since most (if not all) wearable device software is closed source (because they are proprietary) the data may be retrieved using Reverse Engineering (RE). But in this case, RE appeared to be very time-consuming, so it has been decided to use an open-source data exporter like [47] for MiFit. And there are several other solutions for other devices like Garmin.

After data has been retrieved, it needs to be mapped to a FHIR standard. FHIR utilizes resources that are the building blocks of this standard, representing various types of healthcare data in a structured format that can be easily shared and understood by different healthcare systems and applications. The resources we are interested in are Patient, Device, and Observation. According to [48], Patient resource enables us to offer demographic and administrative details about an individual receiving care or health services. This Resource provides information on patients and animals participating in various health-related activities, such as monitoring personal health and exercise data. Device resource monitors specific instances of a device and their location, making it perfect for wearable Personal Health gadgets. Other sources mention it to keep track of which device carried out an activity like a procedure or an observation [49]. The study relies on the observation resource type to store data measurements and basic statements regarding a patient, device, or another subject [50]. Some purposes for the Observation resource are:

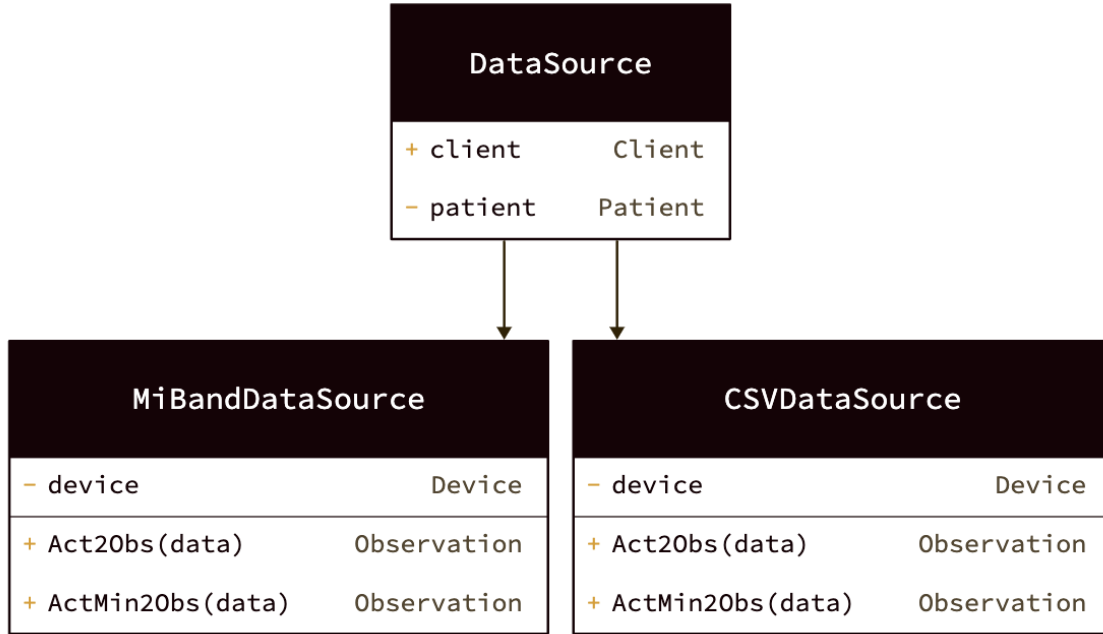
1. Important indicators like weight, blood pressure, and temperature.
2. Laboratory results such as blood sugar levels, or a calculated glomerular filtra-

tion rate (GFR).

3. Measurements taken from devices like EKG or Pulse Oximetry.

Many observations consist of basic name/value pairs along with metadata, while some observations involve grouping other observations logically or are comprised of multiple components.

Figure 3-6: DataSource class diagram with MiBand and CSV implementations.



To create a new data source, the developer needs to implement a new class of custom DataSource (DS). Figure 3-6 shows how DSs are implemented on MiBandDS and CSVDS.

Table 3.1: Observation resource structure

Name	Type	Description	Default value in DS
identifier status	Identifier code	Business Identifier for observation registered preliminary final amended +	sha256 of the resource preliminary
category device	CodeableConcept Reference(Device)	Classification of type of observation A reference to the device that gen- erates the measurements	PhysicalActivity according to [51] A reference to the device (e.g. MiBand)
effectiveDateTime	dateTime	Clinically relevant time/time- period for observation	timestamp of the metric
dataAbsentReason performer	CodeableConcept Reference(Patient)	Why the result is missing Who is responsible for the observa- tion	'unknown' A reference to the User
subject	Reference(Patient)	Who and/or what the observation is about	A reference to the User
component	BackboneElement	Component results	A component result value according to [51]

In order to map the data to the Observation resource, first, we need to investigate

how the Observation resource looks like (see Table 3.1) from [50]. Those fields are mandatory to fill in order to map the metrics to Observation resource.

Figure 3-7: Calories burned in the FHIR format.

```
1 {
2   "code": {
3     "coding": [
4       {
5         "system": "http://loinc.org",
6         "code": "41979-6",
7         "display": "Calories burned in 24 hour Calculated"
8       }
9     ]
10  },
11  "valueQuantity": {
12    "value": 42,
13    "unit": "kilocalories per day",
14    "system": "http://unitsofmeasure.org",
15    "code": "kcal/d"
16  }
17 }
```

Time to take a closer look at the component field. Figure 3-7 shows the example of how calories burned might be translated into FHIR standard. Here we can see several notable parts. Each component must have a coding value in LOINC format [52]. LOINC is a universal standard for identifying health measurements and observations, facilitating standardized coding and interoperability in electronic health records and health information systems. And to keep the unit of measures standardized (the USA uses imperic system, while Europe and the CIA use metric), all the values must use the system of measure from Unified Code for Units of Measure (UCUM) [53]. UCUM is a standardized system for representing units of measurement used in healthcare and other scientific fields, aiming to provide a common language for expressing quantities, enabling interoperability and consistency across different applications and systems.

For now, the agents can run scheduled using default cron that is available on any UNIX-like system. But the use of other job schedulers is discussed in section 5.1 "Future Works".

Centralized learning

As has been mentioned in section 3.2.2 "AutoML framework", PyCaret has been utilized for AutoML and centralized learning fully relies on PyCaret as well. It is low-code and, thus, does not require much of a tuning. Under the hood, the application runs the code similar to Figure 3-8.

Figure 3-8: The example of Classification model training in PyCaret.

```
1  # load sample dataset
2  from pycaret.datasets import get_data
3  data = get_data('diabetes')
4
5  from pycaret.classification import *
6  s = setup(data, target = 'Class variable', session_id = 123)
7
8  # functional API
9  best = compare_models()
```

Below is a description of how the app works in centralized learning. After uploading and preprocessing the dataset (see Figure 3-5), the user needs to go to the "Modelling" step and choose model configurations: target column - the column you want to predict, ignore columns - the columns that model training process should ignore, problem type - the type of supervised learning, and whether you want to use K Fold Cross-Validation technique to estimate the models (not recommended on large datasets and weak hardware as the default K=10. It will be too long). After that, the user needs to run the training. The whole process is shown in Figure 3-9.

Finally, the user needs to finalize the chosen model in "Finalize model" section (Figure 3-10). The process of finalization trains a given model on the entire dataset including the hold-out set. It does not change any parameter of the model.

Figure 3-9: Modelling section of the GUI with uploaded data from Apple Watch.

Model Building

Choose the Target Column: activity_trimmed

Choose the Ignore Columns: Unnamed: 0

Choose the Problem Type: classification

☐ Cross Validation

Run Modelling

	Description	Value
0	Session Id	4451
1	Target	activity_trimmed
2	Target type	MultiClass
3	Original data shape	(2924, 18)
4	Transformed data shape	(2924, 17)
5	Transformed train set shape	(2046, 17)
6	Transformed test set shape	(878, 17)
7	Ignore features	1
8	Numeric features	16
9	Preprocess	True

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
rf	Random Forest Classifier	0.82	0.9696	0.9696	0.82	0.8209	0.8204	0.7832	0.13
lightgbm	Light Gradient Boosting Machine	0.8166	0.9692	0.8166	0.8179	0.8169	0.7793	0.7795	2.26
et	Extra Trees Classifier	0.8123	0.9711	0.8121	0.8135	0.8126	0.7735	0.7736	0.1
catboost	CatBoost Classifier	0.8098	0.9685	0.8098	0.8116	0.8101	0.7711	0.7714	2.69
xgboost	Extreme Gradient Boosting	0.803	0.9708	0.803	0.8051	0.8033	0.7627	0.763	0.35
dt	Decision Tree Classifier	0.6834	0.8089	0.6834	0.6843	0.6832	0.6186	0.6188	0.02
gbc	Gradient Boosting Classifier	0.6788	0.9215	0.6788	0.6766	0.677	0.613	0.6133	2.9
knn	K Neighbors Classifier	0.5661	0.8345	0.5661	0.5584	0.5575	0.4764	0.478	0.01
ada	Ada Boost Classifier	0.4169	0.7406	0.4169	0.4116	0.4095	0.2996	0.301	0.13
qda	Quadratic Discriminant Analysis	0.4089	0.7466	0.4089	0.435	0.3972	0.2874	0.2855	0.01

Figure 3-10: Finalize model section of the GUI.

Finalize model

Choose the Model: Random Forest Classifier

Choose the Target Column: activity_trimmed

Choose the Ignore Columns: Unnamed: 0

Choose the Problem Type: classification

Finalize model

Model finalized successfully

Federated Learning

The main part of the project is Federated Learning. Flower main page [54] clearly explains how federated learning works:

- Step 0. Initialize the global model.

The initial step in federated learning is to initialize the global model on the server. This is similar to classic centralized learning, where the model parameters are initialized either randomly or from a previously saved checkpoint.

- Step 1. Send global model to client nodes (edge devices or servers).

Transmit the global model to various connected devices (client nodes). This ensures that each participating node starts with the same model parameters, enabling them to train their local models simultaneously. By only selecting a small number of client nodes, the training process becomes more efficient, as adding more nodes provides diminishing returns.

- Step 2. Local training on each client node's dataset.

Each selected client node performs local training on its own dataset, using the latest global model parameters. They train their local model for a brief period, typically one epoch or a few mini-batches.

- Step 3. Return model updates to the server.

After completing local training, each client node sends their updates to the central server. These updates could be the full model parameters or just the gradient accumulated during local training.

- Step 4. Aggregate model updates from client nodes to create a new global model.

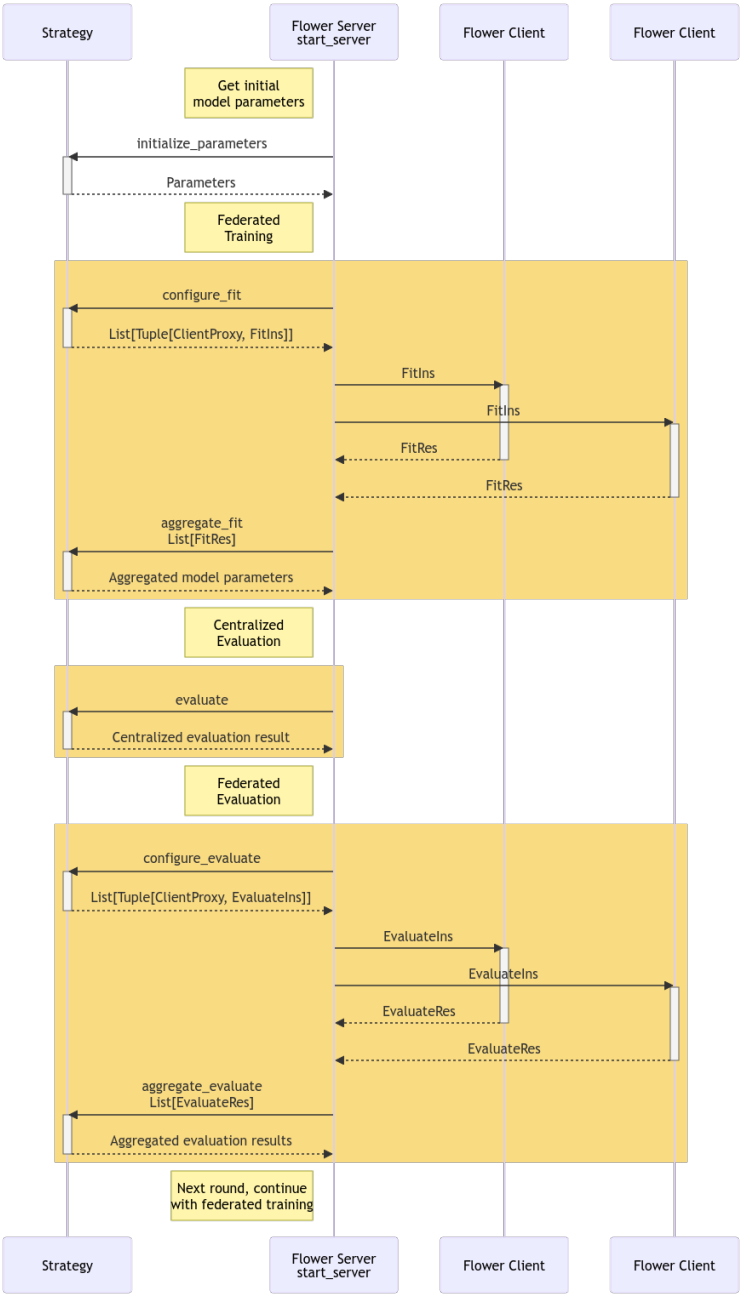
The central server aggregates the updates from all client nodes to create a new global model. This process is called federated averaging (FedAvg [55]), and it is an essential aspect of federated learning. The server takes the weighted average of the model updates, where the weight is the number of examples each client used for training. This ensures that each example has an equal influence on the resulting global model.

- Step 5. Repeat steps 1-4 until the global model converges.

Repeat steps 1-4 until the global model converges. During each iteration, the client nodes train their local models for a brief period, sending their updates to the server for aggregation. By repeating this process, the global model adapts to the collective knowledge of all client nodes, ultimately leading to improved performance on unseen data.

Figure 3-11 shows how the steps above work in terms of methods.

Figure 3-11: Sequence of methods call in Flower [54].



Strategies

The concept of strategy abstraction allows for the execution of completely personalized strategies. A tactic essentially refers to the federated learning algorithm that operates on the server. Strategies determine the client sampling method, client training configuration, update aggregation, and model evaluation. Flower includes several pre-installed strategies, but this study uses FedAvg specifically. Algorithm 3-12 explains how this strategy works.

Figure 3-12: The pseudocode for the FedAvg algorithm [55].

Algorithm 1 FederatedAveraging. The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

```
initialize  $w_0$ 
for each round  $t = 1, 2, \dots$  do
   $m \leftarrow \max(C \cdot K, 1)$ 
   $S_t \leftarrow$  (random set of  $m$  clients)
  for each client  $k \in S_t$  in parallel do
     $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$ 
   $m_t \leftarrow \sum_{k \in S_t} n_k$ 
   $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$  // Erratum4
```

```
ClientUpdate( $k, w$ ): // Run on client  $k$ 
   $\mathcal{B} \leftarrow$  (split  $\mathcal{P}_k$  into batches of size  $B$ )
  for each local epoch  $i$  from 1 to  $E$  do
    for batch  $b \in \mathcal{B}$  do
       $w \leftarrow w - \eta \nabla \ell(w; b)$ 
  return  $w$  to server
```

Since the default FedAvg implementation of Flower is not suited for AutoML, it has been modified to use custom serialization and deserialization methods which will be discussed in section 3.2.3 "Serialization and deserialization".

Unfortunately, FedAvg cannot satisfy all types of models. [56] states that there are two types of classifier models:

- Coefficient-based models. Certain machine learning models can be efficiently described using a collection of coefficients that can be incrementally learned.

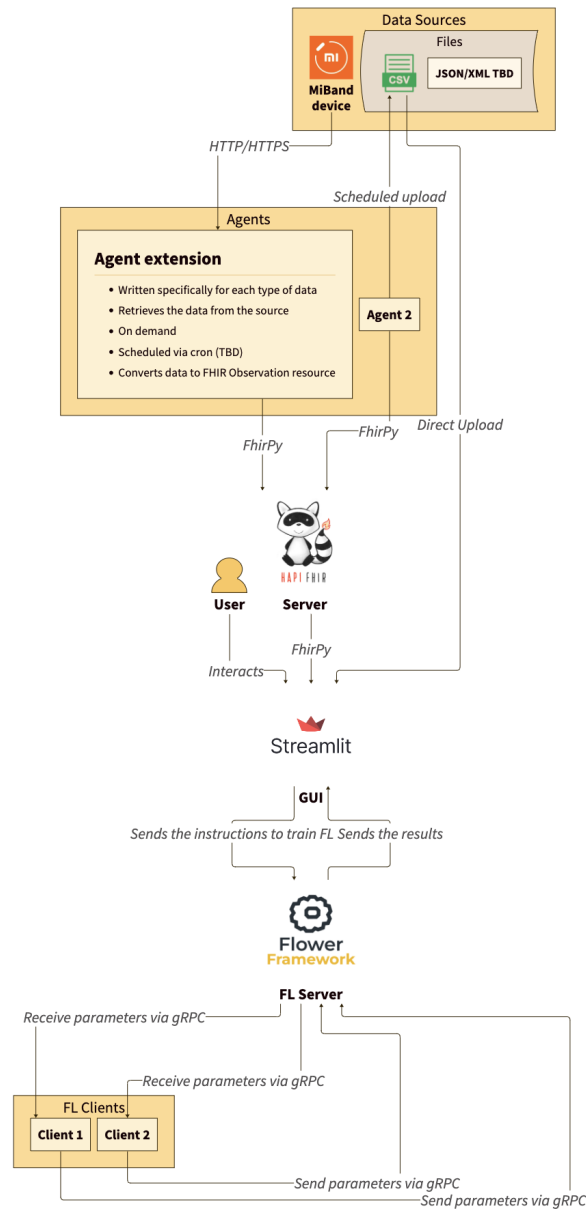
Example models include logistic regression and neural networks. Additionally, it works only with models whose attributes can both be read and directly written to, such that a new aggregated model can be computed by averaging the coefficients.

- Ensemble models. They usually merge a variety of basic models or estimators. Some sample models in this category are the random forest, AdaBoost, and XGBoost classifiers. For instance, a random forest classifier utilizes decision tree models as its base estimators. Thus, it is not easy to find an average of those estimators.

While *Coefficient-based models* may be utilized with FedAvg strategy, *Ensemble models* require something else. [55] proposes one simple yet not optimal method to train a random forest classifier in a federation which suggests that each client needs to build their own model locally, with the coordination server combining these models into a new random forest. In this manner, averaging model updates or enhancing the aggregated model in multiple rounds is unnecessary. Nevertheless, there could be more effective methods to enhance the precision of the combined model while sacrificing a less efficient federated training process, but this is not the intention of this study. However, it can be discussed in section 5.1 "Future Works". This study utilizes the same approach in the case of *Ensemble models* and implements another strategy called FedEnsemble. In essence, FedEnsemble trains individual models locally, assesses them on a server, identifies the model with the least loss, and then shares its weights throughout the system.

The final overview of the application is shown in Figure 3-13.

Figure 3-13: The final overview of the application.



Evaluation

As it is displayed in in Figure 3-11 there are two main approaches to evaluating models in federated learning systems:

- **Centralized (or server-side) evaluation.** Every preset strategy enables centralized evaluation through the inclusion of an evaluation function when initialized. An evaluation function is a function that can receive the current global model parameters and produce evaluation results.

- Federated (or client-side) evaluation. `EvaluateRes` provides client-side evaluation and can be adjusted from the server side. During the training process, it is possible to assess model parameters as well. `FitRes` is capable of providing any evaluation outcomes in the form of a dictionary.

The proposed solution uses them both. The evaluation of the model is handled by PyCaret itself and uses metrics defined in Table 3.2 for Classification and Table 3.3 for Regression.

Table 3.2: Evaluation Metrics for Classification

Metric	Description
Accuracy	Proportion of correctly classified instances
Precision	Proportion of true positive predictions among all positive predictions
Recall (Sensitivity)	Proportion of true positive predictions among all actual positive instances
F1 Score	Harmonic mean of precision and recall
AUC-ROC	Area under the Receiver Operating Characteristic curve
AUC-PR	Area under the Precision-Recall curve
Log Loss	Logarithmic loss between predicted probabilities and actual labels
Kappa	Cohen's Kappa coefficient
Matthews Correlation Coefficient (MCC)	Measures the quality of binary classifications

Table 3.3: Evaluation Metrics for Regression

Metric	Description
R-squared (R2)	Proportion of the variance in the dependent variable predictable from the independent variables
Mean Absolute Error (MAE)	Average of the absolute differences between predicted and actual values
Mean Squared Error (MSE)	Average of the squared differences between predicted and actual values
Root Mean Squared Error (RMSE)	Square root of the MSE
Mean Absolute Percentage Error (MAPE)	Average of the absolute percentage differences between predicted and actual values
Mean Percentage Error (MPE)	Average of the percentage differences between predicted and actual values

Serialization and deserialization

Unfortunately, sci-kit models returned by PyCaret do not provide any methods to get fitted parameters. Fortunately, fitted parameters are named with underscore prefixes and postfixes, e.g. `coef_`, `intercept_`, and `estimators_`. Since it may be tricky to extract some prefixes, most of the parameters are transmitted as it is — packed in models themselves.

Unfortunately, default methods supported by Flower do not handle datatypes other than `numpy` multidimensional arrays (`ndarrays`), thus it has been decided to use `pickle` to serialize any type of data. However, `pickle` is not secure, it is possible to construct malicious pickle data which will execute arbitrary code during unpickling. Thus there is a way to improve this process in section 5.1 "Future Works".

The final application is shown in Figure 3-14.

Figure 3-14: Federated learning section of the GUI.

Federated Learning Test

Choose the Target Column: activity_30mmed

Problem Type: classification

Number of Clients: 3

Number of Rounds: 10

Choose the Model: Random Forest Classifier

Run Test

Running / jupyter-R/venv at 32.3 # classification_activity_30mmed_2u3d28v

```

Starting server
[ Processing: 0% ] [ 0/4 (0.00+0%, 70%) Processing: 75% ] [ 3/4 (0.00+00.00, 24.50%) ]
INFO flux 2024-05-26 21:14:18.083 | app.py:363 | Starting Flower server, config: ServerConfig(num_rounds=10, round_timeout=None)

Starting client 0
INFO flux 2024-05-26 21:14:18.120 | app.py:176 | Flower ECDHE server running (10 rounds), SSL is disabled

Starting client 1
INFO flux 2024-05-26 21:14:18.121 | server.py:80 | Initializing global parameters

Description Value
INFO flux 2024-05-26 21:14:18.121 | server.py:271 | Using initial parameters provided by strategy
0 Session of 42
INFO flux 2024-05-26 21:14:18.121 | server.py:311 | Evaluating initial parameters
1 Target activity_30mmed

Training is done!
  
```

Results

Model	Accuracy	AUC	Recall	Prec.	F1	Fbeta	BCE	Log Loss
Federated RF	0.9714	0.9893	0.9714	0.9796	0.9733	0.9733	0.4021	0.4021

3.3 Dataset

In this section, datasets used for testing and evaluation of the solution are discussed. Since the solution focuses only on two types of machine learning problems: classification and regression, we are only going to focus on the datasets that would be suitable for them and retrieved by wearable devices.

3.3.1 Classification

Unfortunately, as there is insufficient data available from one wearable user, it has been deemed necessary to incorporate datasets from additional sources.

Harvard’s Apple Watch physical activity dataset

The dataset from Harvard University’s research in [57] has been utilized for solving classification problems. The research aimed to investigate if commercial wearable devices could effectively forecast sitting, lying, and varying levels of physical activity. Scientists enlisted 46 volunteers, including 26 women, in a convenience sample to utilize three gadgets - a GENEActiv, an Apple Watch Series 2, and a Fitbit Charge HR2. Attendees finished a 65-minute procedure that included 40 minutes of treadmill

activity and 25 minutes of sitting or lying down. Energy expenditure was measured using indirect calorimetry. The variable measured in the study was the activity class, including sitting, lying, self-paced walking, and METS levels 3, 5, and 7. Four distinct machine learning models incorporated data on steps, distance, heart rate, and calories from Apple Watch and Fitbit on a minute-by-minute basis. There are 3656 minutes of Apple Watch data and 2608 minutes of Fitbit data in the dataset. Scientists examined decision trees, random forest, rotation forest models, and support vector machines. Rotation forest models achieved the top classification accuracies of 82.6% for Apple Watch and 89.3% for Fitbit. Classification accuracies for Apple Watch data varied from 72.5% for sitting to 89.0% for 7 METS. Fitbit’s accuracies ranged from 86.2% for sitting to 92.6% for 7 METS. The research showed that Apple Watch and Fitbit, popular wearable devices, could accurately predict the type of physical activity. The findings endorse the utilization of real-time data from Apple Watch and Fitbit with machine learning methods for large-scale classification of physical activity types among the general population.

HARTH dataset

The HARTH dataset [58] comprises 3-axial accelerometer data from 22 participants, with sensors placed on the thigh and lower back, recording acceleration in three dimensions at a high sampling rate for detailed motion tracking. The time-series data is accompanied by annotations of various activities and includes raw signals that allow for custom processing and feature extraction. Additionally, the dataset provides metadata on participant demographics and experiment protocols, supporting comprehensive analysis for human activity recognition research.

3.3.2 Regression

When it comes to regression, things are getting a bit easier. Regression models do not require polished target data (at some level), thus, the raw data from the device itself can be used without any heavy data processing. The solution is tested using the

activity data from my own MiBand 3, 4, and 7 devices that I wore for approximately 3 years. The models will try to predict the amount of calories burned during activities which include: steps, distance, and run distance.

DAT263x

To prove the robustness of the results, it has been decided to also include a side dataset with a similar nature. DAT263x dataset has been mentioned in EDX "Microsoft DAT263x Introduction to Artificial Intelligence (AI)" and is meant to be used with Azure, but instead found on Kaggle [59]. It contains gender, age, weight, height, duration, heart rate, body temperature, and calories burned of each participant during physical activity.

Chapter 4

Results

In this chapter, the evaluation of the proposed solution is discussed. As has been mentioned in section 3.3 "Dataset", several datasets are used for evaluation. The evaluation of the solution has been executed on MacBook Pro 14 M1 (see Table 4.1 for specifications). Not all models can be trained in a Federated manner since it depends on the support of Incremental (or Online) learning, such that the models are continually updated and trained as new data becomes available, without the need to retrain from scratch. PyCaret provides 5 such models: Extra Trees, LightGBM, Extreme Gradient Boosting, Random Forest, and CatBoost. Thus, the results are obtained using those models. Also, to shorten the tables that show results, codenames are used for the models instead of the full names. For instance, `et` refers to Extra Trees, `xgboost` to extreme gradient boosting, `rf` to random forest, `lightgbm` to LightGBM, and `catboost` to catboost.

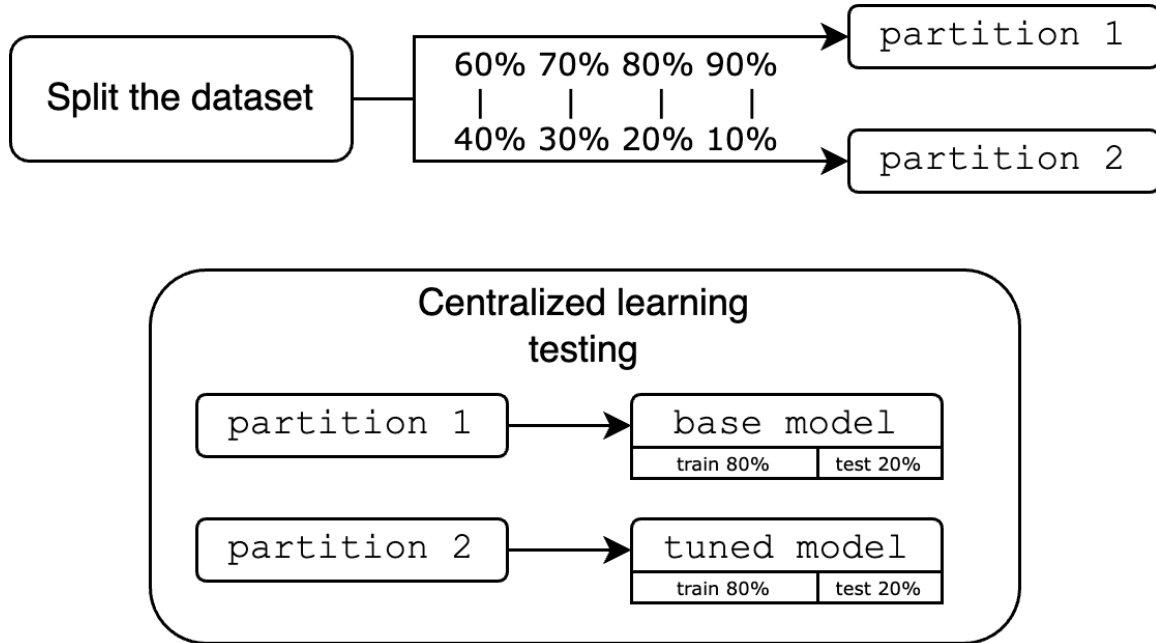
Table 4.1: MacBook Pro 14 Characteristics

Feature	Specification
Processor	Apple M1 Pro 8-core CPU with 6 performance cores and 2 efficiency cores
Memory	16GB unified memory
Graphics	14-core GPU
Operating System	macOS Sonoma 14.3

4.1 Data partitioning

Since experiments are simulated locally using only one dataset, it is crucial to clarify the approach for partitioning the data. Assessing the solution requires utilizing various partition proportions and considering different numbers of clients. To ensure a fair comparison between centralized and FL models, the data will be segmented in the following ways: into pairs of 60%/40%, 70%/30%, 80%/20%, and 90%/10%, creating two separate partitions. The first partition will undergo training using traditional centralized techniques. The resultant model, termed **base model** will then be tuned and tested with the second partition for the CL assessment (see Figure 4-1 for details).

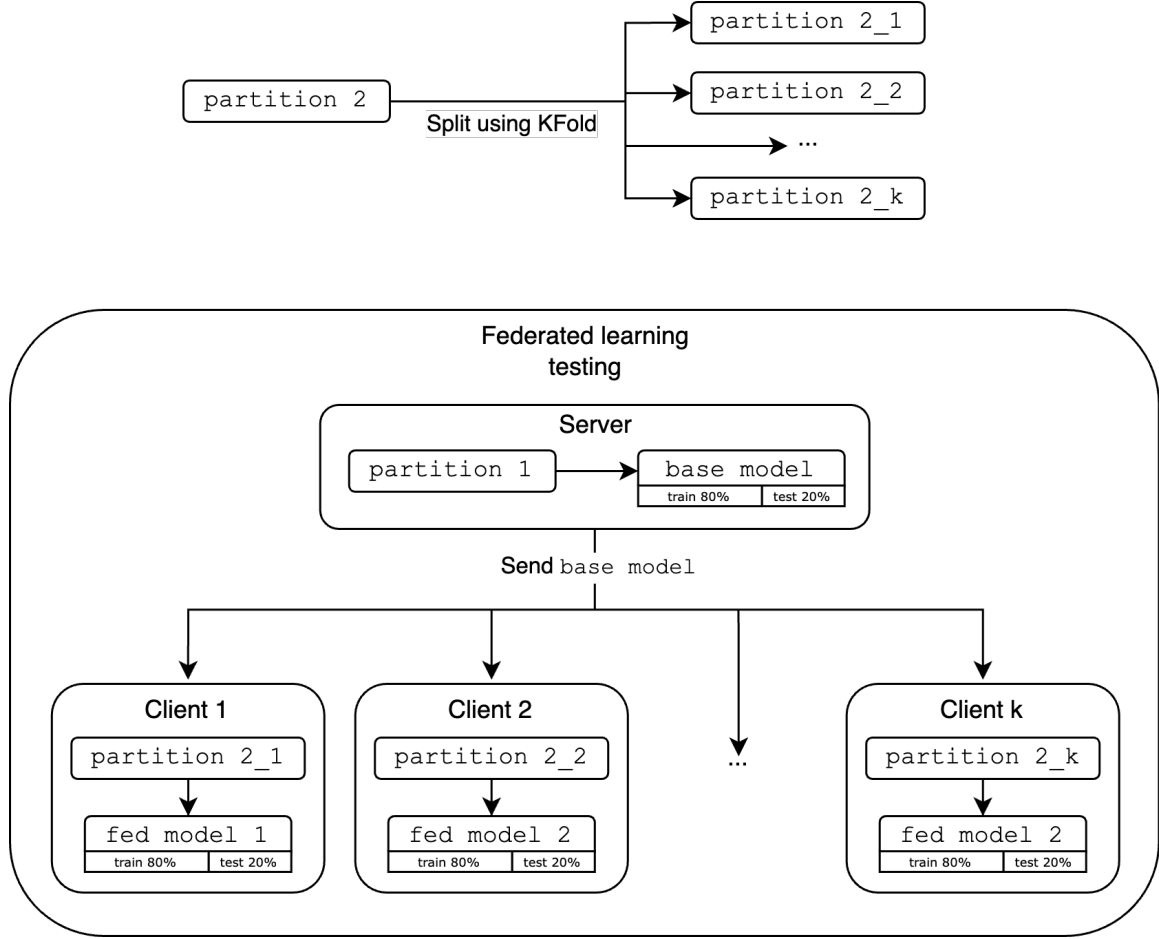
Figure 4-1: Evaluation of Centralized Learning



In the FL evaluation (see Figure 4-2), **base model** will act as the initial global model and will be distributed to k clients (2, 4, 8). The second partition will then be divided using KFold (StratifiedKFold is excluded as it doesn't represent realistic scenarios where data is non-iid and might not reflect the full dataset) and allocated across these k clients. Each client will refine the global model with their respective new fold of data. These results will be aggregated and compared with the performance

of the tuned centralized models.

Figure 4-2: Evaluation of Federated Learning



4.2 Time Metrics Evaluation

Evaluating performance in terms of time efficiency is crucial, particularly for distributed systems like FL which operates in parallel across multiple clients. Although FL has the theoretical advantage of parallel processing that could lead to time savings compared to CL, it introduces additional latency due to data transmission and the overhead of coordination between server and client operations. Considering these operational time costs for an accurate assessment relative to CL is important. The below metrics enumerate the specific time components measured:

- For CL:

- **Initial Model Training Time (T1):** Training time of base model;
 - **Refined Model Training Time (T2):** Training time of tuned model;
 - **Total CL Time:** The sum of T1 and T2;
- For Federated Learning (FL):
 - **Server Initialization Time:** Time required for server setup before model distribution ($\approx 3s$);
 - **Initial Model Training Time (T1):** Training time of base model;
 - **Model Distribution Latency:** Time taken to transmit the initial model from the server to each client. It is almost instant because the simulation is happening locally;
 - **Local Model Training Time (T2):** Time each client spends to train their allocated portion of the federated model. Only the maximum will be considered;
 - **Total FL Time:** Since **Server Initialization Time** and **Model Distribution Latency** are not constant and depend on the current CPU usage, the total is considered to be the sum of T1 and T2, as these are the metrics that will be compared against CL;

4.3 Analysis of Real Data

The raw data is examined in its original form, without any grouping or the addition of synthetic data. Attention is directed towards identifying patterns, variations, and anomalies within the data. This process is instrumental in establishing a baseline understanding of the data's initial state, enabling the evaluation of the impact that subsequent non-grouped partitioning may have. Transparency is upheld throughout, ensuring that a true representation of the data is presented before any modifications. The result tables have been condensed to avoid an unnecessary increase in page count,

so you can find the results on https://docs.google.com/spreadsheets/d/1wI9_TZws_4rPCv0DRu4oQCo5GNDPkWq9pQgaY51d388/edit?usp=sharing

4.3.1 Classification

Harvard’s Apple Watch physical activity dataset

First, the dataset used for Classification needs to be discussed. Table 4.2 provides more detailed information about the characteristics of the Apple Watch dataset, including the number of variables, observations, and types of variables. The number of records is *3656*, which then is split into **partition 1** and **partition 2** with the ratio of 60%/40%, 70%/30%, 80%/20%, and 90%/10%.

Table 4.2: Apple Watch Dataset Statistics

Statistic	Value
Number of variables	18
Number of observations	3656
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	514.2 KiB
Average record size in memory	144.0 B
Variable types	
Numeric	16
Categorical	2

60%/40% partition, $k = 2$ As can be seen from Table 4.3 Random Forest and Extra Trees trained in a federated manner outperform the centralized tuned model by $\approx 3\%$. Also, there is a slight timing advantage in Random Forest.

Table 4.3: AW Dataset 60%/40% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7540	0.9470	0.7540	0.7567	0.7542	0.7040	0.7044	0.7188	3.2386
(xgboost) tuned model	0.7850	0.9528	0.7850	0.7827	0.7835	0.7409	0.7411	0.6937	0.6256
(xgboost) Aggregated Mean	0.7415	0.9417	0.7415	0.7541	0.7437	0.6873	0.6889	0.8337	0.4356
(xgboost) Aggregated Max	0.7551	0.9492	0.7551	0.7632	0.7572	0.7042	0.7049	0.9210	0.5071
(lightgbm) base model	0.7472	0.9471	0.7472	0.7478	0.7472	0.6954	0.6955	0.7483	20.6402
(lightgbm) tuned model	0.7611	0.9493	0.7611	0.7627	0.7617	0.7120	0.7120	0.9073	1.8645
(lightgbm) Aggregated Mean	0.7075	0.9316	0.7075	0.7203	0.7084	0.6458	0.6478	1.1644	1.6203
(lightgbm) Aggregated Max	0.7143	0.9327	0.7143	0.7292	0.7138	0.6536	0.6568	1.1770	1.8747
(catboost) base model	0.7677	0.9497	0.7677	0.7679	0.7673	0.7204	0.7206	0.6732	20.0065
(catboost) tuned model	0.7713	0.9555	0.7713	0.7698	0.7695	0.7242	0.7246	0.6239	2.5457
(catboost) Aggregated Mean	0.7551	0.9506	0.7551	0.7617	0.7553	0.7038	0.7051	0.6747	2.3885
(catboost) Aggregated Max	0.7687	0.9535	0.7687	0.7705	0.7667	0.7206	0.7217	0.6962	2.6137
(rf) base model	0.7813	0.9536	0.7813	0.7817	0.7807	0.7365	0.7368	0.7207	1.0896
(rf) tuned model	0.7474	0.9518	0.7474	0.7501	0.7481	0.6957	0.6959	0.7358	0.0017
(rf) Aggregated Mean	0.7789	0.9526	0.7789	0.7835	0.7769	0.7327	0.7341	0.7305	0.0007
(rf) Aggregated Max	0.7891	0.9544	0.7891	0.7914	0.7870	0.7454	0.7466	0.7396	0.0008
(et) base model	0.7836	0.9570	0.7836	0.7843	0.7835	0.7392	0.7394	0.6651	0.7465
(et) tuned model	0.7406	0.9578	0.7406	0.7443	0.7412	0.6879	0.6885	0.6549	0.0008
(et) Aggregated Mean	0.7789	0.9615	0.7789	0.7824	0.7788	0.7331	0.7337	0.6584	0.0008
(et) Aggregated Max	0.7823	0.9652	0.7823	0.7861	0.7823	0.7373	0.7380	0.6729	0.0008

Table 4.4: AW Dataset 60%/40% partition, $k = 4$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7540	0.9470	0.7540	0.7567	0.7542	0.7040	0.7044	0.7188	2.0621
(xgboost) tuned model	0.7850	0.9528	0.7850	0.7827	0.7835	0.7409	0.7411	0.6937	0.4267
(xgboost) Aggregated Mean	0.7863	0.9515	0.7863	0.7906	0.7862	0.7422	0.7431	0.6995	0.2807
(xgboost) Aggregated Max	0.8108	0.9597	0.8108	0.8203	0.8123	0.7704	0.7716	0.7638	0.3418
(lightgbm) base model	0.7472	0.9471	0.7472	0.7478	0.7472	0.6954	0.6955	0.7483	20.3355
(lightgbm) tuned model	0.7611	0.9493	0.7611	0.7627	0.7617	0.7120	0.7120	0.9073	1.6461
(lightgbm) Aggregated Mean	0.7763	0.9444	0.7763	0.7877	0.7764	0.7300	0.7321	0.9051	0.7361
(lightgbm) Aggregated Max	0.8378	0.9621	0.8378	0.8417	0.8380	0.8033	0.8038	1.1338	0.8404
(catboost) base model	0.7677	0.9497	0.7677	0.7679	0.7673	0.7204	0.7206	0.6732	19.8964
(catboost) tuned model	0.7713	0.9555	0.7713	0.7698	0.7695	0.7242	0.7246	0.6239	3.0478
(catboost) Aggregated Mean	0.7933	0.9597	0.7933	0.8049	0.7933	0.7500	0.7524	0.5902	2.2282
(catboost) Aggregated Max	0.8649	0.9764	0.8649	0.8749	0.8679	0.8356	0.8367	0.7256	2.4382
(rf) base model	0.7813	0.9536	0.7813	0.7817	0.7807	0.7365	0.7368	0.7207	1.0893
(rf) tuned model	0.7474	0.9518	0.7474	0.7501	0.7481	0.6957	0.6959	0.7358	0.0016
(rf) Aggregated Mean	0.8034	0.9571	0.8034	0.8122	0.8029	0.7626	0.7644	0.7034	0.0013
(rf) Aggregated Max	0.8514	0.9695	0.8514	0.8589	0.8514	0.8201	0.8216	0.7687	0.0027
(et) base model	0.7836	0.9570	0.7836	0.7843	0.7835	0.7392	0.7394	0.6651	0.8995
(et) tuned model	0.7406	0.9578	0.7406	0.7443	0.7412	0.6879	0.6885	0.6549	0.0009
(et) Aggregated Mean	0.8035	0.9633	0.8035	0.8110	0.8033	0.7632	0.7648	0.6317	0.0007
(et) Aggregated Max	0.8219	0.9759	0.8219	0.8337	0.8222	0.7860	0.7879	0.7382	0.0008

60%/40% partition, $k = 4$ The information presented in Table 4.4 shows that every model trained in FL either works well with CL or performs better than CL.

Table 4.5: AW Dataset 60%/40% partition, $k = 8$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7540	0.9470	0.7540	0.7567	0.7542	0.7040	0.7044	0.7188	5.2919
(xgboost) tuned model	0.7850	0.9528	0.7850	0.7827	0.7835	0.7409	0.7411	0.6937	0.5018
(xgboost) Aggregated Mean	0.7703	0.9500	0.7703	0.7877	0.7643	0.7213	0.7260	0.7650	0.3213
(xgboost) Aggregated Max	0.8649	0.9812	0.8649	0.8740	0.8580	0.8361	0.8399	1.4643	0.8470
(lightgbm) base model	0.7472	0.9471	0.7472	0.7478	0.7472	0.6954	0.6955	0.7483	18.2567
(lightgbm) tuned model	0.7611	0.9493	0.7611	0.7627	0.7617	0.7120	0.7120	0.9073	1.9065
(lightgbm) Aggregated Mean	0.7297	0.9397	0.7297	0.7539	0.7258	0.6722	0.6780	0.9746	0.3295
(lightgbm) Aggregated Max	0.8649	0.9766	0.8649	0.8721	0.8584	0.8367	0.8397	1.3976	0.4120
(catboost) base model	0.7677	0.9497	0.7677	0.7679	0.7673	0.7204	0.7206	0.6732	19.2406
(catboost) tuned model	0.7713	0.9555	0.7713	0.7698	0.7695	0.7242	0.7246	0.6239	2.2813
(catboost) Aggregated Mean	0.7568	0.9569	0.7568	0.7734	0.7527	0.7051	0.7088	0.6468	1.3629
(catboost) Aggregated Max	0.8649	0.9939	0.8649	0.8687	0.8632	0.8369	0.8383	1.0747	1.4847
(rf) base model	0.7813	0.9536	0.7813	0.7817	0.7807	0.7365	0.7368	0.7207	1.0347
(rf) tuned model	0.7474	0.9518	0.7474	0.7501	0.7481	0.6957	0.6959	0.7358	0.0026
(rf) Aggregated Mean	0.8040	0.9619	0.8040	0.8275	0.8062	0.7634	0.7670	0.7854	0.0007
(rf) Aggregated Max	0.9459	0.9854	0.9459	0.9535	0.9454	0.9347	0.9363	1.5949	0.0007
(et) base model	0.7836	0.9570	0.7836	0.7843	0.7835	0.7392	0.7394	0.6651	0.7119
(et) tuned model	0.7406	0.9578	0.7406	0.7443	0.7412	0.6879	0.6885	0.6549	0.0008
(et) Aggregated Mean	0.8209	0.9714	0.8209	0.8440	0.8211	0.7839	0.7887	0.5940	0.0008
(et) Aggregated Max	0.8919	0.9930	0.8919	0.8996	0.8910	0.8693	0.8708	0.7576	0.0009

60%/40% partition, $k = 8$ Table 4.5 shows that Random Forest FL surpasses CL by $\approx 6\%$; Extra Trees FL surpasses CL by $\approx 8\%$.

Table 4.6: AW Dataset 70%/30% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8457	0.9776	0.8457	0.8489	0.8464	0.8143	0.8147	0.4307	4.4140
(xgboost) tuned model	0.7818	0.9636	0.7818	0.7861	0.7819	0.7372	0.7381	0.6042	0.3680
(xgboost) Aggregated Mean	0.7864	0.9586	0.7864	0.8032	0.7886	0.7420	0.7444	0.6609	0.2602
(xgboost) Aggregated Max	0.8000	0.9617	0.8000	0.8048	0.7977	0.7593	0.7613	0.7079	0.2888
(lightgbm) base model	0.8340	0.9741	0.8340	0.8383	0.8346	0.8002	0.8007	0.4734	20.1328
(lightgbm) tuned model	0.7591	0.9612	0.7591	0.7569	0.7572	0.7089	0.7092	0.7945	1.7213
(lightgbm) Aggregated Mean	0.7455	0.9424	0.7455	0.7638	0.7481	0.6922	0.6948	1.0604	0.9673
(lightgbm) Aggregated Max	0.7455	0.9535	0.7455	0.7671	0.7490	0.6939	0.6960	1.1986	0.9940
(catboost) base model	0.8457	0.9771	0.8457	0.8487	0.8458	0.8145	0.8150	0.4791	20.1468
(catboost) tuned model	0.8091	0.9700	0.8091	0.8097	0.8085	0.7699	0.7704	0.5055	2.5519
(catboost) Aggregated Mean	0.7818	0.9610	0.7818	0.7860	0.7804	0.7366	0.7380	0.6003	2.4578
(catboost) Aggregated Max	0.8091	0.9643	0.8091	0.8148	0.8059	0.7705	0.7729	0.6155	2.7077
(rf) base model	0.8281	0.9749	0.8281	0.8301	0.8286	0.7931	0.7933	0.5967	1.3484
(rf) tuned model	0.8091	0.9653	0.8091	0.8158	0.8094	0.7702	0.7714	0.6547	0.0017
(rf) Aggregated Mean	0.8091	0.9745	0.8091	0.8171	0.8085	0.7698	0.7717	0.6120	0.0008
(rf) Aggregated Max	0.8273	0.9770	0.8273	0.8348	0.8238	0.7921	0.7953	0.6162	0.0010
(et) base model	0.8457	0.9791	0.8457	0.8458	0.8455	0.8143	0.8144	0.5200	0.7931
(et) tuned model	0.8136	0.9680	0.8136	0.8168	0.8134	0.7755	0.7762	0.7264	0.0007
(et) Aggregated Mean	0.8000	0.9767	0.8000	0.8143	0.7950	0.7589	0.7638	0.5481	0.0010
(et) Aggregated Max	0.8091	0.9793	0.8091	0.8227	0.7963	0.7715	0.7793	0.5561	0.0011

70%/30% partition, k = 2 The results of Extreme Gradient Boosting and Random Forest when trained in FL and CL manner are comparable, but FL performs faster, as shown in Table 4.6.

70%/30% partition, k = 4 According to the data in Table 4.7, Random Forest and Extreme Trees yield similar results in both FL and CL training methods, with FL being quicker in terms of time. Extreme Gradient Boosting, when trained in FL, slightly surpasses the CL version by $\approx 1\%$.

Table 4.7: AW Dataset 70%/30% partition, k = 4

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8457	0.9776	0.8457	0.8489	0.8464	0.8143	0.8147	0.4307	6.4030
(xgboost) tuned model	0.7818	0.9636	0.7818	0.7861	0.7819	0.7372	0.7381	0.6042	0.8172
(xgboost) Aggregated Mean	0.7955	0.9662	0.7955	0.8117	0.7930	0.7515	0.7552	0.6354	0.3391
(xgboost) Aggregated Max	0.8182	0.9751	0.8182	0.8506	0.8164	0.7805	0.7872	0.8015	0.3548
(lightgbm) base model	0.8340	0.9741	0.8340	0.8383	0.8346	0.8002	0.8007	0.4734	22.1026
(lightgbm) tuned model	0.7591	0.9612	0.7591	0.7569	0.7572	0.7089	0.7092	0.7945	1.7096
(lightgbm) Aggregated Mean	0.7000	0.9328	0.7000	0.7140	0.6921	0.6366	0.6414	1.0596	0.5219
(lightgbm) Aggregated Max	0.7273	0.9628	0.7273	0.7522	0.7219	0.6724	0.6756	1.3973	0.6422
(catboost) base model	0.8457	0.9771	0.8457	0.8487	0.8458	0.8145	0.8150	0.4791	20.0104
(catboost) tuned model	0.8091	0.9700	0.8091	0.8097	0.8085	0.7699	0.7704	0.5055	2.5318
(catboost) Aggregated Mean	0.7818	0.9672	0.7818	0.7992	0.7768	0.7349	0.7391	0.5714	1.8999
(catboost) Aggregated Max	0.8182	0.9760	0.8182	0.8318	0.8192	0.7779	0.7798	0.6235	2.1231
(rf) base model	0.8281	0.9749	0.8281	0.8301	0.8286	0.7931	0.7933	0.5967	1.1212
(rf) tuned model	0.8091	0.9653	0.8091	0.8158	0.8094	0.7702	0.7714	0.6547	0.0014
(rf) Aggregated Mean	0.8045	0.9736	0.8045	0.8229	0.8049	0.7632	0.7659	0.6250	0.0007
(rf) Aggregated Max	0.8727	0.9815	0.8727	0.8835	0.8739	0.8454	0.8468	0.7114	0.0008
(et) base model	0.8457	0.9791	0.8457	0.8458	0.8455	0.8143	0.8144	0.5200	0.8646
(et) tuned model	0.8136	0.9680	0.8136	0.8168	0.8134	0.7755	0.7762	0.7264	0.0007
(et) Aggregated Mean	0.8136	0.9763	0.8136	0.8317	0.8134	0.7746	0.7779	0.5479	0.0007
(et) Aggregated Max	0.8727	0.9805	0.8727	0.8989	0.8762	0.8462	0.8507	0.5762	0.0008

70%/30% partition, k = 8 The outcomes of Extreme Gradient Boosting FL are superior to CL by $\approx 3\%$; Random Forest FL surpasses CL by $\approx 3\%$; Extra Trees FL outperforms CL by $\approx 1\%$, as shown in Table 4.8.

Table 4.8: AW Dataset 70%/30% partition, $k = 8$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8457	0.9776	0.8457	0.8489	0.8464	0.8143	0.8147	0.4307	5.5766
(xgboost) tuned model	0.7818	0.9636	0.7818	0.7861	0.7819	0.7372	0.7381	0.6042	0.4731
(xgboost) Aggregated Mean	0.8125	0.9658	0.8125	0.8482	0.8081	0.7705	0.7779	0.6111	0.2156
(xgboost) Aggregated Max	0.8929	0.9905	0.8929	0.9256	0.8920	0.8714	0.8781	0.7973	0.3084
(lightgbm) base model	0.8340	0.9741	0.8340	0.8383	0.8346	0.8002	0.8007	0.4734	23.2130
(lightgbm) tuned model	0.7591	0.9612	0.7591	0.7569	0.7572	0.7089	0.7092	0.7945	1.5958
(lightgbm) Aggregated Mean	0.7411	0.9329	0.7411	0.7762	0.7348	0.6829	0.6899	1.0995	0.2694
(lightgbm) Aggregated Max	0.8214	0.9680	0.8214	0.8798	0.8123	0.7869	0.8003	1.3985	0.3986
(catboost) base model	0.8457	0.9771	0.8457	0.8487	0.8458	0.8145	0.8150	0.4791	19.8735
(catboost) tuned model	0.8091	0.9700	0.8091	0.8097	0.8085	0.7699	0.7704	0.5055	2.6815
(catboost) Aggregated Mean	0.7857	0.9630	0.7857	0.8235	0.7759	0.7379	0.7474	0.6090	1.4277
(catboost) Aggregated Max	0.8214	0.9861	0.8214	0.8745	0.8231	0.7853	0.7975	0.8093	2.0512
(rf) base model	0.8281	0.9749	0.8281	0.8301	0.8286	0.7931	0.7933	0.5967	1.1687
(rf) tuned model	0.8091	0.9653	0.8091	0.8158	0.8094	0.7702	0.7714	0.6547	0.0024
(rf) Aggregated Mean	0.8348	0.9801	0.8348	0.8637	0.8300	0.7987	0.8051	0.5881	0.0008
(rf) Aggregated Max	0.9286	0.9900	0.9286	0.9286	0.9286	0.9105	0.9105	0.7683	0.0011
(et) base model	0.8457	0.9791	0.8457	0.8458	0.8455	0.8143	0.8144	0.5200	0.8346
(et) tuned model	0.8136	0.9680	0.8136	0.8168	0.8134	0.7755	0.7762	0.7264	0.0010
(et) Aggregated Mean	0.8259	0.9804	0.8259	0.8543	0.8245	0.7874	0.7938	0.5363	0.0009
(et) Aggregated Max	0.8929	0.9985	0.8929	0.9330	0.8913	0.8708	0.8803	0.7241	0.0019

80%/20% partition, $k = 2$ The results from Table 4.9 show that all models trained in FL approach perform better than their CL counterparts, except for Extra Trees.

Table 4.9: AW Dataset 80%/20% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7983	0.9631	0.7983	0.7980	0.7980	0.7572	0.7573	0.5665	6.4171
(xgboost) tuned model	0.7823	0.9513	0.7823	0.7868	0.7829	0.7371	0.7377	0.6984	0.3811
(xgboost) Aggregated Mean	0.7906	0.9484	0.7906	0.7876	0.7801	0.7472	0.7506	0.7745	0.2795
(xgboost) Aggregated Max	0.8514	0.9690	0.8514	0.8535	0.8509	0.8190	0.8195	1.0068	0.2994
(lightgbm) base model	0.7897	0.9621	0.7897	0.7884	0.7888	0.7469	0.7470	0.5832	20.0211
(lightgbm) tuned model	0.7007	0.9311	0.7007	0.7088	0.7029	0.6376	0.6383	1.0995	1.2489
(lightgbm) Aggregated Mean	0.7095	0.9345	0.7095	0.7073	0.6968	0.6493	0.6537	1.1297	0.7048
(lightgbm) Aggregated Max	0.7973	0.9609	0.7973	0.7976	0.7932	0.7530	0.7544	1.5189	0.7302
(catboost) base model	0.8103	0.9663	0.8103	0.8096	0.8088	0.7715	0.7719	0.5478	21.3602
(catboost) tuned model	0.7279	0.9479	0.7279	0.7287	0.7277	0.6714	0.6715	0.6706	2.4282
(catboost) Aggregated Mean	0.7635	0.9616	0.7635	0.7540	0.7457	0.7142	0.7195	0.6387	2.2842
(catboost) Aggregated Max	0.8378	0.9758	0.8378	0.8331	0.8313	0.8018	0.8036	0.7831	2.3572
(rf) base model	0.8068	0.9644	0.8068	0.8054	0.8056	0.7675	0.7677	0.6266	1.0893
(rf) tuned model	0.7959	0.9740	0.7959	0.7963	0.7951	0.7532	0.7535	0.5831	0.0015
(rf) Aggregated Mean	0.8176	0.9747	0.8176	0.8320	0.8156	0.7789	0.7825	0.5568	0.0007
(rf) Aggregated Max	0.8784	0.9899	0.8784	0.8936	0.8796	0.8525	0.8557	0.6676	0.0008
(et) base model	0.8376	0.9694	0.8376	0.8366	0.8352	0.8046	0.8053	0.6170	1.0265
(et) tuned model	0.8503	0.9763	0.8503	0.8571	0.8518	0.8189	0.8196	0.5174	0.0007
(et) Aggregated Mean	0.8244	0.9826	0.8244	0.8202	0.8134	0.7878	0.7910	0.4641	0.0007
(et) Aggregated Max	0.8919	0.9950	0.8919	0.8952	0.8919	0.8687	0.8693	0.5719	0.0007

80%/20% partition, k = 4 The data in Table 4.10 demonstrates that CatBoost FL outperforms CL by $\approx 4\%$; Random Forest FL outperforms CL by $\approx 2\%$; Extra Trees FL shows similar performance to CL.

Table 4.10: AW Dataset 80%/20% partition, k = 4

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7983	0.9631	0.7983	0.7980	0.7980	0.7572	0.7573	0.5665	2.2916
(xgboost) tuned model	0.7823	0.9513	0.7823	0.7868	0.7829	0.7371	0.7377	0.6984	0.6787
(xgboost) Aggregated Mean	0.7770	0.9630	0.7770	0.7983	0.7734	0.7293	0.7346	0.6345	0.2904
(xgboost) Aggregated Max	0.8378	0.9805	0.8378	0.8420	0.8306	0.8035	0.8064	0.9364	0.4504
(lightgbm) base model	0.7897	0.9621	0.7897	0.7884	0.7888	0.7469	0.7470	0.5832	22.5458
(lightgbm) tuned model	0.7007	0.9311	0.7007	0.7088	0.7029	0.6376	0.6383	1.0995	1.4725
(lightgbm) Aggregated Mean	0.6892	0.9147	0.6892	0.7207	0.6909	0.6215	0.6267	1.2652	0.3440
(lightgbm) Aggregated Max	0.7838	0.9532	0.7838	0.8047	0.7886	0.7383	0.7403	1.8115	0.3993
(catboost) base model	0.8103	0.9663	0.8103	0.8096	0.8088	0.7715	0.7719	0.5478	21.7641
(catboost) tuned model	0.7279	0.9479	0.7279	0.7287	0.7277	0.6714	0.6715	0.6706	2.3787
(catboost) Aggregated Mean	0.7635	0.9568	0.7635	0.7748	0.7607	0.7123	0.7156	0.6250	1.4716
(catboost) Aggregated Max	0.8108	0.9713	0.8108	0.8342	0.8116	0.7718	0.7773	0.7592	1.5739
(rf) base model	0.8068	0.9644	0.8068	0.8054	0.8056	0.7675	0.7677	0.6266	1.1641
(rf) tuned model	0.7959	0.9740	0.7959	0.7963	0.7951	0.7532	0.7535	0.5831	0.0013
(rf) Aggregated Mean	0.8176	0.9791	0.8176	0.8405	0.8168	0.7785	0.7827	0.5586	0.0007
(rf) Aggregated Max	0.8649	0.9883	0.8649	0.8878	0.8724	0.8364	0.8372	0.6560	0.0008
(et) base model	0.8376	0.9694	0.8376	0.8366	0.8352	0.8046	0.8053	0.6170	0.7612
(et) tuned model	0.8503	0.9763	0.8503	0.8571	0.8518	0.8189	0.8196	0.5174	0.0007
(et) Aggregated Mean	0.8514	0.9789	0.8514	0.8701	0.8520	0.8198	0.8233	0.4952	0.0007
(et) Aggregated Max	0.8919	0.9871	0.8919	0.9159	0.8903	0.8689	0.8744	0.5593	0.0008

Table 4.11: AW Dataset 80%/20% partition, k = 8

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.7983	0.9631	0.7983	0.7980	0.7980	0.7572	0.7573	0.5665	2.4177
(xgboost) tuned model	0.7823	0.9513	0.7823	0.7868	0.7829	0.7371	0.7377	0.6984	0.5446
(xgboost) Aggregated Mean	0.7566	0.9608	0.7566	0.8005	0.7467	0.7044	0.7158	0.6185	0.1846
(xgboost) Aggregated Max	0.8947	0.9934	0.8947	0.9079	0.8965	0.8725	0.8754	0.9152	0.2915
(lightgbm) base model	0.7897	0.9621	0.7897	0.7884	0.7888	0.7469	0.7470	0.5832	22.2125
(lightgbm) tuned model	0.7007	0.9311	0.7007	0.7088	0.7029	0.6376	0.6383	1.0995	1.2743
(lightgbm) Aggregated Mean	0.7632	0.9480	0.7632	0.7775	0.7516	0.7110	0.7203	0.7710	0.1759
(lightgbm) Aggregated Max	0.9474	1.0000	0.9474	0.9368	0.9248	0.9317	0.9351	1.0218	0.1927
(catboost) base model	0.8103	0.9663	0.8103	0.8096	0.8088	0.7715	0.7719	0.5478	20.8553
(catboost) tuned model	0.7279	0.9479	0.7279	0.7287	0.7277	0.6714	0.6715	0.6706	2.1150
(catboost) Aggregated Mean	0.8092	0.9732	0.8092	0.8423	0.8008	0.7671	0.7775	0.5150	0.9496
(catboost) Aggregated Max	0.8947	0.9934	0.8947	0.8904	0.8722	0.8633	0.8665	0.7883	1.0485
(rf) base model	0.8068	0.9644	0.8068	0.8054	0.8056	0.7675	0.7677	0.6266	1.0938
(rf) tuned model	0.7959	0.9740	0.7959	0.7963	0.7951	0.7532	0.7535	0.5831	0.0024
(rf) Aggregated Mean	0.7961	0.9788	0.7961	0.8237	0.7908	0.7516	0.7588	0.5638	0.0008
(rf) Aggregated Max	0.8947	0.9971	0.8947	0.9342	0.8985	0.8721	0.8783	0.6810	0.0012
(et) base model	0.8376	0.9694	0.8376	0.8366	0.8352	0.8046	0.8053	0.6170	0.9844
(et) tuned model	0.8503	0.9763	0.8503	0.8571	0.8518	0.8189	0.8196	0.5174	0.0008
(et) Aggregated Mean	0.8750	0.9801	0.8750	0.9113	0.8694	0.8486	0.8571	0.4902	0.0007
(et) Aggregated Max	0.9474	1.0000	0.9474	0.9737	0.9541	0.9362	0.9394	0.6615	0.0010

80%/20% partition, $k = 8$ The outcomes of all models trained in FL approach, with the exception of Extreme Gradient Boosting, surpass their CL counterparts, as shown in Table 4.11.

Table 4.12: AW Dataset 90%/10% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8237	0.9729	0.8237	0.8250	0.8236	0.7879	0.7882	0.5004	2.6865
(xgboost) tuned model	0.7432	0.9486	0.7432	0.7419	0.7380	0.6894	0.6913	0.7105	0.3945
(xgboost) Aggregated Mean	0.7162	0.9581	0.7162	0.7062	0.6765	0.6593	0.6770	0.7222	0.2993
(xgboost) Aggregated Max	0.8108	0.9730	0.8108	0.8286	0.7988	0.7714	0.7804	0.9329	0.3195
(lightgbm) base model	0.8343	0.9748	0.8343	0.8363	0.8345	0.8007	0.8010	0.4806	21.4490
(lightgbm) tuned model	0.6757	0.9161	0.6757	0.6687	0.6685	0.6078	0.6090	1.1399	0.7461
(lightgbm) Aggregated Mean	0.7433	0.9342	0.7433	0.7274	0.7188	0.6892	0.6991	0.8854	0.3028
(lightgbm) Aggregated Max	0.8108	0.9650	0.8108	0.8293	0.8070	0.7698	0.7753	1.2203	0.3090
(catboost) base model	0.8131	0.9731	0.8131	0.8151	0.8133	0.7748	0.7751	0.5046	20.0432
(catboost) tuned model	0.7703	0.9463	0.7703	0.7799	0.7671	0.7216	0.7243	0.6765	1.9775
(catboost) Aggregated Mean	0.7298	0.9551	0.7298	0.6894	0.6933	0.6741	0.6855	0.6715	1.5452
(catboost) Aggregated Max	0.7838	0.9835	0.7838	0.7921	0.7787	0.7383	0.7422	0.8794	1.7307
(rf) base model	0.8040	0.9713	0.8040	0.8087	0.8042	0.7641	0.7648	0.5850	1.1739
(rf) tuned model	0.8378	0.9665	0.8378	0.8381	0.8332	0.8042	0.8063	0.5854	0.0018
(rf) Aggregated Mean	0.8379	0.9812	0.8379	0.8368	0.8277	0.8034	0.8074	0.5394	0.0007
(rf) Aggregated Max	0.8919	0.9936	0.8919	0.9061	0.8953	0.8686	0.8701	0.6349	0.0007
(et) base model	0.8283	0.9766	0.8283	0.8321	0.8291	0.7933	0.7937	0.5058	0.8353
(et) tuned model	0.8108	0.9768	0.8108	0.8127	0.8055	0.7715	0.7744	0.4994	0.0007
(et) Aggregated Mean	0.7973	0.9798	0.7973	0.7995	0.7848	0.7545	0.7598	0.5197	0.0007
(et) Aggregated Max	0.8108	0.9864	0.8108	0.8315	0.8095	0.7708	0.7749	0.5861	0.0007

Table 4.13: AW Dataset 90%/10% partition, $k = 4$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8237	0.9729	0.8237	0.8250	0.8236	0.7879	0.7882	0.5004	2.3706
(xgboost) tuned model	0.7432	0.9486	0.7432	0.7419	0.7380	0.6894	0.6913	0.7105	0.3044
(xgboost) Aggregated Mean	0.7500	0.9480	0.7500	0.7820	0.7468	0.6947	0.7023	0.7267	0.1544
(xgboost) Aggregated Max	0.8421	0.9798	0.8421	0.8526	0.8413	0.8087	0.8115	1.0726	0.2007
(lightgbm) base model	0.8343	0.9748	0.8343	0.8363	0.8345	0.8007	0.8010	0.4806	20.3815
(lightgbm) tuned model	0.6757	0.9161	0.6757	0.6687	0.6685	0.6078	0.6090	1.1399	0.6169
(lightgbm) Aggregated Mean	0.6842	0.9154	0.6842	0.7556	0.6767	0.6125	0.6325	1.0546	0.1692
(lightgbm) Aggregated Max	0.7895	0.9824	0.7895	0.8105	0.7930	0.7441	0.7466	1.7082	0.1779
(catboost) base model	0.8131	0.9731	0.8131	0.8151	0.8133	0.7748	0.7751	0.5046	24.1421
(catboost) tuned model	0.7703	0.9463	0.7703	0.7799	0.7671	0.7216	0.7243	0.6765	2.1177
(catboost) Aggregated Mean	0.7763	0.9584	0.7763	0.8024	0.7751	0.7275	0.7329	0.6241	0.9871
(catboost) Aggregated Max	0.8947	0.9776	0.8947	0.9123	0.8947	0.8694	0.8724	0.7479	1.0456
(rf) base model	0.8040	0.9713	0.8040	0.8087	0.8042	0.7641	0.7648	0.5850	1.1360
(rf) tuned model	0.8378	0.9665	0.8378	0.8381	0.8332	0.8042	0.8063	0.5854	0.0017
(rf) Aggregated Mean	0.8290	0.9754	0.8290	0.8430	0.8217	0.7903	0.7957	0.5499	0.0011
(rf) Aggregated Max	0.8421	0.9885	0.8421	0.8833	0.8398	0.8094	0.8176	0.6207	0.0025
(et) base model	0.8283	0.9766	0.8283	0.8321	0.8291	0.7933	0.7937	0.5058	0.9292
(et) tuned model	0.8108	0.9768	0.8108	0.8127	0.8055	0.7715	0.7744	0.4994	0.0008
(et) Aggregated Mean	0.7500	0.9805	0.7500	0.7796	0.7282	0.6976	0.7129	0.5405	0.0008
(et) Aggregated Max	0.8421	0.9838	0.8421	0.8526	0.8398	0.8081	0.8108	0.5699	0.0010

90%/10% partition, $k = 2$ Table 4.12 illustrates how LightGBM FL surpasses CL by $\approx 7\%$; Random Forest FL is comparable to the CL edition.

90%/10% partition, $k = 4$ The findings from Table 4.13 indicate that the outcomes of Extreme Gradient Boosting, LightGBM, and CatBoost are in agreement.

Table 4.14: AW Dataset 90%/10% partition, $k = 8$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8237	0.9729	0.8237	0.8250	0.8236	0.7879	0.7882	0.5004	6.5692
(xgboost) tuned model	0.7432	0.9486	0.7432	0.7419	0.7380	0.6894	0.6913	0.7105	0.2755
(xgboost) Aggregated Mean	0.8125	0.7080	0.8125	0.8042	0.7921	0.7694	0.7827	0.5050	0.1252
(xgboost) Aggregated Max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.1684	0.2163
(lightgbm) base model	0.8343	0.9748	0.8343	0.8363	0.8345	0.8007	0.8010	0.4806	19.0952
(lightgbm) tuned model	0.6757	0.9161	0.6757	0.6687	0.6685	0.6078	0.6090	1.1399	0.6098
(lightgbm) Aggregated Mean	0.8111	0.7253	0.8111	0.8012	0.7928	0.7658	0.7766	0.3564	0.0367
(lightgbm) Aggregated Max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.7047	0.0431
(catboost) base model	0.8131	0.9731	0.8131	0.8151	0.8133	0.7748	0.7751	0.5046	20.6167
(catboost) tuned model	0.7703	0.9463	0.7703	0.7799	0.7671	0.7216	0.7243	0.6765	2.0081
(catboost) Aggregated Mean	0.8222	0.7127	0.8222	0.8188	0.8029	0.7787	0.7892	0.4420	0.6728
(catboost) Aggregated Max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8285	0.7043
(rf) base model	0.8040	0.9713	0.8040	0.8087	0.8042	0.7641	0.7648	0.5850	1.3310
(rf) tuned model	0.8378	0.9665	0.8378	0.8381	0.8332	0.8042	0.8063	0.5854	0.0019
(rf) Aggregated Mean	0.8222	0.7368	0.8222	0.8292	0.8061	0.7808	0.7966	0.4153	0.0007
(rf) Aggregated Max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.7037	0.0009
(et) base model	0.8283	0.9766	0.8283	0.8321	0.8291	0.7933	0.7937	0.5058	0.8748
(et) tuned model	0.8108	0.9768	0.8108	0.8127	0.8055	0.7715	0.7744	0.4994	0.0007
(et) Aggregated Mean	0.8097	0.7328	0.8097	0.8104	0.7886	0.7661	0.7822	0.7948	0.0008
(et) Aggregated Max	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	4.0672	0.0011

90%/10% partition, $k = 8$ The results of Extreme Gradient Boosting FL are $\approx 7\%$; better than CL; LightGBM FL performs $\approx 14\%$; than CL, and CatBoost FL is $\approx 5\%$; better than CL, as shown in Table 4.14.

HARTH dataset

The HARTH dataset offers 22 divisions for every individual. Regrettably, as a result of hardware restrictions, it is not feasible to run all of them, as each partition is considerable in size. Therefore, the 17th partition will be utilized as it is lightweight and offers an adequate number of classes. Table 4.15 offers a more in-depth look at the features of the HARTH dataset, such as the quantity of variables, observations, and types of variables. As AW Dataset, it will be tested using the same configuration.

Table 4.15: HARTH Dataset Statistics

Statistic	Value
Number of variables	7
Number of observations	110116
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	618
Duplicate rows (%)	0.6%
Total size in memory	5.9 MiB
Average record size in memory	56.0 B
Variable types	
Numeric	7

60%/40% partition, $k = 2$ The results shown in Table 4.16 demonstrate that the majority of models exhibit similar levels of performance.

Table 4.16: HARTH Dataset 60%/40% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.9328	0.9891	0.9328	0.9229	0.9237	0.8996	0.9007	0.2208	13.3276
(xgboost) tuned model	0.9355	0.9896	0.9355	0.9275	0.9288	0.9035	0.9043	0.2155	1.2953
(xgboost) Aggregated Mean	0.9314	0.9873	0.9314	0.9217	0.9236	0.8969	0.8980	0.2522	1.0037
(xgboost) Aggregated Max	0.9328	0.9875	0.9328	0.9237	0.9254	0.8986	0.8996	0.2536	1.0877
(lightgbm) base model	0.9311	0.9888	0.9311	0.9206	0.9223	0.8970	0.8981	0.2248	40.3384
(lightgbm) tuned model	0.8955	0.9616	0.8955	0.8876	0.8908	0.8458	0.8460	1.4233	3.4386
(lightgbm) Aggregated Mean	0.8802	0.9564	0.8802	0.8733	0.8760	0.8232	0.8234	1.6473	2.9145
(lightgbm) Aggregated Max	0.8823	0.9593	0.8823	0.8756	0.8781	0.8270	0.8272	1.8356	2.9404
(catboost) base model	0.9330	0.9897	0.9330	0.9230	0.9237	0.9000	0.9010	0.2145	74.4167
(catboost) tuned model	0.9356	0.9900	0.9356	0.9266	0.9280	0.9036	0.9045	0.2054	7.9251
(catboost) Aggregated Mean	0.9319	0.9881	0.9319	0.9214	0.9229	0.8977	0.8989	0.2287	5.3846
(catboost) Aggregated Max	0.9337	0.9882	0.9337	0.9246	0.9256	0.8999	0.9010	0.2302	5.3939
(rf) base model	0.9353	0.9896	0.9353	0.9271	0.9247	0.9028	0.9044	0.2886	17.6496
(rf) tuned model	0.9358	0.9892	0.9358	0.9279	0.9245	0.9031	0.9048	0.2681	0.0051
(rf) Aggregated Mean	0.9344	0.9881	0.9344	0.9253	0.9232	0.9007	0.9027	0.3082	0.0015
(rf) Aggregated Max	0.9361	0.9883	0.9361	0.9274	0.9251	0.9028	0.9047	0.3205	0.0015
(et) base model	0.9361	0.9898	0.9361	0.9295	0.9245	0.9039	0.9057	0.2976	7.4982
(et) tuned model	0.9372	0.9890	0.9372	0.9295	0.9244	0.9050	0.9069	0.2914	0.0023
(et) Aggregated Mean	0.9344	0.9887	0.9344	0.9267	0.9217	0.9005	0.9028	0.3057	0.0015
(et) Aggregated Max	0.9352	0.9889	0.9352	0.9271	0.9231	0.9012	0.9034	0.3067	0.0016

60%/40% partition, $k = 4$ Both Random Forest and Extra Trees models trained in FL are compatible with CL versions, according to Table 4.17.

Table 4.17: HARTH Dataset 60%/40% partition, k = 4

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.9328	0.9891	0.9328	0.9229	0.9237	0.8996	0.9007	0.2208	13.2385
(xgboost) tuned model	0.9355	0.9896	0.9355	0.9275	0.9288	0.9035	0.9043	0.2155	1.2946
(xgboost) Aggregated Mean	0.9277	0.9868	0.9277	0.9165	0.9190	0.8914	0.8927	0.2706	0.8847
(xgboost) Aggregated Max	0.9357	0.9886	0.9357	0.9252	0.9286	0.9034	0.9039	0.3115	1.0925
(lightgbm) base model	0.9311	0.9888	0.9311	0.9206	0.9223	0.8970	0.8981	0.2248	34.5899
(lightgbm) tuned model	0.8955	0.9616	0.8955	0.8876	0.8908	0.8458	0.8460	1.4233	4.9212
(lightgbm) Aggregated Mean	0.8455	0.9272	0.8455	0.8491	0.8463	0.7753	0.7757	3.3859	2.8819
(lightgbm) Aggregated Max	0.8812	0.9578	0.8812	0.8716	0.8748	0.8256	0.8259	5.7008	4.0864
(catboost) base model	0.9330	0.9897	0.9330	0.9230	0.9237	0.9000	0.9010	0.2145	74.8122
(catboost) tuned model	0.9356	0.9900	0.9356	0.9266	0.9280	0.9036	0.9045	0.2054	9.9470
(catboost) Aggregated Mean	0.9290	0.9881	0.9290	0.9167	0.9201	0.8936	0.8946	0.2300	4.5019
(catboost) Aggregated Max	0.9408	0.9905	0.9408	0.9307	0.9333	0.9109	0.9115	0.2578	5.2094
(rf) base model	0.9353	0.9896	0.9353	0.9271	0.9247	0.9028	0.9044	0.2886	21.8134
(rf) tuned model	0.9358	0.9892	0.9358	0.9279	0.9245	0.9031	0.9048	0.2681	0.0053
(rf) Aggregated Mean	0.9326	0.9894	0.9326	0.9231	0.9212	0.8981	0.8999	0.2860	0.0011
(rf) Aggregated Max	0.9412	0.9910	0.9412	0.9346	0.9313	0.9108	0.9120	0.3261	0.0012
(et) base model	0.9361	0.9898	0.9361	0.9295	0.9245	0.9039	0.9057	0.2976	8.4478
(et) tuned model	0.9372	0.9890	0.9372	0.9295	0.9244	0.9050	0.9069	0.2914	0.0026
(et) Aggregated Mean	0.9354	0.9894	0.9354	0.9279	0.9231	0.9023	0.9044	0.2890	0.0011
(et) Aggregated Max	0.9422	0.9906	0.9422	0.9343	0.9311	0.9123	0.9137	0.2945	0.0013

60%/40% partition, k = 8 Both Random Forest and Extra Trees models trained in Federated Learning are shown to be compatible with Centralized Learning versions in Table 4.18.

Table 4.18: HARTH Dataset 60%/40% partition, k = 8

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.9328	0.9891	0.9328	0.9229	0.9237	0.8996	0.9007	0.2208	12.4472
(xgboost) tuned model	0.9355	0.9896	0.9355	0.9275	0.9288	0.9035	0.9043	0.2155	1.4491
(xgboost) Aggregated Mean	0.9226	0.9848	0.9226	0.9100	0.9129	0.8835	0.8848	0.3185	0.6731
(xgboost) Aggregated Max	0.9296	0.9868	0.9296	0.9196	0.9209	0.8947	0.8961	0.3714	0.8517
(lightgbm) base model	0.9311	0.9888	0.9311	0.9206	0.9223	0.8970	0.8981	0.2248	34.8883
(lightgbm) tuned model	0.8955	0.9616	0.8955	0.8876	0.8908	0.8458	0.8460	1.4233	4.4794
(lightgbm) Aggregated Mean	0.7931	0.8631	0.7931	0.7971	0.7813	0.6775	0.6900	5.6852	1.9795
(lightgbm) Aggregated Max	0.8453	0.9415	0.8453	0.8580	0.8510	0.7774	0.7780	13.4280	2.3952
(catboost) base model	0.9330	0.9897	0.9330	0.9230	0.9237	0.9000	0.9010	0.2145	81.2461
(catboost) tuned model	0.9356	0.9900	0.9356	0.9266	0.9280	0.9036	0.9045	0.2054	8.5314
(catboost) Aggregated Mean	0.9269	0.9867	0.9269	0.9142	0.9166	0.8900	0.8913	0.2509	3.7142
(catboost) Aggregated Max	0.9303	0.9894	0.9303	0.9198	0.9224	0.8951	0.8967	0.2837	6.0738
(rf) base model	0.9353	0.9896	0.9353	0.9271	0.9247	0.9028	0.9044	0.2886	21.7904
(rf) tuned model	0.9358	0.9892	0.9358	0.9279	0.9245	0.9031	0.9048	0.2681	0.0059
(rf) Aggregated Mean	0.9318	0.9892	0.9318	0.9237	0.9200	0.8969	0.8988	0.2786	0.0009
(rf) Aggregated Max	0.9368	0.9920	0.9368	0.9375	0.9263	0.9056	0.9070	0.3075	0.0013
(et) base model	0.9361	0.9898	0.9361	0.9295	0.9245	0.9039	0.9057	0.2976	8.1979
(et) tuned model	0.9372	0.9890	0.9372	0.9295	0.9244	0.9050	0.9069	0.2914	0.0025
(et) Aggregated Mean	0.9332	0.9895	0.9332	0.9263	0.9206	0.8988	0.9009	0.2922	0.0010
(et) Aggregated Max	0.9397	0.9920	0.9397	0.9354	0.9292	0.9095	0.9115	0.3508	0.0012

70%/30% partition, k = 2 Nearly all models, with the exception of LightGBM, are found to be compatible based on the information in Table 4.19.

Table 4.19: HARTH Dataset 70%/30% partition, k = 2

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.9335	0.9899	0.9335	0.9244	0.9247	0.9005	0.9016	0.2079	11.6420
(xgboost) tuned model	0.9313	0.9888	0.9313	0.9235	0.9249	0.8973	0.8982	0.2359	1.8353
(xgboost) Aggregated Mean	0.9325	0.9883	0.9325	0.9238	0.9242	0.8987	0.8999	0.2499	0.8735
(xgboost) Aggregated Max	0.9344	0.9885	0.9344	0.9264	0.9268	0.9018	0.9028	0.2579	0.8804
(lightgbm) base model	0.9328	0.9893	0.9328	0.9232	0.9245	0.8994	0.9005	0.2143	35.9614
(lightgbm) tuned model	0.8570	0.9273	0.8570	0.8538	0.8539	0.7905	0.7910	2.9988	4.2112
(lightgbm) Aggregated Mean	0.8267	0.8949	0.8267	0.8215	0.8220	0.7460	0.7468	4.1403	2.9470
(lightgbm) Aggregated Max	0.8562	0.9343	0.8562	0.8567	0.8558	0.7905	0.7908	5.5076	2.9874
(catboost) base model	0.9340	0.9899	0.9340	0.9253	0.9247	0.9011	0.9023	0.2083	91.8426
(catboost) tuned model	0.9316	0.9894	0.9316	0.9232	0.9244	0.8977	0.8986	0.2176	7.2403
(catboost) Aggregated Mean	0.9348	0.9891	0.9348	0.9266	0.9268	0.9023	0.9033	0.2199	5.1975
(catboost) Aggregated Max	0.9358	0.9892	0.9358	0.9277	0.9268	0.9036	0.9049	0.2204	5.2990
(rf) base model	0.9374	0.9895	0.9374	0.9305	0.9269	0.9057	0.9075	0.2920	25.4742
(rf) tuned model	0.9350	0.9890	0.9350	0.9259	0.9239	0.9020	0.9037	0.2934	0.0044
(rf) Aggregated Mean	0.9379	0.9896	0.9379	0.9311	0.9281	0.9065	0.9081	0.2903	0.0015
(rf) Aggregated Max	0.9390	0.9904	0.9390	0.9331	0.9293	0.9080	0.9097	0.3075	0.0015
(et) base model	0.9376	0.9903	0.9376	0.9322	0.9267	0.9059	0.9078	0.2693	9.4903
(et) tuned model	0.9349	0.9899	0.9349	0.9286	0.9226	0.9016	0.9036	0.2894	0.0019
(et) Aggregated Mean	0.9386	0.9902	0.9386	0.9351	0.9279	0.9074	0.9093	0.2875	0.0014
(et) Aggregated Max	0.9387	0.9908	0.9387	0.9358	0.9287	0.9078	0.9094	0.2955	0.0015

Table 4.20: HARTH Dataset 70%/30% partition, k = 4

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.9335	0.9899	0.9335	0.9244	0.9247	0.9005	0.9016	0.2079	10.5623
(xgboost) tuned model	0.9313	0.9888	0.9313	0.9235	0.9249	0.8973	0.8982	0.2359	1.2358
(xgboost) Aggregated Mean	0.9281	0.9867	0.9281	0.9184	0.9196	0.8918	0.8933	0.2814	0.7820
(xgboost) Aggregated Max	0.9308	0.9892	0.9308	0.9209	0.9222	0.8958	0.8973	0.2972	1.1756
(lightgbm) base model	0.9328	0.9893	0.9328	0.9232	0.9245	0.8994	0.9005	0.2143	35.7436
(lightgbm) tuned model	0.8570	0.9273	0.8570	0.8538	0.8539	0.7905	0.7910	2.9988	3.3279
(lightgbm) Aggregated Mean	0.7909	0.8736	0.7909	0.8024	0.7955	0.6990	0.7001	6.0164	2.6250
(lightgbm) Aggregated Max	0.8344	0.9012	0.8344	0.8363	0.8349	0.7583	0.7584	7.5852	3.2218
(catboost) base model	0.9340	0.9899	0.9340	0.9253	0.9247	0.9011	0.9023	0.2083	100.6635
(catboost) tuned model	0.9316	0.9894	0.9316	0.9232	0.9244	0.8977	0.8986	0.2176	6.9749
(catboost) Aggregated Mean	0.9297	0.9877	0.9297	0.9214	0.9216	0.8943	0.8957	0.2360	4.1107
(catboost) Aggregated Max	0.9361	0.9899	0.9361	0.9276	0.9271	0.9037	0.9053	0.2470	4.3133
(rf) base model	0.9374	0.9895	0.9374	0.9305	0.9269	0.9057	0.9075	0.2920	27.1224
(rf) tuned model	0.9350	0.9890	0.9350	0.9259	0.9239	0.9020	0.9037	0.2934	0.0057
(rf) Aggregated Mean	0.9351	0.9899	0.9351	0.9294	0.9255	0.9021	0.9039	0.2583	0.0014
(rf) Aggregated Max	0.9390	0.9911	0.9390	0.9349	0.9310	0.9080	0.9097	0.2780	0.0027
(et) base model	0.9376	0.9903	0.9376	0.9322	0.9267	0.9059	0.9078	0.2693	8.8603
(et) tuned model	0.9349	0.9899	0.9349	0.9286	0.9226	0.9016	0.9036	0.2894	0.0020
(et) Aggregated Mean	0.9375	0.9902	0.9375	0.9353	0.9263	0.9055	0.9077	0.2826	0.0011
(et) Aggregated Max	0.9433	0.9918	0.9433	0.9425	0.9334	0.9143	0.9164	0.2938	0.0012

70%/30% partition, $k = 4$ Both Random Forest and Extra Trees models trained using FL techniques show compatibility with CL versions, as indicated in Table 4.20. Due to LaTeX platform limitations, unnecessary tables have been removed as the outcomes from different distributions (80%/20% and 90%/10%) are almost identical. However, these findings can be found in the external excel document as outlined in section 4.3 "Analysis of Real Data".

4.3.2 Regression

This section discusses the experiment conducted on the data set utilized for Regression. Note that results will be evaluated by MAE metric, although other types of metrics are also available, e.g. R2 which is more similar to Accuracy.

MiBand dataset

In Table 4.21, the specific attributes of the Mi Band dataset are presented. The statistics indicate that the MiBand Dataset is organized and clean, making it suitable for analysis without any necessary preprocessing for missing data or duplicates. The distribution settings are the same.

Table 4.21: MiBand Dataset Statistics

Statistic	Value
Number of variables	5
Number of observations	2454
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	96.0 KiB
Average record size in memory	40.1 B
Variable types	
DateTime	1
Numeric	4

60%/40% partition, $k = 2$ Table 4.22 demonstrates that Extreme Gradient Boosting and Random Forest models performed similarly in FL and CL training, while Extra Trees FL showed superior performance compared to its CL counterpart.

Table 4.22: MiBand Dataset 60%/40% partition, $k = 2$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.0747	9468.5647	97.3066	0.7712	0.3020	0.1988	0.9002
(xgboost) tuned model	61.0546	12514.5606	111.8685	0.6926	0.4402	0.2505	0.0672
(xgboost) Aggregated Mean	61.2773	12872.8869	113.4315	0.6851	0.3804	0.2709	0.1366
(xgboost) Aggregated Max	66.5588	13437.1169	115.9186	0.7383	0.4001	0.2779	0.1916
(lightgbm) base model	48.2992	8393.8192	91.6178	0.7971	0.3735	0.2094	4.2602
(lightgbm) tuned model	51.9401	8498.7541	92.1887	0.7913	0.3748	0.2293	0.2774
(lightgbm) Aggregated Mean	54.9092	9344.4710	96.0003	0.7660	0.3740	0.2253	0.1368
(lightgbm) Aggregated Max	61.5715	11520.1998	107.3322	0.8476	0.4159	0.2273	0.1460
(catboost) base model	43.7343	8321.4017	91.2217	0.7989	0.3471	0.1750	6.0612
(catboost) tuned model	47.1213	8880.4177	94.2360	0.7819	0.3184	0.1725	0.4656
(catboost) Aggregated Mean	56.4724	18232.8563	132.6429	0.5359	0.3591	0.2162	0.4816
(catboost) Aggregated Max	65.3370	24937.4828	157.9161	0.7549	0.4136	0.2220	0.4907
(rf) base model	44.8732	9544.0494	97.6937	0.7693	0.2716	0.1790	1.5234
(rf) tuned model	51.3832	10650.0131	103.1989	0.7384	0.3277	0.2004	0.0109
(rf) Aggregated Mean	51.8509	11292.5294	105.7146	0.7183	0.3161	0.1921	0.0007
(rf) Aggregated Max	60.1059	13579.1012	116.5294	0.8085	0.3481	0.1959	0.0008
(et) base model	48.5508	10797.5340	103.9112	0.7390	0.2901	0.1901	0.8028
(et) tuned model	55.2975	11193.7159	105.8004	0.7251	0.3477	0.2249	0.0007
(et) Aggregated Mean	48.5280	9822.5582	97.7883	0.7514	0.2863	0.1739	0.0008
(et) Aggregated Max	58.0632	12976.1965	113.9131	0.8582	0.3428	0.2013	0.0009

Table 4.23: MiBand Dataset 60%/40% partition, $k = 4$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.0747	9468.5647	97.3066	0.7712	0.3020	0.1988	4.8670
(xgboost) tuned model	61.0546	12514.5606	111.8685	0.6926	0.4402	0.2505	0.0739
(xgboost) Aggregated Mean	72.9153	21893.2614	145.5142	0.6282	0.3826	0.2676	0.0672
(xgboost) Aggregated Max	78.3026	28215.9267	167.9760	0.8000	0.4875	0.3440	0.0975
(lightgbm) base model	48.2992	8393.8192	91.6178	0.7971	0.3735	0.2094	2.9490
(lightgbm) tuned model	51.9401	8498.7541	92.1887	0.7913	0.3748	0.2293	0.5214
(lightgbm) Aggregated Mean	61.7270	14677.4064	119.4821	0.7493	0.3382	0.2383	0.0636
(lightgbm) Aggregated Max	72.4870	21761.9855	147.5194	0.8170	0.3943	0.2963	0.0702
(catboost) base model	43.7343	8321.4017	91.2217	0.7989	0.3471	0.1750	2.5253
(catboost) tuned model	47.1213	8880.4177	94.2360	0.7819	0.3184	0.1725	0.4636
(catboost) Aggregated Mean	56.7613	15055.7841	121.5429	0.7428	0.3215	0.1967	0.3416
(catboost) Aggregated Max	60.2828	19832.9884	140.8296	0.8248	0.3911	0.2543	0.3629
(rf) base model	44.8732	9544.0494	97.6937	0.7693	0.2716	0.1790	0.6634
(rf) tuned model	51.3832	10650.0131	103.1989	0.7384	0.3277	0.2004	0.0010
(rf) Aggregated Mean	52.2310	12706.0984	111.6320	0.7828	0.2461	0.1651	0.0006
(rf) Aggregated Max	60.1086	18390.4604	135.6114	0.8259	0.2752	0.1738	0.0007
(et) base model	48.5508	10797.5340	103.9112	0.7390	0.2901	0.1901	0.5685
(et) tuned model	55.2975	11193.7159	105.8004	0.7251	0.3477	0.2249	0.0007
(et) Aggregated Mean	58.7153	15212.9074	122.4568	0.7386	0.2638	0.1848	0.0006
(et) Aggregated Max	67.1352	21781.4823	147.5855	0.7968	0.2956	0.2217	0.0007

60%/40% partition, $k = 4$ Only the Random Forest FL outperforms CL in Table 4.23.

60%/40% partition, $k = 8$ Table 4.22 displays that Federated Random Forest and Federated Extra Trees models outperform the centralized tuned model by approximately 3-4 times in terms of MAE performance.

Table 4.24: MiBand Dataset 60%/40% partition, $k = 8$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.0747	9468.5647	97.3066	0.7712	0.3020	0.1988	1.0011
(xgboost) tuned model	61.0546	12514.5606	111.8685	0.6926	0.4402	0.2505	0.0902
(xgboost) Aggregated Mean	65.8528	14292.4846	116.7600	0.6813	0.4616	0.2936	0.0711
(xgboost) Aggregated Max	91.1468	26555.1031	162.9574	0.8966	0.8915	0.3447	0.1344
(lightgbm) base model	48.2992	8393.8192	91.6178	0.7971	0.3735	0.2094	3.2402
(lightgbm) tuned model	51.9401	8498.7541	92.1887	0.7913	0.3748	0.2293	0.2136
(lightgbm) Aggregated Mean	58.0431	10409.9942	98.2643	0.7832	0.4323	0.3915	0.0372
(lightgbm) Aggregated Max	72.6083	23022.0507	151.7302	0.9037	1.1222	1.3737	0.0456
(catboost) base model	43.7343	8321.4017	91.2217	0.7989	0.3471	0.1750	1.9932
(catboost) tuned model	47.1213	8880.4177	94.2360	0.7819	0.3184	0.1725	0.4585
(catboost) Aggregated Mean	49.5009	10216.8653	95.9313	0.7838	0.3676	0.2178	0.2510
(catboost) Aggregated Max	75.8130	25201.1507	158.7487	0.9596	0.6177	0.4349	0.2894
(rf) base model	44.8732	9544.0494	97.6937	0.7693	0.2716	0.1790	0.7158
(rf) tuned model	51.3832	10650.0131	103.1989	0.7384	0.3277	0.2004	0.0013
(rf) Aggregated Mean	48.9095	10882.7035	98.8223	0.7783	0.2673	0.1707	0.0006
(rf) Aggregated Max	78.8160	26219.4580	161.9242	0.9072	0.4080	0.2370	0.0007
(et) base model	48.5508	10797.5340	103.9112	0.7390	0.2901	0.1901	0.5421
(et) tuned model	55.2975	11193.7159	105.8004	0.7251	0.3477	0.2249	0.0006
(et) Aggregated Mean	51.9628	11277.6568	99.5822	0.7653	0.2624	0.1758	0.0006
(et) Aggregated Max	81.8104	27237.7890	165.0387	0.9409	0.3929	0.2209	0.0006

70%/30% partition, $k = 2$ In Table 4.25, models trained with federated learning outperform those trained with centralized learning.

Table 4.25: MiBand Dataset 70%/30% partition, k = 2

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.2418	18034.8089	134.2937	0.6968	0.3014	0.1796	4.0637
(xgboost) tuned model	72.3481	16601.8166	128.8480	0.6824	0.3701	0.2726	0.1108
(xgboost) Aggregated Mean	61.6486	16238.9933	126.2697	0.6675	0.4095	0.2614	0.0838
(xgboost) Aggregated Max	68.9009	20576.1039	143.4437	0.7609	0.4613	0.2865	0.0938
(lightgbm) base model	46.8321	14267.4279	119.4463	0.7601	0.3569	0.1711	3.3562
(lightgbm) tuned model	61.9230	12744.4247	112.8912	0.7562	0.4218	0.2589	0.1968
(lightgbm) Aggregated Mean	52.9352	9744.3611	98.4083	0.8017	0.4171	0.3171	0.0901
(lightgbm) Aggregated Max	55.2845	11271.0831	106.1654	0.8299	0.4819	0.3654	0.0903
(catboost) base model	42.1258	15177.5706	123.1973	0.7448	0.3732	0.1502	2.5706
(catboost) tuned model	56.7003	12551.4025	112.0330	0.7599	0.3252	0.2038	0.5050
(catboost) Aggregated Mean	48.4646	10329.0269	101.4444	0.7897	0.3325	0.2274	0.4272
(catboost) Aggregated Max	50.5482	11581.0216	107.6152	0.8121	0.3667	0.2557	0.4317
(rf) base model	44.7761	16172.4188	127.1708	0.7281	0.2302	0.1574	0.7968
(rf) tuned model	55.2217	12720.4156	112.7848	0.7566	0.2507	0.1757	0.0015
(rf) Aggregated Mean	43.8121	7682.5786	87.0258	0.8439	0.2448	0.1642	0.0007
(rf) Aggregated Max	49.4095	9500.4296	97.4701	0.8786	0.2587	0.1884	0.0007
(et) base model	44.7958	16257.8704	127.5064	0.7267	0.2330	0.1574	0.5441
(et) tuned model	58.0764	13195.2819	114.8707	0.7476	0.2873	0.2043	0.0006
(et) Aggregated Mean	44.1404	7352.8722	84.8364	0.8507	0.2348	0.1567	0.0006
(et) Aggregated Max	50.8268	9469.7925	97.3129	0.8916	0.2624	0.1939	0.0007

Table 4.26: MiBand Dataset 70%/30% partition, k = 4

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.2418	18034.8089	134.2937	0.6968	0.3014	0.1796	3.8866
(xgboost) tuned model	72.3481	16601.8166	128.8480	0.6824	0.3701	0.2726	0.1320
(xgboost) Aggregated Mean	51.8286	9225.6633	93.6985	0.7306	0.3256	0.2428	0.1282
(xgboost) Aggregated Max	59.6258	14444.2826	120.1844	0.7731	0.3773	0.2878	0.1427
(lightgbm) base model	46.8321	14267.4279	119.4463	0.7601	0.3569	0.1711	3.9563
(lightgbm) tuned model	61.9230	12744.4247	112.8912	0.7562	0.4218	0.2589	0.2075
(lightgbm) Aggregated Mean	43.0456	4488.3370	66.2351	0.8594	0.4702	0.2885	0.0669
(lightgbm) Aggregated Max	51.2148	6200.9669	78.7462	0.8942	0.7068	0.3544	0.0919
(catboost) base model	42.1258	15177.5706	123.1973	0.7448	0.3732	0.1502	4.2972
(catboost) tuned model	56.7003	12551.4025	112.0330	0.7599	0.3252	0.2038	0.4602
(catboost) Aggregated Mean	41.0056	7506.3694	83.3891	0.7838	0.3475	0.2154	0.2952
(catboost) Aggregated Max	47.9707	12426.8499	111.4758	0.8748	0.4894	0.2810	0.3031
(rf) base model	44.7761	16172.4188	127.1708	0.7281	0.2302	0.1574	0.9849
(rf) tuned model	55.2217	12720.4156	112.7848	0.7566	0.2507	0.1757	0.0012
(rf) Aggregated Mean	33.7709	4709.4362	65.5356	0.8683	0.1731	0.1369	0.0007
(rf) Aggregated Max	43.0441	7021.7078	83.7956	0.9354	0.1979	0.1621	0.0008
(et) base model	44.7958	16257.8704	127.5064	0.7267	0.2330	0.1574	0.5382
(et) tuned model	58.0764	13195.2819	114.8707	0.7476	0.2873	0.2043	0.0006
(et) Aggregated Mean	36.0606	5479.1454	73.7365	0.8168	0.2159	0.1579	0.0012
(et) Aggregated Max	41.7705	6942.5164	83.3218	0.8745	0.2416	0.1953	0.0030

70%/30% partition, k = 4 All models trained in FL outperform CL in Table 4.26 as well.

70%/30% partition, k = 8 In Table 4.27, LightGBM, CatBoost, and Random Forest showed improved performance with FL over CL.

Table 4.27: MiBand Dataset 70%/30% partition, k = 8

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	48.2418	18034.8089	134.2937	0.6968	0.3014	0.1796	4.0826
(xgboost) tuned model	58.7379	11307.2813	106.3357	0.7705	0.3399	0.2651	0.1005
(xgboost) Aggregated Mean	71.0760	18266.0692	127.4373	0.0023	0.5482	0.3487	0.1554
(xgboost) Aggregated Max	98.6213	55899.4962	236.4307	0.8845	1.1270	0.6697	0.6194
(lightgbm) base model	46.8321	14267.4279	119.4463	0.7601	0.3569	0.1711	4.0353
(lightgbm) tuned model	47.8894	7403.3978	86.0430	0.8497	0.3129	0.2226	0.1745
(lightgbm) Aggregated Mean	44.7323	4629.8497	65.6672	0.7835	0.6043	0.2784	0.0314
(lightgbm) Aggregated Max	64.5276	7241.3780	85.0963	0.9375	1.1944	0.6387	0.0596
(catboost) base model	42.1258	15177.5706	123.1973	0.7448	0.3732	0.1502	2.8091
(catboost) tuned model	52.7538	14496.1859	120.4001	0.7058	0.3239	0.2090	0.4324
(catboost) Aggregated Mean	50.4241	11815.6030	92.9370	0.4961	0.3837	0.2344	0.2021
(catboost) Aggregated Max	101.8126	54629.2151	233.7289	0.9060	0.6198	0.2964	0.2266
(rf) base model	44.7761	16172.4188	127.1708	0.7281	0.2302	0.1574	0.7425
(rf) tuned model	50.3904	8975.5345	94.7393	0.8178	0.2559	0.1906	0.0012
(rf) Aggregated Mean	36.4290	4484.3577	64.5135	0.7966	0.2634	0.1801	0.0006
(rf) Aggregated Max	50.2208	10094.6983	100.4724	0.9495	0.4290	0.2609	0.0007
(et) base model	44.7958	16257.8704	127.5064	0.7267	0.2330	0.1574	0.5361
(et) tuned model	46.1143	7947.0719	89.1464	0.8387	0.2644	0.1816	0.0006
(et) Aggregated Mean	47.0009	8269.8335	86.8566	0.5776	0.3384	0.2507	0.0007
(et) Aggregated Max	63.2915	12548.5418	112.0203	0.9161	0.4439	0.4579	0.0009

Due to the limitations of the platform, further results can be found in the excel document referenced in section 4.3 "Analysis of Real Data".

DAT263x dataset

Table 4.28 displays the particular characteristics of the DAT263x dataset. The data shows that the DAT263x Dataset has 12000 entries, significantly surpassing MiBand, giving it greater resilience. The distribution configurations remain unchanged.

Table 4.28: DAT263x Dataset Statistics

Statistic	Value
Number of variables	8
Number of observations	12000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	1
Duplicate rows (%)	< 0.1%
Total size in memory	750.1 KiB
Average record size in memory	64.0 B
Variable types	
Categorical	1
Numeric	7

60%/40% partition, $k = 2$ In Table 4.29, Random Forest and Extra Trees continue to show superior performance over CL in FL.

Table 4.29: DAT263x Dataset 60%/40% partition, $k = 2$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	1.7279	7.4491	2.7293	0.9982	0.0407	0.0295	5.3248
(xgboost) tuned model	1.5651	5.5451	2.3548	0.9986	0.0432	0.0266	0.1047
(xgboost) Aggregated Mean	1.6708	5.9676	2.4424	0.9985	0.0489	0.0295	0.1068
(xgboost) Aggregated Max	1.6795	6.2160	2.4932	0.9985	0.0495	0.0298	0.1340
(lightgbm) base model	1.5598	7.5482	2.7474	0.9982	0.0440	0.0280	4.3894
(lightgbm) tuned model	1.2272	3.7107	1.9263	0.9990	0.0424	0.0240	0.3587
(lightgbm) Aggregated Mean	1.2692	3.3905	1.8387	0.9991	0.0433	0.0253	0.2675
(lightgbm) Aggregated Max	1.3079	3.7529	1.9372	0.9992	0.0458	0.0254	0.2751
(catboost) base model	0.5759	1.3657	1.1686	0.9997	0.0285	0.0150	3.8013
(catboost) tuned model	0.4766	0.9054	0.9515	0.9998	0.0309	0.0136	0.7304
(catboost) Aggregated Mean	0.5039	0.5181	0.7185	0.9999	0.0339	0.0154	0.5110
(catboost) Aggregated Max	0.5225	0.5814	0.7625	0.9999	0.0381	0.0164	0.5228
(rf) base model	2.3024	16.6014	4.0745	0.9960	0.0460	0.0321	3.1244
(rf) tuned model	2.3202	14.8195	3.8496	0.9962	0.0519	0.0345	0.0023
(rf) Aggregated Mean	2.1857	12.0791	3.4733	0.9969	0.0490	0.0323	0.0008
(rf) Aggregated Max	2.2802	12.9400	3.5972	0.9971	0.0504	0.0332	0.0009
(et) base model	1.7461	10.3187	3.2123	0.9975	0.0365	0.0250	2.3966
(et) tuned model	1.8534	9.8652	3.1409	0.9975	0.0416	0.0276	0.0009
(et) Aggregated Mean	1.7146	7.4229	2.7234	0.9981	0.0414	0.0262	0.0008
(et) Aggregated Max	1.7837	7.8573	2.8031	0.9982	0.0426	0.0272	0.0009

Table 4.30: DAT263x Dataset 60%/40% partition, $k = 4$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	1.7279	7.4491	2.7293	0.9982	0.0407	0.0295	1.8481
(xgboost) tuned model	1.5651	5.5451	2.3548	0.9986	0.0432	0.0266	0.2590
(xgboost) Aggregated Mean	1.7581	6.2075	2.4896	0.9983	0.0605	0.0370	0.0781
(xgboost) Aggregated Max	1.7909	6.7121	2.5908	0.9984	0.0830	0.0542	0.1158
(lightgbm) base model	1.5598	7.5482	2.7474	0.9982	0.0440	0.0280	3.5867
(lightgbm) tuned model	1.2272	3.7107	1.9263	0.9990	0.0424	0.0240	0.2660
(lightgbm) Aggregated Mean	1.3890	3.6360	1.9062	0.9991	0.0492	0.0277	0.2705
(lightgbm) Aggregated Max	1.4400	3.9083	1.9769	0.9992	0.0560	0.0321	0.3447
(catboost) base model	0.5759	1.3657	1.1686	0.9997	0.0285	0.0150	3.2958
(catboost) tuned model	0.4766	0.9054	0.9515	0.9998	0.0309	0.0136	0.7328
(catboost) Aggregated Mean	0.5377	0.6528	0.7930	0.9999	0.0339	0.0150	0.3798
(catboost) Aggregated Max	0.5739	1.1220	1.0592	0.9999	0.0443	0.0190	0.3837
(rf) base model	2.3024	16.6014	4.0745	0.9960	0.0460	0.0321	2.3748
(rf) tuned model	2.3202	14.8195	3.8496	0.9962	0.0519	0.0345	0.0012
(rf) Aggregated Mean	2.1986	11.6698	3.4099	0.9968	0.0479	0.0339	0.0008
(rf) Aggregated Max	2.4391	13.7882	3.7132	0.9975	0.0585	0.0414	0.0008
(et) base model	1.7461	10.3187	3.2123	0.9975	0.0365	0.0250	1.8135
(et) tuned model	1.8534	9.8652	3.1409	0.9975	0.0416	0.0276	0.0007
(et) Aggregated Mean	1.6838	6.8948	2.6184	0.9982	0.0411	0.0272	0.0007
(et) Aggregated Max	1.8214	7.9379	2.8174	0.9987	0.0508	0.0335	0.0007

Table 4.31: DAT263x Dataset 60%/40% partition, $k = 8$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
(xgboost) base model	1.7279	7.4491	2.7293	0.9982	0.0407	0.0295	0.8728
(xgboost) tuned model	1.5651	5.5451	2.3548	0.9986	0.0432	0.0266	0.2152
(xgboost) Aggregated Mean	1.8993	7.0444	2.6418	0.9982	0.0500	0.0324	0.0850
(xgboost) Aggregated Max	2.1789	10.1208	3.1813	0.9987	0.0683	0.0389	0.1474
(lightgbm) base model	1.5598	7.5482	2.7474	0.9982	0.0440	0.0280	3.8074
(lightgbm) tuned model	1.2272	3.7107	1.9263	0.9990	0.0424	0.0240	0.2520
(lightgbm) Aggregated Mean	1.5329	4.7507	2.1723	0.9988	0.0482	0.0283	0.1747
(lightgbm) Aggregated Max	1.7390	6.1614	2.4822	0.9991	0.0747	0.0332	0.2157
(catboost) base model	0.5759	1.3657	1.1686	0.9997	0.0285	0.0150	3.1783
(catboost) tuned model	0.4766	0.9054	0.9515	0.9998	0.0309	0.0136	0.7062
(catboost) Aggregated Mean	0.5510	0.5812	0.7604	0.9999	0.0302	0.0146	0.3241
(catboost) Aggregated Max	0.6314	0.7032	0.8386	0.9999	0.0386	0.0193	0.3724
(rf) base model	2.3024	16.6014	4.0745	0.9960	0.0460	0.0321	2.5569
(rf) tuned model	2.3202	14.8195	3.8496	0.9962	0.0519	0.0345	0.0021
(rf) Aggregated Mean	2.1912	11.2689	3.3314	0.9972	0.0478	0.0320	0.0020
(rf) Aggregated Max	2.6197	15.2966	3.9111	0.9983	0.0629	0.0354	0.0078
(et) base model	1.7461	10.3187	3.2123	0.9975	0.0365	0.0250	2.3202
(et) tuned model	1.8534	9.8652	3.1409	0.9975	0.0416	0.0276	0.0011
(et) Aggregated Mean	1.7129	6.8769	2.6081	0.9983	0.0393	0.0258	0.0008
(et) Aggregated Max	2.0910	9.8368	3.1364	0.9988	0.0495	0.0285	0.0009

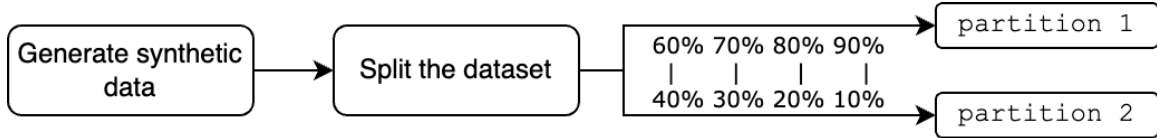
60%/40% partition, $k = 4, 8$ Table 4.30 and Table 4.31 reveal that both Random Forest and Extra Trees trained in FL outperform their CL counterparts, with minimal changes in the results. Because of the same cause, like in past trials, complete findings

are available in section 4.3 "Analysis of Real Data".

4.4 Analysis of Synthetic Data

In this section, another analysis has been conducted as a POC where the results were obtained by using a complete set of synthetic data generated on a dataset, such that the evaluations have been done exactly like in Figure 4-1 and Figure 4-2, but the initial dataset went through the process of synthetic data generation (see Figure 4-3), which is discussed later. Attention is given to how this entire artificial dataset behaves and what it can reveal. The objective is to assess its effectiveness in imitating real-life situations and understanding the insights it offers when maintained as a single entity. Through an examination of the undivided dataset, its fundamental qualities and limitations are brought to light.

Figure 4-3: Generation of Synthetic Data



4.4.1 Classification

Since the real data might be non-iid, partitioning can hugely affect the results of CL and FL. That is why it has been decided to use Synthetic Minority Over-sampling Technique (SMOTE) to synthesize the data as it has been proven in [60], where researchers compared different ML models in analyzing fitness data with and without SMOTE and they revealed that SMOTE significantly increases the performance of the models.

The SMOTE is a method commonly used to address imbalanced datasets by generating synthetic samples for the minority class. This technique involves creating new instances by interpolating between existing minority class samples. SMOTE works by identifying the nearest neighbors of each minority class instance and creating new synthetic examples by interpolating between these neighbors. These synthetic

samples are designed to balance the class distribution, thereby preventing overfitting that may occur with standard oversampling methods.

Due to time limitations, SMOTE has been applied to the dataset discussed in section 3.3.1 "Harvard's Apple Watch physical activity dataset". Figure 4-4 shows graphical representations of data quality analysis, focusing on the similarity between real and synthetic data and the numerical correlations within both datasets. It suggests that the synthetic data closely mimics the real data, with an average similarity score of 0.99 on a scale from 0 to 1, where 1 indicates perfect similarity. Another Figure 4-5 represents the distribution of the data in the *Age* column. As can be seen from the density plot, Synthetic and Real data fit almost perfectly.

Figure 4-4: Data Quality of Classification dataset using SMOTE.

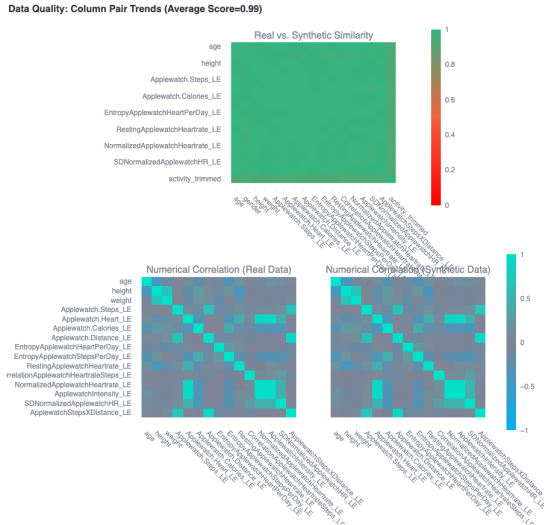
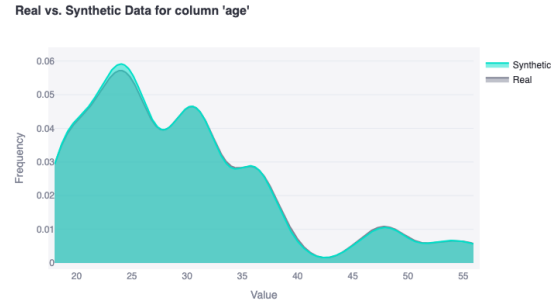


Figure 4-5: Classification: SMOTE vs Real data distribution.



Now, this synthetic dataset can be used for training. Table 4.32 convey statistics for a synthetic Apple Watch dataset. The dataset contains 3774 observations, and 17 variables (16 numeric and one categorical), with no missing or duplicate entries. Compared to Raw Dataset (Table 4.2), the number of observations has been increased by 840 rows.

Table 4.32: Apple Watch using SMOTE Dataset Statistics

Statistic	Value
Number of variables	17
Number of observations	3774
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	501.4 KiB
Average record size in memory	136.0 B
Variable types	
Numeric	16
Categorical	1

60%/40% partition, $k = 2$ According to the information in Table 4.33, the Extra Trees classifier shows superior performance in FL compared to CL. The remaining models exhibit comparable performance in both scenarios.

Table 4.33: AW Synthetic Dataset 60%/40% partition, $k = 2$

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8289	0.9742	0.8289	0.8270	0.8273	0.7946	0.7949	0.4668	5.9463
(xgboost) tuned model	0.8624	0.9818	0.8624	0.8673	0.8626	0.8347	0.8357	0.3893	0.5948
(xgboost) Aggregated Mean	0.8254	0.9722	0.8254	0.8257	0.8241	0.7897	0.7903	0.5231	0.4881
(xgboost) Aggregated Max	0.8307	0.9739	0.8307	0.8336	0.8299	0.7962	0.7971	0.5244	0.5482
(lightgbm) base model	0.8342	0.9762	0.8342	0.8336	0.8334	0.8010	0.8012	0.4567	20.1700
(lightgbm) tuned model	0.8598	0.9819	0.8598	0.8649	0.8607	0.8316	0.8323	0.4606	1.8038
(lightgbm) Aggregated Mean	0.7937	0.9637	0.7937	0.7928	0.7914	0.7516	0.7523	0.7671	1.5418
(lightgbm) Aggregated Max	0.8095	0.9689	0.8095	0.8034	0.8051	0.7705	0.7711	0.8215	1.6381
(catboost) base model	0.8254	0.9782	0.8254	0.8264	0.8248	0.7903	0.7908	0.4627	23.1546
(catboost) tuned model	0.8704	0.9842	0.8704	0.8741	0.8709	0.8443	0.8449	0.3688	2.6914
(catboost) Aggregated Mean	0.8307	0.9754	0.8307	0.8346	0.8304	0.7961	0.7970	0.4712	2.3578
(catboost) Aggregated Max	0.8360	0.9792	0.8360	0.8453	0.8377	0.8024	0.8036	0.5026	2.3776
(rf) base model	0.8483	0.9782	0.8483	0.8483	0.8477	0.8179	0.8181	0.5906	1.6789
(rf) tuned model	0.8307	0.9789	0.8307	0.8322	0.8304	0.7966	0.7971	0.5526	0.0024
(rf) Aggregated Mean	0.8175	0.9740	0.8175	0.8204	0.8163	0.7807	0.7817	0.5701	0.0007
(rf) Aggregated Max	0.8201	0.9773	0.8201	0.8253	0.8215	0.7837	0.7841	0.6007	0.0007
(et) base model	0.8607	0.9823	0.8607	0.8606	0.8605	0.8327	0.8328	0.4681	0.7785
(et) tuned model	0.8333	0.9751	0.8333	0.8389	0.8352	0.7999	0.8003	0.5320	0.0007
(et) Aggregated Mean	0.8572	0.9759	0.8572	0.8610	0.8569	0.8283	0.8291	0.5197	0.0007
(et) Aggregated Max	0.8783	0.9772	0.8783	0.8806	0.8789	0.8536	0.8538	0.5377	0.0007

60%/40% partition, k = 8 In Table 4.34, it is evident that CL remains superior to all other models. The outcomes in FL remain consistent with the previous trial, as Extra Trees and Random Forest continue to exhibit the strongest performance. This can be explained by the fact that data was divided using KFold, which makes the data not independent and identically distributed (iid), preventing models from learning from the entire dataset. All the results can be located at the exact url mentioned in section 4.3 "Analysis of Real Data".

Table 4.34: AW Synthetic Dataset 60%/40% partition, k = 8

Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	Log Loss	TT (Sec)
(xgboost) base model	0.8289	0.9742	0.8289	0.8270	0.8273	0.7946	0.7949	0.4668	5.4454
(xgboost) tuned model	0.8624	0.9818	0.8624	0.8673	0.8626	0.8347	0.8357	0.3893	0.7104
(xgboost) Aggregated Mean	0.7786	0.9513	0.7786	0.7996	0.7779	0.7323	0.7364	0.7077	0.2230
(xgboost) Aggregated Max	0.8333	0.9606	0.8333	0.8510	0.8303	0.7991	0.8033	0.7834	0.3791
(lightgbm) base model	0.8342	0.9762	0.8342	0.8336	0.8334	0.8010	0.8012	0.4567	21.1002
(lightgbm) tuned model	0.8598	0.9819	0.8598	0.8649	0.8607	0.8316	0.8323	0.4606	2.2908
(lightgbm) Aggregated Mean	0.7266	0.9305	0.7266	0.7377	0.7238	0.6693	0.6722	1.0746	0.4820
(lightgbm) Aggregated Max	0.8125	0.9503	0.8125	0.8289	0.8125	0.7729	0.7762	1.2992	0.8380
(catboost) base model	0.8254	0.9782	0.8254	0.8264	0.8248	0.7903	0.7908	0.4627	25.6365
(catboost) tuned model	0.8704	0.9842	0.8704	0.8741	0.8709	0.8443	0.8449	0.3688	2.6175
(catboost) Aggregated Mean	0.7995	0.9478	0.7995	0.8225	0.7985	0.7574	0.7618	0.6819	1.7437
(catboost) Aggregated Max	0.8750	0.9671	0.8750	0.9050	0.8682	0.8490	0.8571	0.8800	1.9033
(rf) base model	0.8483	0.9782	0.8483	0.8483	0.8477	0.8179	0.8181	0.5906	1.0942
(rf) tuned model	0.8307	0.9789	0.8307	0.8322	0.8304	0.7966	0.7971	0.5526	0.0018
(rf) Aggregated Mean	0.8255	0.9717	0.8255	0.8475	0.8235	0.7897	0.7947	0.5950	0.0007
(rf) Aggregated Max	0.8750	0.9806	0.8750	0.8961	0.8754	0.8497	0.8542	0.6637	0.0008
(et) base model	0.8607	0.9823	0.8607	0.8606	0.8605	0.8327	0.8328	0.4681	0.7343
(et) tuned model	0.8333	0.9751	0.8333	0.8389	0.8352	0.7999	0.8003	0.5320	0.0007
(et) Aggregated Mean	0.8281	0.9760	0.8281	0.8447	0.8259	0.7929	0.7964	0.5319	0.0007
(et) Aggregated Max	0.8542	0.9861	0.8542	0.8782	0.8503	0.8248	0.8305	0.5995	0.0007

4.4.2 Regression

For the same reason as has been said in Classification, the data from Mi Band is real and thus might be non-iid. This time partitioning has been turned off as well. Unfortunately, SMOTE cannot be used in case of Regression problems, as the target column is not expressed by class. Thus, other oversampling techniques have been tested. Among them, Adaptive Synthetic Sampling (ADASYN) has shown the best fit for real data.

ADASYN is an algorithm utilized to mitigate the issue of imbalanced class dis-

tributions in machine learning. It works by adaptively generating synthetic samples primarily for the minority class, focusing more on those samples that are harder to learn. Unlike simple oversampling techniques that may replicate minority class data, ADASYN improves upon this by creating new, synthetic data points that are similar but not identical to existing ones. The number of synthetic samples generated for each minority class example is weighted based on the learning difficulty of that example, promoting better model generalization.

The adaptive nature of ADASYN makes it especially useful in scenarios where some minority class examples are more difficult to classify than others. By generating more synthetic data for these challenging examples, ADASYN helps to achieve a more balanced dataset that can contribute to better performance in classification tasks.

The proposed solution utilizes ADASYN implementation from *ImbalancedLearningRegression* library [61]. It has been applied to the same Regression dataset (particularly the train set) that has been discussed before. Figure 4-6.

Figure 4-6: Data Quality of Regression dataset using ADASYN.

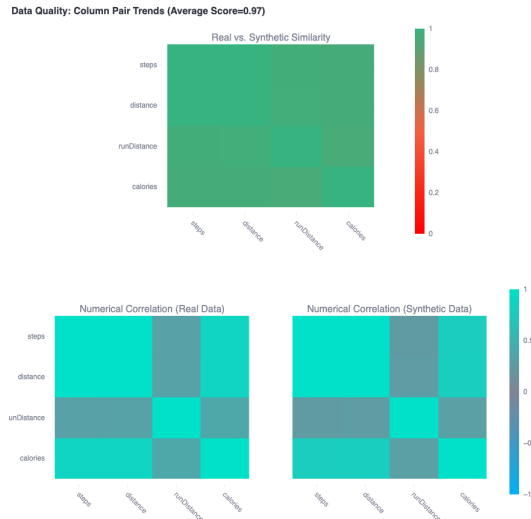
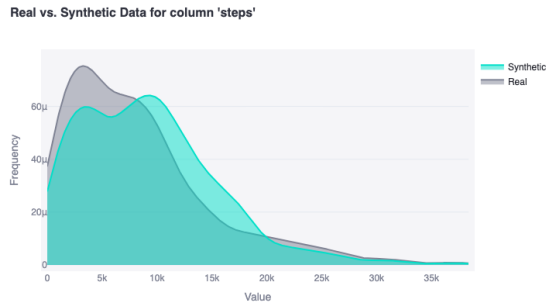


Figure 4-7: Regression: ADASYN vs Real data distribution.



As can be seen in Table 4.35 the number of observations increased by 1420 rows.

Table 4.35: MiBand Dataset using ADASYN Statistics

Statistic	Value
Number of variables	4
Number of observations	2630
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.2%
Total size in memory	82.3 KiB
Average record size in memory	40.1 B
Variable types	
Numeric	4

60%/40% partition, $k = 2$ Table 4.36 shows that Extreme Gradient Boosting and Random Forest are compatible with CL versions, while Extra Trees outperforms CL when trained with FL.

Table 4.36: MiBand Synthetic Dataset 60%/40% partition, $k = 2$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	
(xgboost) base model	48.0747	9468.5647	97.3066	0.7712	0.3020	0.1988	5.5779
(xgboost) tuned model	61.0546	12514.5606	111.8685	0.6926	0.4402	0.2505	0.0859
(xgboost) Aggregated Mean	61.2773	12872.8869	113.4315	0.6851	0.3804	0.2709	0.0738
(xgboost) Aggregated Max	66.5588	13437.1169	115.9186	0.7383	0.4001	0.2779	0.0774
(lightgbm) base model	48.2992	8393.8192	91.6178	0.7971	0.3735	0.2094	3.1823
(lightgbm) tuned model	51.9401	8498.7541	92.1887	0.7913	0.3748	0.2293	0.2269
(lightgbm) Aggregated Mean	54.9092	9344.4710	96.0003	0.7660	0.3740	0.2253	0.1133
(lightgbm) Aggregated Max	61.5715	11520.1998	107.3322	0.8476	0.4159	0.2273	0.1140
(catboost) base model	43.7343	8321.4017	91.2217	0.7989	0.3471	0.1750	2.6928
(catboost) tuned model	47.1213	8880.4177	94.2360	0.7819	0.3184	0.1725	0.4749
(catboost) Aggregated Mean	56.4724	18232.8563	132.6429	0.5359	0.3591	0.2162	0.4511
(catboost) Aggregated Max	65.3370	24937.4828	157.9161	0.7549	0.4136	0.2220	0.4700
(rf) base model	44.8732	9544.0494	97.6937	0.7693	0.2716	0.1790	0.6822
(rf) tuned model	51.3832	10650.0131	103.1989	0.7384	0.3277	0.2004	0.0013
(rf) Aggregated Mean	51.8509	11292.5294	105.7146	0.7183	0.3161	0.1921	0.0006
(rf) Aggregated Max	60.1059	13579.1012	116.5294	0.8085	0.3481	0.1959	0.0006
(et) base model	48.5508	10797.5340	103.9112	0.7390	0.2901	0.1901	0.5171
(et) tuned model	55.2975	11193.7159	105.8004	0.7251	0.3477	0.2249	0.0009
(et) Aggregated Mean	48.5280	9822.5582	97.7883	0.7514	0.2863	0.1739	0.0006
(et) Aggregated Max	58.0632	12976.1965	113.9131	0.8582	0.3428	0.2013	0.0006

60%/40% partition, $k = 8$ The results in Table 4.37 indicate that Random Forest and Extra Trees are better than CL. Shortly, the results of synthetic data do not differ

much from using real data. The final results can be found at the source mentioned in 4.3.

Table 4.37: MiBand Synthetic Dataset 60%/40% partition, $k = 8$

Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	
(xgboost) base model	48.0747	9468.5647	97.3066	0.7712	0.3020	0.1988	0.8351
(xgboost) tuned model	61.0546	12514.5606	111.8685	0.6926	0.4402	0.2505	0.0779
(xgboost) Aggregated Mean	65.8528	14292.4846	116.7600	0.6813	0.4616	0.2936	0.0587
(xgboost) Aggregated Max	91.1468	26555.1031	162.9574	0.8966	0.8915	0.3447	0.1103
(lightgbm) base model	48.2992	8393.8192	91.6178	0.7971	0.3735	0.2094	3.1343
(lightgbm) tuned model	51.9401	8498.7541	92.1887	0.7913	0.3748	0.2293	0.2649
(lightgbm) Aggregated Mean	58.0431	10409.9942	98.2643	0.7832	0.4323	0.3915	0.0405
(lightgbm) Aggregated Max	72.6083	23022.0507	151.7302	0.9037	1.1222	1.3737	0.0573
(catboost) base model	43.7343	8321.4017	91.2217	0.7989	0.3471	0.1750	2.4009
(catboost) tuned model	47.1213	8880.4177	94.2360	0.7819	0.3184	0.1725	0.5104
(catboost) Aggregated Mean	49.5009	10216.8653	95.9313	0.7838	0.3676	0.2178	0.2708
(catboost) Aggregated Max	75.8130	25201.1507	158.7487	0.9596	0.6177	0.4349	0.3999
(rf) base model	44.8732	9544.0494	97.6937	0.7693	0.2716	0.1790	0.7421
(rf) tuned model	51.3832	10650.0131	103.1989	0.7384	0.3277	0.2004	0.0015
(rf) Aggregated Mean	48.9095	10882.7035	98.8223	0.7783	0.2673	0.1707	0.0007
(rf) Aggregated Max	78.8160	26219.4580	161.9242	0.9072	0.4080	0.2370	0.0008
(et) base model	48.5508	10797.5340	103.9112	0.7390	0.2901	0.1901	0.5276
(et) tuned model	55.2975	11193.7159	105.8004	0.7251	0.3477	0.2249	0.0007
(et) Aggregated Mean	51.9628	11277.6568	99.5822	0.7653	0.2624	0.1758	0.0007
(et) Aggregated Max	81.8104	27237.7890	165.0387	0.9409	0.3929	0.2209	0.0010

4.5 Discussion

In this part, the findings of the experiments will be analyzed and the objectives will be addressed.

Reach data interoperability via FHIR protocol. A solution has been architected using the HAPI FHIR server in alignment with FHIR standards for facilitating uniform data management across health IT infrastructures. Systematic data interactions have been enabled by the deployment of a FHIRPy client, ensuring that a secure and efficient data exchange protocol is maintained, thus enhancing the integration and utility of health information systems.

Compare the use of FL and CL performances in analyzing interoperable data. A technical analysis has been performed to assess the differences in perfor-

mance between Federated Learning (FL) and Centralized Learning (CL) when processing interoperable medical data sets. The distributed approach of FL, which prioritizes data privacy, is compared to the centralized data aggregation model inherent in CL, with a focus on their respective impacts on data security and systemic scalability. Such comparisons are indispensable for decision-making by those responsible for the stewardship of health data and the design of analytics systems.

Results from AW dataset were optimal when using a 60%/40% partition and setting $k = 2$. All models were either compatible or exceeded expectations, both in terms of accuracy and training time (TT). This can be attributed to the fact that 40% of `partition 2` can be evenly spread among 4 clients and distributions for testing are alike. In an experiment where the data was split 70%/30%, $k = 4$ performed the best, but for a split of 80%/20%, both $k = 2$ and $k = 8$ were effective. In the end, the results of the 90%/10% experiment are similar, possibly due to the limited size of the testing set.

In the HARTH dataset, the best results were observed with $k = 2$ in experiments with the ratios 60%/40% and 70%/30%, indicating that the data is well preprocessed and follows an independent and identically distributed pattern. Nevertheless, in trials with ratios of 80%/20% and 90%/10%, the outcomes reached a plateau and deteriorated, possibly due to the model developing overfitting.

In the MiBand dataset, the best results were consistently achieved when $k = 2$ during all experiments. It is interesting that as the `partition 1` ratio increases, the combined results begin to even out and yield similar outcomes as the optimized models in CL.

The results from the DAT263x dataset have been consistently positive, making it difficult to assess based on MAE since the differences are minimal. However, it appears that only Random Forest and Extra Trees have performed well in these experiments across different distribution settings.

Random Forest and Extra Trees are commonly seen together and consistently produce strong outcomes. The reason for this can be attributed to the fact that both Random Forest and Extra Trees are combinations of Decision Trees.

Evaluation was also conducted on synthetic data produced from AW and MiBand datasets. The results were discovered to be similar to actual data, although slightly less effective. The reduced effectiveness could be because synthetic data is unable to capture the complexity found in real data, potentially resulting in overfitting or generalization problems.

Design and develop a user-friendly application utilizing AutoML that trains ML models on interoperable data in FL and CL manner. An application characterized by its user-friendliness has been designed and developed, integrated with AutoML technologies, to streamline the training process of machine learning models on interoperable data sets following both FL and CL methodologies. It is intended that users with limited technical expertise in machine learning will be facilitated in creating, refining, and evaluating models through an easily accessible platform. Thus, a sophisticated yet user-centric training environment has been provided, catering to diverse model training and analysis requirements.

Table 4.38: Comparison between Federated and Centralized Learning

Feature/Aspect	Federated Learning	Centralized Learning
Data Privacy	✓	
Reduced Data Movement	✓	
Model Personalization	✓	
Scalability to Large Number of Users	✓	
Data Repository is Decentralized	✓	
Distributed and parallel computation	✓	
Bandwidth optimization	✓	
Easier to Implement and Manage		✓
Consistent Data Quality		✓
Simpler Data Access Patterns		✓
Consistent Data Aggregation		✓
Streamlined Model Training Process		✓

Overall, it can be said that FL has its own advantages over CL. This study showed that FL not only equals to CL, but also outperforms it in some cases. Table 4.38 compares Federated Learning (FL) and Centralized Learning (CL) across various features and aspects, highlighting the strengths of each approach. FL excels in preserving data privacy as it avoids central data collection, supports reduced data movement by training models locally, and offers model personalization based on local data characteristics. It is also highly scalable to a large number of users since the learning process is distributed across multiple nodes. Moreover, FL promotes decentralized data repositories and utilizes distributed and parallel computation, which can optimize bandwidth usage by transmitting only model updates instead of raw data.

On the other hand, CL tends to be easier to implement and manage due to its centralized nature. It generally ensures consistent data quality, benefits from simpler access patterns to the centralized data repository, and provides uniform data distribution, which can lead to more stable model training. Additionally, CL can offer higher model training efficiency due to the centralized processing power and streamlined data handling.

Chapter 5

Conclusion

The digital era has notably changed healthcare, but the plentiful data from IoT devices is not used to its full potential due to constraints in analyzing and sharing the data. This study offers a method to allow different data sources, especially data from wearables collected by health enthusiasts, to communicate with each other using the FHIR standard. This interoperable data is critical for personal health tracking and improvement. But just making data compatible is not enough – we need Machine Learning (ML) to truly harness it. This project is centered on using supervised learning methods for predictive analysis. Since health data is sensitive and personal, it is also crucial to maintain privacy. Federated Learning (FL) looks promising because it lets different data sources collectively train ML models without revealing personal data.

The research aims to develop a web-based tool to analyze data from health wearables and sensors securely and effectively. Here are the main goals of this project:

- To make data from different sources work together using FHIR, agents have been implemented for different types of data. They push transformed data to a HAPI FHIR server that stores and serves it.
- To evaluate FL against traditional centralized approaches (CL) in processing this interoperable data, the research uses known data sets to predict physical activities, a typical classification task, and calculates energy expenditure from

activity data in regression tasks. It evaluates classifications using measures like Accuracy, AUC, Recall, Precision F1, Kappa, MCC, and Log Loss, while regression uses MAE, MSE, RMSE, R2, RMSLE, and MAPE.

- To create an easy-to-use app with AutoML for ML training on interoperable data with both FL and CL, the project upholds open-source principles and modular design. For the user interface, it uses Streamlit, a free open-source framework ideal for quickly making and sharing ML and data science apps. The app applies PyCaret, an open-source ML library for Python that simplifies the ML process, for centralized model training and as a library for FL using Flower, which is a toolkit for building FL systems.

This study found that models trained via FL exceeded the performance of those trained using traditional CL, demonstrating a notable accuracy advantage in classification tasks and comparable efficacy in regression. Contrary to the common belief that FL compromises model performance for the sake of privacy, these findings illustrate robust model outputs, characterized by lower errors and improved accuracy values for FL when compared to CL. Such enhanced performance can be attributed to FL's ability to leverage distributed data effectively, despite potential complexity in communication and integration of updates from other models. The choice to utilize FL's capabilities does not merely prioritize privacy but also embraces improved accuracy which could be vital across various applications. The strong performance of FL models could be predicated on the availability of extensive datasets for model training as well as comprehensive training processes. In contexts where data confidentiality is vital, FL has proven to be not only the more secure route but also the path to superior model accuracy over CL.

In conclusion, the thesis demonstrates that FL holds the potential for handling data from wearable devices and providing insights for future healthcare research involving FL. The newly created and evaluated web application proved effective in performing predictive analysis with centralized data, thereby enhancing the understanding of this technological approach.

5.1 Future Works

This subsection suggests potential improvements and further areas of work based on the findings of the current work. It highlights several approaches to refine the solution proposed in the research.

- **Tackling deserialization risks**

Using `Pickle` in Python can introduce security risks, as it might allow harmful code to run when loading objects back into memory. While this might not be an immediate concern since the system only uses trusted models, the risk is still present. To reduce this risk, other safer methods like `skops` could be used for saving and loading objects. The study provides a reference [62] for more information on safe model storage.

- **Securing communication in Flower**

Security measures are critical, especially when working with sensitive data. One method is to encrypt data transmission with SSL [63]. Protecting user information in areas like health and finance is crucial, and techniques like Differential Privacy offer a way to analyze data without violating privacy [64]. The Flower framework also looks at different secure ways to combine data from multiple sources, called Secure Aggregation Protocols, with examples being SecAgg, SecAgg+, and LightSecAgg [65].

- **Automating agents**

In the current setup, agents run when requested. Yet, it is possible to schedule them to run automatically. A simple tool for this is 'cron', which is a scheduler in Unix systems. Other free and open-source schedulers include Openjob, Chronos, and Dkron, and there are also automation services like IFTTT, Zapier, and n8n that can help automate tasks.

- **Adopting various strategies in federated learning**

FL workflows can be tweaked with different strategies. This project has so far used an average-based strategy, but there are many other ways to combine model parameters. Researching different strategies, like Vertical Learning where clients hold different types of data, might be an interesting direction.

- **Incorporating additional data sources**

When considering Vertical Learning, it is also feasible to add more diverse data sources to wearable device data. This could include integrating data from weather reports, magnetic field variations, dietary apps, or smart home devices that monitor environmental conditions. There is significant potential to discover new patterns by combining various types of data.

Bibliography

- [1] M. Aledhari, R. Razzak, B. Qolomany, A. Al-Fuqaha, and F. Saeed, “Biomedical IoT: Enabling Technologies, Architectural Elements, Challenges, and Future Directions,” vol. 10, pp. 31 306–31 339.
- [2] F. Famá, J. N. Faria, and D. Portugal, “An IoT-based interoperable architecture for wireless biomonitoring of patients with sensor patches,” vol. 19, p. 100547.
- [3] M. Lehne, J. Sass, A. Essenwanger, J. Schepers, and S. Thun, “Why digital medicine depends on interoperability,” vol. 2, no. 1, pp. 1–5.
- [4] N. Gupta and B. Gupta, “Big Data Interoperability in e-Health Systems,” in *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 217–222.
- [5] G. White, Z. Liang, and S. Clarke, “A Quantified-Self Framework for Exploring and Enhancing Personal Productivity,” in *2019 International Conference on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6.
- [6] B. Pardamean, H. Soeparno, A. Budiarto, B. Mahesworo, and J. Baurley, “Quantified Self-Using Consumer Wearable Device: Predicting Physical and Mental Health,” vol. 26, no. 2, pp. 83–92.
- [7] R. Sharma and S. Rani, “Quantified Self: From Self-Learning to Machine Learning,” vol. 23, no. 4, pp. 69–74.
- [8] D. Akdevelioglu, S. Hansen, and A. Venkatesh, “Wearable technologies, brand community and the growth of a transhumanist vision,” vol. 38, no. 5-6, pp. 569–604.
- [9] Z. Liang, “Association Rules Mining on Multimodal Quantified-Self Data,” in *2021 International Seminar on Machine Learning, Optimization, and Data Science (ISMODE)*, pp. 342–346.
- [10] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, “Federated Learning for Healthcare Informatics,” vol. 5, no. 1, pp. 1–19.
- [11] O. Seneviratne, “Enabling Data Interoperability for Decentralized, Smart, and Connected Health Applications,” in *2023 IEEE/ACM Conference on Connected*

Health: Applications, Systems and Engineering Technologies (CHASE), pp. 214–215.

- [12] B. P. Lo, H. Ip, and G.-Z. Yang, “Transforming Health Care: Body Sensor Networks, Wearables, and the Internet of Things,” vol. 7, no. 1, pp. 4–8.
- [13] D. R. Witt, R. A. Kellogg, M. P. Snyder, and J. Dunn, “Windows into human health through wearables data analytics,” vol. 9, pp. 28–46.
- [14] F. John Dian, R. Vahidnia, and A. Rahmati, “Wearables and the Internet of Things (IoT), Applications, Opportunities, and Challenges: A Survey,” vol. 8, pp. 69 200–69 211.
- [15] A. Haleem, M. Javaid, R. Pratap Singh, and R. Suman, “Medical 4.0 technologies for healthcare: Features, capabilities, and applications,” vol. 2, pp. 12–30.
- [16] N. Pathak, A. Mukherjee, and S. Misra, “SemBox: Semantic Interoperability in a Box for Wearable e-Health Devices,” vol. 27, no. 5, pp. 2306–2313.
- [17] S. F. Ahmed, M. S. B. Alam, S. Afrin, S. J. Rafa, N. Rafa, and A. H. Gandomi, “Insights into Internet of Medical Things (IoMT): Data fusion, security issues and potential solutions,” vol. 102, p. 102060.
- [18] H. Fei and M. Ur-Rehman, “A Wearable Health Monitoring System,” in *2020 International Conference on UK-China Emerging Technologies (UCET)*, pp. 1–4.
- [19] p. l. u. family=Casa Pérez, given=Ana, P. Latorre Román, M. Muñoz Jiménez, M. Lucena Zurita, J. A. Laredo Aguilera, J. A. Párraga Montilla, and J. C. Cabrera Linares, “Is the Xiaomi Mi Band 4 an Accuracy Tool for Measuring Health-Related Parameters in Adults and Older People? An Original Validation Study,” vol. 19, no. 3, p. 1593.
- [20] Z. Liang and M. A. Chapa Martell, “Validity of Consumer Activity Wristbands and Wearable EEG for Measuring Overall Sleep Parameters and Sleep Structure in Free-Living Conditions,” vol. 2, no. 1, pp. 152–178.
- [21] M. Swan, “The Quantified Self: Fundamental Disruption in Big Data Science and Biological Discovery,” vol. 1, no. 2, pp. 85–99.
- [22] A. S. Billis, A. Batziakas, and P. D. Bamidis, “Towards a Quantified-Self web application for seniors’ self-tracking,” in *2015 International Conference on Interactive Mobile Communication Technologies and Learning (IMCL)*, pp. 315–317.
- [23] M. Almalki, K. Gray, and F. M. Sanchez, “The use of self-quantification systems for personal health information: Big data management activities and prospects,” vol. 3, no. 1, p. S1.

- [24] T. Sharon, "Self-Tracking for Health and the Quantified Self: Re-Articulating Autonomy, Solidarity, and Authenticity in an Age of Personalized Healthcare," vol. 30, no. 1, pp. 93–121.
- [25] S. P. Erdeniz, A. Menychtas, I. Maglogiannis, A. Felfernig, and T. N. T. Tran, "Recommender systems for IoT enabled quantified-self applications," vol. 11, no. 2, pp. 291–304.
- [26] S. Tyagi, I. S. Rajput, and R. Pandey, "Federated learning: Applications, Security hazards and Defense measures," in *2023 International Conference on Device Intelligence, Computing and Communication Technologies, (DICCT)*, pp. 477–482.
- [27] I. A. Khan, I. Razzak, D. Pi, N. Khan, Y. Hussain, B. Li, and T. Kousar, "Fed-Inforce-Fusion: A federated reinforcement-based fusion model for security and privacy protection of IoMT networks against cyber-attacks," vol. 101, p. 102002.
- [28] M. G. Crowson, D. Moukheiber, A. R. Arévalo, B. D. Lam, S. Mantena, A. Rana, D. Goss, D. W. Bates, and L. A. Celi, "A systematic review of federated learning applications for biomedical data," vol. 1, no. 5, p. e0000033.
- [29] A. Gupta, C. Maurya, K. Dhere, and V. K. Chaurasiya, "Wellness Detection Using Clustered Federated Learning," in *2022 IEEE 6th Conference on Information and Communication Technology (CICT)*, pp. 1–5.
- [30] A. Roehrs, p. u. family=Costa, given=Cristiano André, p. u. family=Rosa Righi, given=Rodrigo, S. J. Rigo, and M. H. Wichman, "Toward a Model for Personal Health Record Interoperability," vol. 23, no. 2, pp. 867–873.
- [31] E. Adel, S. El-Sappagh, S. Barakat, K. S. Kwak, and M. Elmogy, "Semantic Architecture for Interoperability in Distributed Healthcare Systems," vol. 10, pp. 126 161–126 179.
- [32] E. V. Bernstam, J. L. Warner, J. C. Krauss, E. Ambinder, W. S. Rubinstein, G. Komatsoulis, R. S. Miller, and J. L. Chen, "Quantitating and assessing interoperability between electronic health records," vol. 29, no. 5, pp. 753–760.
- [33] V. Daliya and T. Ramesh, "Data Interoperability Enhancement of Electronic Health Record data using a hybrid model," in *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pp. 318–322.
- [34] F. Khalique, S. A. Khan, and I. Nosheen, "A Framework for Public Health Monitoring, Analytics and Research," vol. 7, pp. 101 309–101 326.
- [35] S. Silvestri, A. Esposito, F. Gargiulo, M. Sicuranza, M. Ciampi, and G. De Pietro, "A Big Data Architecture for the Extraction and Analysis of EHR Data," in *2019 IEEE World Congress on Services (SERVICES)*, vol. 2642-939X, pp. 283–288.

- [36] S. Bhartiya, D. Mehrotra, and A. Girdhar, “Issues in Achieving Complete Interoperability while Sharing Electronic Health Records,” vol. 78, pp. 192–198.
- [37] R. S. Evans, “Electronic Health Records: Then, Now, and in the Future,” vol. 25, pp. S48–S61.
- [38] “FOSS Principles Explained Ch.II: Open Source’s Four Freedoms.” [Online]. Available: <https://yairudi.com/foss-principles-explained-ch-ii-open-sources-four-freedoms/>
- [39] “Streamlit - A faster way to build and share data apps.” [Online]. Available: <https://streamlit.io/>
- [40] M. Ali, *PyCaret: An open source, low-code machine learning library in Python*, April 2020, pyCaret version 1.0. [Online]. Available: <https://www.pycaret.org>
- [41] F. Authors, “Flower: a friendly federated learning framework.” [Online]. Available: <https://flower.dev/>
- [42] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, p. u. family=Gusmão, given=Pedro Porto Buarque, and N. D. Lane. Flower: A Friendly Federated Learning Research Framework.
- [43] A. Braungardt. Flower, PySyft & Co. — Federated Learning Frameworks in Python. ELCA IT. [Online]. Available: <https://medium.com/elca-it/flower-pysyft-co-federated-learning-frameworks-in-python-b1a8eda68b0d>
- [44] “FHIR - Open Source Implementations.” [Online]. Available: <https://confluence.hl7.org/display/FHIR/Open+Source+Implementations/>
- [45] “Hapi FHIR: an open-source implementation of the FHIR specification in Java.” [Online]. Available: <https://hapifhir.io/>
- [46] “FhirPy - async/sync FHIR client for python3.” [Online]. Available: <https://github.com/beda-software/fhir-py>
- [47] “Mi Fit and Zepp workout exporter .” [Online]. Available: <https://github.com/rolandsz/Mi-Fit-and-Zepp-workout-exporter>
- [48] “Patient - FHIR v5.0.0.” [Online]. Available: <https://www.hl7.org/fhir/Patient.html>
- [49] “Device - FHIR v5.0.0.” [Online]. Available: <https://www.hl7.org/fhir/Device.html>
- [50] “Observation - FHIR v5.0.0.” [Online]. Available: <https://www.hl7.org/fhir/Observation.html>

- [51] “Physical Activity Measures - FHIR v5.0.0.” [Online]. Available: <https://build.fhir.org/ig/HL7/physical-activity/measures.html>
- [52] “LOINC - The international standard for identifying health measurements, observations, and documents.” [Online]. Available: <https://loinc.org/>
- [53] “Unified Code for Units of Measure (UCUM) is a code system intended to include all units of measures being contemporarily used in international science, engineering, and business.” [Online]. Available: <https://ucum.org/>
- [54] “Flower - What is Federated Learning?” [Online]. Available: <https://flower.ai/docs/framework/tutorial-series-what-is-federated-learning.html>
- [55] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” 2023.
- [56] D. Preuveneers, “AutoFL: Towards AutoML in a Federated Learning Context,” vol. 13, no. 14, p. 8019.
- [57] D. Fuller, “Replication Data for: Using machine learning methods to predict physical activity types with Apple Watch and Fitbit data using indirect calorimetry as the criterion.” 2020. [Online]. Available: <https://doi.org/10.7910/DVN/ZS2Z2J>
- [58] K. A. B. K. B. H. B. Logacjov, Aleksej and P. J. Mork, “HARTH,” UCI Machine Learning Repository, 2023, DOI: <https://doi.org/10.24432/C5NC90>.
- [59] “fmendesdat263xdemos,” <https://www.kaggle.com/datasets/fmendes/fmendesdat263xdemos>.
- [60] S. Chua, C. I. Sii, and P. N. Ellyza Nohuddin, “Comparative Analysis of Machine Learning Models for Fitness Level Prediction with Imbalanced Dataset,” in *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, pp. 102–106.
- [61] W. Wu, N. Kunz, and P. Branco, “Imbalancedlearningregression-a python package to tackle the imbalanced regression problem,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2022, pp. 645–648.
- [62] “Scikit - Model persistence.” [Online]. Available: https://scikit-learn.org/stable/model_persistence.html
- [63] “Flower - Enable SSL connections.” [Online]. Available: <https://flower.ai/docs/framework/how-to-enable-ssl-connections.html>
- [64] “Flower - Differential Privacy.” [Online]. Available: <https://flower.ai/docs/framework/explanation-differential-privacy.html>

- [65] “Flower - Secure Aggregation Protocols.” [Online]. Available: <https://flower.ai/docs/framework/contributor-ref-secure-aggregation-protocols.html>