UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Leyla Rahimli

Model Drift in Federated Learning: An Experimental Analysis

Master's Thesis (30 ECTS)

Supervisor: Feras M. Awaysheh, PhD

Tartu 2024

Model drift in Federated Learning: an experimental analysis

Abstract:

Our ability to extract knowledge beyond data silos will drive the future of machine learning towards more accurate and comprehensive models. Federated Learning (FL) has emerged as a transformative paradigm in machine learning, enabling decentralized model training while preserving data privacy across multiple clients. By distributing the learning process, FL addresses critical privacy concerns but introduces challenges related to model drift. Model drift is a phenomenon where the model degrades over time due to changes in the underlying data distribution or the relationships between input features and target variables. This issue is especially pronounced in FL environments, where data is not independently and identically distributed (non-IID) across clients, leading to asynchronous and heterogeneous updates that intensify drift.

In response to the challenge of model drift in FL, this thesis proposes a novel methodology for drift detection and management within federated environments. By implementing the Flower federated learning framework integrated with Alibi Detect, a specialized tool for drift detection, the study introduces an effective strategy to monitor and identify both concept drift (changes in the relationship between inputs and outputs) and data drift (changes in the input data distribution). The proposed methodology uses statistical tests to accurately detect significant deviations in model performance, ensuring timely intervention and model updates.

Our experimental analysis demonstrates the effectiveness of the proposed drift detection framework. By simulating FL scenarios with varying degrees of drift introduced across different clients, the study systematically evaluates the impact of drift on model performance metrics, including accuracy, F1 score, Cohen's kappa, and ROC. The findings indicate that even minimal drift in a subset of clients can significantly degrade the global model's performance, underscoring the importance of robust drift detection. The proposed solution enhances the reliability and accuracy of federated models and addresses the scalability and privacy-preserving requirements inherent in FL environments.

The contributions of this thesis are significant for the future development and application of FL systems. This study paves the way for more resilient FL models capable of maintaining high performance in dynamic and distributed settings by providing a framework for detecting model drift. The implications of this work extend to various domains where FL is employed, such as healthcare, finance, and personalized services, where the accuracy and reliability of models are critical. This research sets the foundation for future explorations into more advanced drift management techniques, ultimately contributing to FL's broader adoption and efficacy in real-world applications.

Keywords:
Federated Learning, Drift Detection, Model Drift, Concept Drift, Supervised Machine Learning

CERCS: P170 - Computer science, numerical analysis, systems, control; P176 - Artificial Intelligence

Mudeli triiv födereeritud õppes: eksperimentaalne analüüs

Lühikokkuvõte: [1]

Meie võime ammutada teadmisi andmesilodest kaugemale juhib masinõppe tulevikku täpsemate ja põhjalikumate mudelite suunas. Federated Learning (FL) on kujunenud masinõppes transformeerivaks paradigmaks, võimaldades detsentraliseeritud mudelikoolitust, säilitades samal ajal andmete privaatsuse mitme kliendi vahel. Õppeprotsessi levitamisega tegeleb FL kriitiliste privaatsusprobleemidega, kuid tutvustab mudeli triiviga seotud väljakutseid. Mudeli triiv on nähtus, kus mudel laguneb aja jooksul, kuna selle aluseks oleva andmejaotuse muutused või sisendfunktsioonide ja sihtmuutujate vahelised seosed. Eriti väljendub see probleem FL-keskkondades, kus andmeid ei levitata iseseisvalt ja identselt (non-IID) klientide vahel, tuues kaasa asünkroonsed ja heterogeensed uuendused, mis süvendavad triivimist.

Vastuseks mudelitriivi väljakutsele FL-is pakutakse käesolevas lõputöös välja uudne metoodika triivi avastamiseks ja juhtimiseks föderatsioonis. Rakendades triivituvastuse spetsialiseeritud vahendiga Alibi Detect integreeritud Flower õpperaamistikku, tutvustab uuring tõhusat strateegiat nii mõiste triivi jälgimiseks kui ka tuvastamiseks (sisendite ja väljundite vahelise seose muutused) kui ka andmete triivimiseks (sisendandmete jaotuse muutused). Kavandatavas metoodikas kasutatakse statistilisi teste, et täpselt tuvastada olulised kõrvalekalded mudeli toimivuses, tagades õigeaegse sekkumise ja mudeliuuendused.

Meie eksperimentaalne analüüs näitab kavandatava drifti tuvastamise raamistiku tõhusust. Simuleerides FL-stsenaariume erinevate klientide vahel kasutusele võetud erineva driftiastmega, hinnatakse uuringus süstemaatiliselt drifti mõju mudeli jõudluse mõõdikutele, sealhulgas täpsusele, F1-skoorile, Coheni kappale ja ROC-le. Leiud näitavad, et isegi minimaalne triiv klientide hulgas võib oluliselt halvendada ülemaailmse mudeli jõudlust, rõhutades jõulise triivi tuvastamise tähtsust. Kavandatav lahendus suurendab föderatsioonimudelite töökindlust ja täpsust ning käsitleb FL keskkondadele omaseid skaleeritavust ja privaatsust säilitavaid nõudeid.

Käesoleva väitekirja panus on oluline FL-süsteemide edasiseks arendamiseks ja rakendamiseks. See uuring sillutab teed vastupidavamatele FL mudelitele, mis suudavad dünaamilistes ja hajutatud seadetes säilitada suure jõudluse, pakkudes raamistiku mudeli triivi avastamiseks. Selle töö tagajärjed laienevad erinevatele valdkondadele, kus FL töötab, nagu tervishoid, rahandus ja personaliseeritud teenused, kus mudelite täpsus ja usaldusväärsus on kriitilised. See uurimus loob aluse tulevasteks uurimusteks arenenumate driftijuhtimise tehnikate osas, aidates lõppkokkuvõttes kaasa FL-i laiemale vastuvõtmisele ja efektiivsusele reaalsetes rakendustes.

Võtmesõnad:

[1] The abstract was translated with TartuNLP neurotõlge tool [32]

Federeeritud Õppimine, Triivi tuvastamine, Mudeli Triiv, Concepti Triiv, Juhendatud Masinõpe

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria); P176 - Tehisintellekt

Contents

1 Introduction

Machine Learning (ML) has become a cornerstone of modern technology, driving advancements in various fields and making intelligent decisions. The need for distributed ML has emerged as ML models have grown in complexity and application. Distributed ML allows training models across large-scale systems where data is allocated across multiple devices or servers, often spread out geographically. This approach is beneficial for handling vast amounts of data and computational tasks that would be infeasible on a single machine. However, as the volume of data continues to grow, so do the concerns around data privacy. Ensuring that sensitive information remains secure during training has become paramount, especially in healthcare, finance, and other sectors. The increasing focus on data privacy has led to ongoing efforts to collect and utilize data without compromising security. However, this continuous data collection, while necessary for improving model accuracy and adapting to new information, also contributes to the risk of model drift. As new data reflects changes in the underlying patterns that the model has learned, its performance can degrade over time, necessitating regular updates and monitoring to maintain its effectiveness.

The phenomenon where an ML model's performance declines over time is known as model drift, also referred to as model degradation or model shift. This issue arises when the assumptions underlying the model's training phase no longer align with real-world conditions. Model drift is often driven by dynamic environmental changes, including shifts in consumer behavior, fluctuations in economic conditions, the advent of new technologies, and unforeseen events. These factors modify the underlying patterns in data that the model relies on for making predictions, leading to a decline in accuracy and reliability [6, 24]. A fundamental assumption in building ML models is that future data will reflect the patterns observed in historical data. During the training phase, the model identifies and learns these patterns, which are then used to predict outcomes on new, unseen data. However, the relationships between inputs and outputs may change as the environment evolves, creating a divergence between the model's predictions and the actual outcomes. This divergence characterizes model drift, necessitating continual monitoring and updating of models to maintain their effectiveness.

Federated Learning (FL) represents a new trend in distributed and privacy-preserving ML, addressing the growing need to balance data utility with stringent privacy requirements [28]. Unlike traditional centralized ML approaches, FL enables model training directly on users' devices, ensuring that data remains local and is never transferred to a central server. This method not only enhances data privacy but also allows for the inclusion of diverse datasets from a wide range of devices, potentially leading to more robust and generalized models. However, the inherent nature of FL also introduces a greater susceptibility to model drift. Since data is distributed across numerous devices, each with potentially different data distributions, the aggregated model must contend

with various non-IID (independent and identically distributed) [14] data. This heterogeneity can exacerbate model drift, as the model may need help to adapt uniformly across all clients. Especially when the data or environmental conditions change at different rates for participants. Consequently, managing and mitigating model drift in Federated Learning systems is more challenging, requiring innovative strategies to ensure that the model remains accurate and effective despite the training data's decentralized and diverse nature.

FL systems have demonstrated success in environments where all participating clients use the same model type and have similar data, creating a homogeneous training environment. However, in real-world scenarios, the situation is often far more complex. Participants may possess different data types, utilize different model structures, and operate on a wide range of communication and system edge devices. These disparities pose significant challenges to effective collaboration within a FL system [36]. Traditional centralized solutions must be more comprehensive for addressing concept drift in a federated environment, where data is heterogeneous over time and across clients. A single global model may struggle to meet the diverse needs of all clients, particularly when data drift occurs at different times for different clients. Moreover, centralized training approaches are limited in their ability to provide optimal outcomes in situations where multiple, concurrent concept drifts are present [18].

Given the significant challenge of managing model drift in FL environments, there remains a notable gap in the existing literature. While FL has gained attention for its ability to preserve privacy [20] and leverage distributed data, relatively few studies have specifically addressed the issue of model drift within this context. The unique challenges posed by FL—such as data heterogeneity, asynchronous updates, and the decentralized nature of model training—make drift detection and mitigation particularly complex. Despite the importance of this issue, comprehensive solutions for effectively detecting and addressing model drift in FL systems are still underexplored. This study proposes and validates a novel approach for detecting model drift in FL environments to fill this research gap. Our approach aims to enhance the reliability and performance of FL models by identifying drift in real-time, allowing for timely interventions and model updates to maintain accuracy and effectiveness across diverse and evolving datasets.

In this study, we introduce an integrated approach using the Flower federated learning framework alongside Alibi Detect to handle the challenges of model drift in federated learning environments. Our methodology is carefully designed to assess the effectiveness of these tools in detecting model drift. It involves a detailed exploration of various drift detection techniques, adapting them for federated settings where data privacy and non-centralized data management are essential. We developed an experimental framework to test and evaluate this method under real-world conditions and provide a more detailed understanding of its performance and limitations.

The main results from our experiments demonstrate that the adapted drift detection techniques can effectively identify changes in data behavior over time across distributed clients. These results demonstrate the effectiveness of our methodology and highlight the importance of timely and precise drift detection in maintaining the accuracy and reliability of models in federated learning settings.

1.1 Research Motivation

The motivation for this research comes from the growing importance of federated learning (FL) in various applications where data privacy and decentralized data management are critical. Federated learning offers a promising alternative to traditional centralized machine learning approaches by enabling multiple participants to collaboratively train a model while keeping all the data localized, which improves data security and privacy. This approach is relevant in healthcare, finance, and telecommunications, where protecting sensitive information is critical.

However, federated learning introduces challenges related to model drift that are relatively unexplored within the research community. Model drift, or the change in model performance due to the growth of data patterns over time, causes risks to the reliability and accuracy of predictive models in federated networks. The decentralized nature of FL complicates the detection and management of model drift due to the lack of visibility into individual data sources. These restrictions make it more difficult to preserve and guarantee the federated models' long-term reliability and accuracy.

Because of the privacy regulations and the distributed nature of data, the traditional drift detection methods designed for centralized data systems are not applicable to federated environments. In these networks, each client's local data may change differently, which leads to differences in model performance that are hard to detect without a centralized view. Considering these limitations, there is a critical need for techniques that can effectively identify and adapt to concept and data drift. Moreover, understanding how drift impacts federated learning systems contributes significantly to the literature and provides a foundation for future innovations in secure, decentralized machine learning technologies.

This research aims to extend academic knowledge and close these gaps by developing and validating methodologies that can effectively identify and analyze the effects of model drift in federated environments. It focuses on improving the federated learning technology, which is important in real-world areas that need high model accuracy and strong data privacy.

1.2 Objectives of the study

This study's main objective is to handle model drift detection challenges in federated learning environments. The goal of each objective is to understand and implement practical ways to ensure model robustness and reliability across decentralized networks. Here is a detailed explanation of each objective:

- **Defining, analyzing, and classifying drift types in FL**: The first objective involves a clear and detailed understanding of the types of model drift that can occur in federated learning environments. This includes defining each drift type, analyzing the conditions under which drift arises, and classifying them based on their characteristics and impacts. This underlying work is critical for building centered detection and preventative strategies that are specific to the nature of federated learning.

- **Exploring different drift detection techniques in the literature and their applicability for FL**: This objective aims to review and evaluate existing drift detection techniques discussed in the literature, especially those developed for traditional centralized learning environments. The study will discuss the effectiveness and limitations of these techniques in the decentralized and privacy-preserving context of federated learning. The goal is identifying which methods can be effectively adapted or need modification in FL scenarios.

- **Examining existing frameworks for drift detection**: The third goal is to create a reliable framework for identifying model drift in federated learning systems based on the knowledge gained from the previous objectives. This framework will use the most suitable techniques proposed in the literature or used in real-world applications. The key goals of the framework are to provide simplicity, scalability, and effectiveness and to focus on generalized drift detection that can be applied across various federated scenarios

- **Evaluating our proposed solution in the area of FL implementation**: The final objective is to develop and evaluate the drift detection framework in federated learning. This involves testing the framework in a simulated FL environment to assess its applicability and effectiveness in detecting model drift. Evaluation will focus on the framework's ability to accurately detect drift and its impact on the overall performance of the federated model.

By achieving these objectives, the study aims to make significant contributions to the field of federated learning by providing an understanding and robust solutions for managing model drift. This will improve the reliability and accuracy of federated learning models and expand their applicability in various industries and applications where data privacy and decentralized data preservation are critical.

1.3 Research questions or hypotheses

The study will explore several key research questions to address the complexities related to model drift in federated learning environments and test the effectiveness of proposed methodologies. These questions are designed to deepen the understanding of the nature of drift in federated systems and to develop effective strategies for managing these challenges.

- **Q1: What types of drift occur in federated learning environments?**
  This research question aims to describe the various drift types that can impact models in federated learning environments. Drift in this context could include concept drift, where the relationship between input data and the target variable changes over time, and data drift, where changes occur in the input data distribution. Understanding the types and characteristics of drift will help develop a clear framework for addressing the most challenging types in federated settings.
- **Q2: How can drift be effectively detected in federated learning environments?**
  This question focuses on the core of the research problem: identifying effective methods for drift detection in the federated learning environment. It explores various techniques and tools that can be adapted or developed to detect drift suited to the decentralized and privacy-preserving nature of federated learning. The goal is to identify strategies that can robustly signal the presence of drift, which enable timely interventions to maintain or restore model accuracy in the future.
- **Q3: What are the primary challenges in drift detection in federated learning?**
  This question explores the unique challenges of drift detection in federated learning architectures. Critical factors that must be considered include managing privacy requirements, asynchronous model updates, handling non-IID data distributions, and making sure drift detection systems are scalable. Understanding these challenges is essential to creating realistic, practical, and long-lasting solutions for federated learning applications in the real world.

1.4 Structure of the thesis

This thesis is organized into seven sections, each designed to systematically explore model drift detection in federated learning environments.

- **Section 1: Introduction** opens the thesis by outlining the research's background and motivation, defining the research problem, and setting the objectives. It introduces the key research questions and highlights the study's contributions to federated learning and model drift detection. In addition, this chapter offers the reader a roadmap of the thesis structure, guiding the reader through what to expect in the following chapters.

- **Section 2: Theoretical Background** provides the essential theoretical foundations for understanding model drift in machine learning, specifically focusing on federated learning scenarios. It explains the basics of machine learning and details of federated learning, discusses various aspects of model drift, and proves that reliable drift detection techniques are essential in these situations.
- **Section 3: Literature Review** provides existing literature on model drift in federated learning, identifying existing approaches and highlighting significant gaps supporting this study's need.
- **Section 4: Methodology** covers the methods used to detect and analyze model drift. It explains the Flower framework's architecture, the developed drift detection methods, and the reasoning behind the selected methods. This chapter also describes the data collection and preparation methods and the criteria used for evaluating the effectiveness of the drift detection methods.
- **Section 5: Experimental Setup and Results** describes the experimental design and the study results. It discusses the practical implementation of the proposed methodology and evaluates its effectiveness for drift detection in federated learning environments.
- **Section 6: Discussion** provides a deep analysis of the experimental results, discussing them in the context of the research questions and comparing them to existing methods. It explores the importance of the findings for federated machine learning and describes the study's limitations.
- **Section 7: Conclusion and Future Work** concludes the thesis by summarizing the main findings and considering the study's contributions. It reviews the research objectives and questions, explains how they were addressed through the study, and proposes future research directions based on the results and experiences gained.

This structure guarantees a logical path from foundational concepts to in-depth analysis and potential future directions. It provides a detailed understanding of the challenges and solutions of model drift detection in the federated learning environment

2 Theoretical Background

This section provides the foundational understanding for analyzing model drift in federated learning environments. After reviewing the basic machine learning concepts, it describes the concepts of Federated Machine Learning, highlighting its unique challenges, such as data privacy and decentralized data management. The section discusses model drift in FML, including the impact of changing data properties on model performance. It reviews existing model drift detection methods and highlights the importance of effective drift detection in maintaining the reliability and accuracy of models in decentralized networks. This section lays the foundation for a deeper analysis of model drift detection specific to the dynamics of federated learning.

2.1 Basic concepts of Machine learning

Machine Learning is an Artificial Intelligence branch focused on building algorithms and models that help computers perform particular tasks without human intervention. These models are trained using large amounts of data to identify patterns and make decisions and predictions based on learned patterns [17] [22].

Machine learning systems are in demand in areas where many computational operations are performed. These are banking scoring, marketing analytics and statistics, business analysis, selection of investment strategies, and detection of fake news and fraudulent transactions. Moreover, Machine Learning can be used in the medical field to diagnose diseases, predict the effectiveness of treatments, and analyze medical data. The prediction algorithms can help doctors to make diagnoses faster and more accurately, leading to more effective treatment for patients. [22]

There are four basic types of Machine Learning [17] [22]:

- *Supervised Learning*: The model learns from a dataset that includes input features and the corresponding target outputs (labels). The main goal of the model is to identify patterns and relationships between the inputs and outputs to predict the output for new, unseen data based on these learned patterns.
- *Unsupervised Learning*: The model is trained on data that does not include any labels or specific target outputs. Instead, the model looks for hidden structures, patterns, and groups in the data.
- *Semi-supervised Learning*: As the name suggests, this learning method combines supervised and unsupervised learning. It can use a small amount of labeled data to increase the accuracy of model predictions with a larger amount of unlabeled data.
- *Reinforcement Learning*: The model learns to make decisions by receiving true or false signals for actions, learning through trials and errors.

The basic principle of machine learning consists of collecting data from different sources and sending them to a central server. There, the data is cleaned and prepared for further work. After this, the model begins training: it learns from the data, trying to make as few mistakes as possible. Once the model has learned well, it is tested on new data to ensure it works correctly. Finally, the model is run in a natural environment, where it helps solve real problems, such as recommending products or recognizing texts and other purposes.

The main problem with this approach is the unrestricted access to potentially confidential and private information gained from different clients and gathered on the central server, which is addressed in Federated Learning.

2.2 Overview of Federated Machine Learning (FML)

Federated Learning (FL) was first introduced by Google in 2017 [28] to enhance query suggestions on users' mobile device keyboards through machine learning models trained across multiple devices. It is an approach to training machine learning models in which the data remains distributed across users' devices, and the model is trained collaboratively across these devices. This method allows us to improve models without transferring data to the server, which is especially important for ensuring data privacy and security. In simple terms, it is a decentralized Machine Learning method.

Figure 1 shows the architecture of Federated Learning. The key components of FL include clients, the server, local training, and model aggregation. Clients characterize individual data sources that hold local data and participate in training. These could be mobile devices, edge devices, or organizational servers. Each of these devices uses its own data to train the model locally. After local training, devices do not send the data themselves but only updates to the model - for example, the changed weights of the neural network. [21] [22]

These updates are then sent to a central server, where they are aggregated to improve the global model. This process can be repeated several times, with the server distributing the improved model back to the devices for further training iterations. This approach significantly improves privacy because end-user data does not leave their device, and learning occurs without centralized information collection. It could be applied in medical applications, banking, personalized recommendations, and other areas. [21] [22]

Federated learning can be classified into several types based on how data and models are managed and the environment in which they perform. Understanding these types helps design appropriate machine learning strategies and manage specific data privacy, security, and distribution challenges. Figure 2 indicates the primary types of federated learning [35] [26]:

- **Horizontal Federated Learning (HFL)** (Figure 2a) is also known as sample-

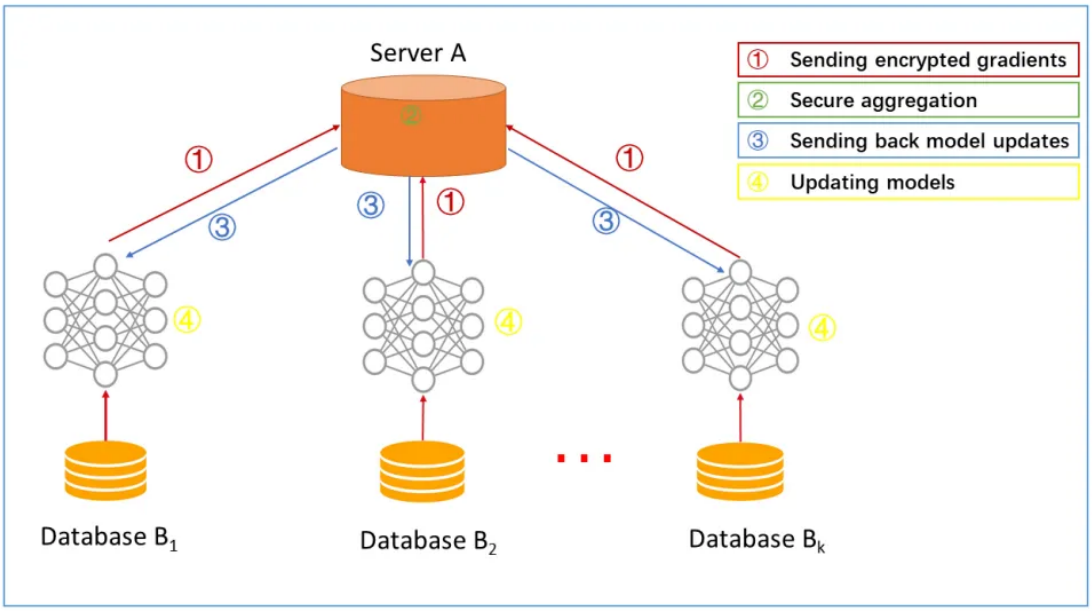
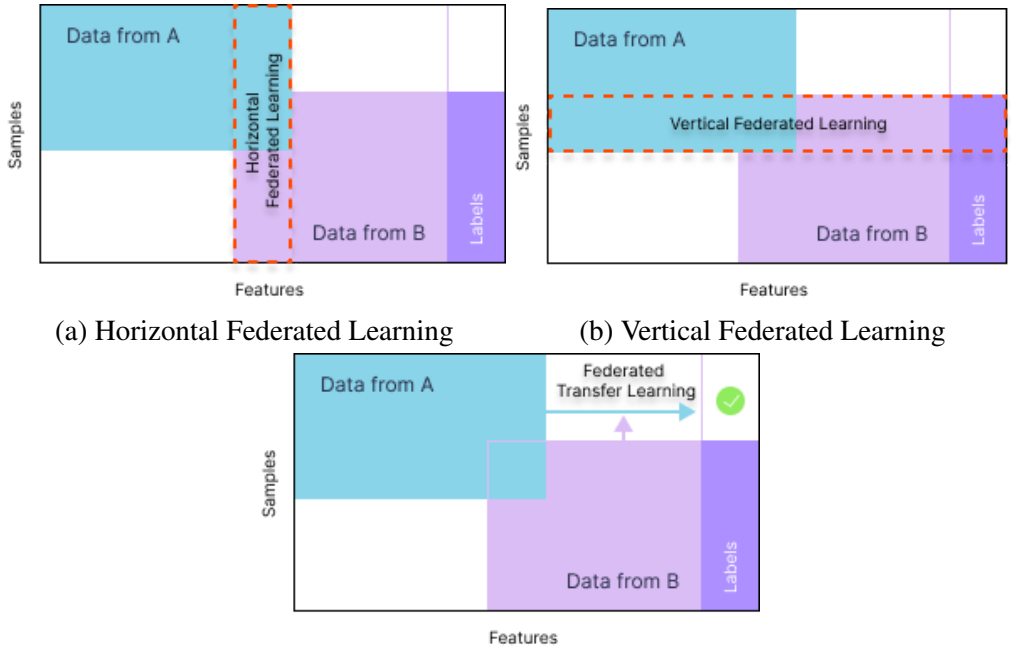


Figure 1. Flower architecture [35].

based federated learning. HFL occurs when different participants (clients) have different samples but share the same feature space. This type of federated learning is most common when participants, like mobile phones or hospitals in the same region, collect data in the same feature space, but the data samples themselves are different.

- **Vertical Federated Learning (VFL)** (Figure 2b) is also known as feature-based federated learning. VFL happens when different participants hold different subsets of features for the same samples. This type is applicable when two entities have different kinds of information on the same set of individuals. For example, it can be used in cases when businesses from various industries have different data features about the same users, like a bank and a retail store with shared customers but specific data types.
- **Federated Transfer Learning (FTL)** (Figure 2c) combines federated learning and transfer learning concepts, enabling model training across domains with different feature spaces. It is useful when there are insufficient overlapping features between different parties. This type is beneficial in different domains where two datasets may share different feature spaces but need to collaborate, such as between a web search engine and a medical research institution.

Moreover, based on the number of participants and training scale, FL can be divided



(a) Horizontal Federated Learning (b) Vertical Federated Learning

(c) Federated Transfer Learning

Figure 2. Federated Learning types

into [26] [16]:

- **Cross-device FL** is a large-scale type of federated learning involving many devices (potentially millions), such as mobile phones or IoT devices [5], contributing to the learning process. The reason for training so many clients is that all these clients have a relatively small amount of local data; in this case, a successful result requires a large number of edge devices to be involved. The cross-device FL is common in applications where millions of users might contribute data for model improvement without directly sharing personal data, such as in improving keyboard predictions or smartphone app recommendations.

- **Cross-silo FL** is used when the participating clients are fewer in number and available for all rounds (from two to a hundred). The training data can be in horizontal or vertical FL format. This type of federated learning is practiced among organizations or silos (like hospitals, banks, or regions) rather than individual devices. It involves fewer participants than cross-device federated learning but with larger, more reliable data contributions.

A traditional federated learning process involves several iterative steps, often called rounds, including local learning and global aggregation. At each round, both the clients

Algorithm 1 FederatedAveraging [28]

Input: K clients 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**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
end for
$m_t \leftarrow \sum_{k \in S_t} n_k$
$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for

ClientUpdate(k, w) // Run on client k
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
for batch $b \in B$ **do**
$w \leftarrow w - \eta \nabla \ell(w; b)$
end for
end for
return w to server

and the server go through two different learning phases: local parameter updating and global parameter aggregating.

There are many algorithms for aggregation; however, the most common aggregation method is Federated Averaging [28].

The FedAvg (Federated Averaging) was introduced by researchers at Google. It is a fundamental technique used in federated learning to aggregate models trained locally on distributed devices (like smartphones or edge devices) into a global model, which improves privacy and scalability.

Algorithm 1 [28] represents the Federated Averaging algorithm which involves the following steps:

- At the beginning, the server initializes the model's learnable parameters w_0, which represent the weights and biases;
- Every round starts with the server choosing a random subset of clients;

- All of these chosen clients receive a message from the server containing the current model parameters or the current state of the global algorithm;
- The model is trained on the local data of each client device. The training can be performed several times, equal to the number of epochs, to increase the model's performance;
- After the training, the clients sent back their updated models with the new weights and biases to the central server;
- The central server aggregates the updated models received from the clients by averaging the model parameters. This averaging method updated the global model while maintaining data privacy. Here, a single round terminates;
- If there are more rounds, then the aggregated global model is sent back to the devices and repeats all steps, starting from training on local data for a specific number of rounds.

2.3 Model Drift & FL

Model Drift [2, 13], in the context of machine learning, describes a situation where the distribution of data based on which the model was initially trained changes over time. As a result, the model's performance may decline because the predictions and patterns identified during training no longer match the new data it encounters in real-world use.

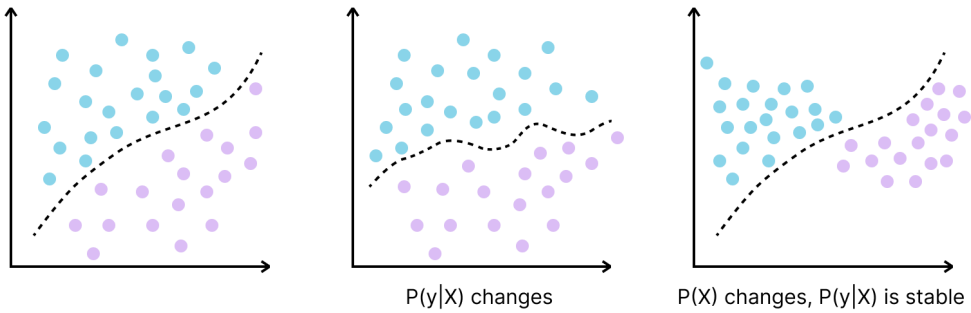
In the context of federated learning, drift also has significant consequences. As federated learning involves multiple devices or clients training a model on their local data, drift can occur differently across devices. For example, user preferences may change over time, or different trends may occur in different regions. This means that a trained and updated model on such a distributed system must consider that data may drift differently across different clients.

Figure 3 represents different types of model drift in federated learning compared to the scenario (Figure 3a) where drift does not occur:

- **Concept Drift**: changes in the target variable's statistical properties or the relationship between the input data and the target output (Figure 3b). In federal learning, concept drift can occur, for example, in changing user preferences across regions or over time. For example, a model trained on user purchase data may give incorrect predictions if user preferences change due to seasonal trends or new products. Mathematically concept drift [13] can be defined as:

  $$P_t(y \mid X) \neq P_{t+1}(y \mid X)$$

  where $P_t(y \mid X)$ is the conditional probability distribution at time t.



(a) Original data (no drift) (b) Concept drift (c) Data drift

Figure 3. Model drift types

- **Data Drift**: a change in input data distribution or features while the relationship between the input data and the target variable remains unchanged (Figure 3c). Data drift [13, 2] can occur in federated learning, for example, if user demographics change. For example, a model trained on user purchase data may begin to make incorrect predictions if a model was initially trained on young users between the ages of 18 and 25 and then starts receiving data from users over 50; the model may not consider the behavior of older users. Mathematically data drift can be presented as:

$$P_t(X) \neq P_{t+1}(X)$$

where $P_t(X)$ is the distribution of input features at time t.

In federated learning, drift is important as it can lead to model performance degradation if not detected and addressed. In practice, the global model needs to be monitored for drift, and strategies like retraining, updating features, or even changing the model architecture might be required to maintain accuracy.

In summary, drift is a key concept in the lifecycle of machine learning models that refers to changes in data over time, potentially impacting model performance. Different types of drift describe different aspects of how these changes occur and what effects they have on the model.

2.4 Existing model drift detection methods

Detecting model drift is an essential aspect of machine learning, especially in scenarios where models make predictions and decisions based on data that may change over time. Model drift can occur in various forms and can significantly impact the accuracy and reliability of the model. Both concept and data drift types can affect model performance if they are not properly managed.

Among the first methods for monitoring model performance are **statistical process control** (SPC) [6] methods. These techniques define the typical operating conditions of a model by defining statistical thresholds using control charts. For instance, methods like the Cumulative Sum (CUSUM) and Exponentially Weighted Moving Average (EWMA) are used to monitor the model's performance metrics, such as accuracy or error rates.

Window-based [6] techniques use sliding or expanding windows to compare current data with a historical baseline. This method is often paired with statistical tests such as the Kolmogorov-Smirnov or Anderson-Darling test to determine any significant differences between the "current" data window and a "reference" window. This approach is adaptable and applicable to classical and continuous machine-learning scenarios. It is especially advantageous for streaming data, where it is essential to evaluate ongoing data features continuously.

Online Learning and Adaptive Methods [24] are designed for settings with continuous data streams. These techniques include incremental learning and adaptive windowing [6], where the model or its parameters are updated in real-time to adapt to new data. The methods mentioned help keep the machine learning model up-to-date as new data comes in. This is crucial for applications where the model needs to continually learn and adjust to new patterns in the data quickly and smoothly.

With **ensemble techniques** [6], drift detection becomes more reliable by combining the strengths of multiple models or versions of a model. This approach may include methods such as Model Rebalancing [4], in which models are frequently adjusted based on their performance over time, or Weighted Average, in which many models contribute to the final decision based on their accuracy. Ensemble techniques are effective in both classical and continuous learning scenarios, especially in complex systems where multiple types of drift might co-occur.

Adapting these methods in federated learning brings difficulties due to the decentralized nature and data privacy concerns, complicating the process. However, despite the presented challenges by the distributed data environment, this adaptation is essential to guarantee that federated learning models are robust, accurate, and reliable over time.

2.5 Relevance of model drift detection in federated settings

Model drift detection is crucial in Federated Learning (FL) because it directly impacts the accuracy and performance of the global model. In FL, models are trained on data distributed across multiple clients, each with unique and potentially changing data distributions. The model's predictive accuracy may degrade without appropriate drift detection techniques due to changes in the underlying data. This can include data distribution drift or differences in the link between inputs and outputs (concept drift), which may lead to incorrect predictions and decisions. Detecting and addressing these

drifts guarantees that the global model remains robust and performs well even as the data changes over time.

The diverse and non-IID (independent and identically distributed) nature of data across clients makes drift detection in FL particularly challenging and necessary. In a Federated Learning (FL) configuration, clients often hold data that differs significantly in distribution due to differences in user behavior, geographic location, or other contextual factors. Detecting drift allows the system to recognize when data characteristics have shifted. This allows us to make adjustments at the client level or during model aggregation, which ensures that the global model remains fair and effective across all participating clients.

In federated learning, communication between clients and the central server is key to the system's overall efficiency. Communication between clients and the central server is a significant barrier in FL because frequent and large-scale communication of model updates might require many resources and lead to delays and increased costs. In this case, detecting and addressing model drift can reduce the need for frequent updates. For example, if drift is detected early, corrective actions can be implemented locally at the client level or through selective communication, which minimizes the need for global model updates. This approach improves the overall efficiency of the FL process by focusing resources on significant updates that improve model performance while optimizing bandwidth (network capacity) and reducing latency.

3 Literature Review

The literature review explores the concept of model drift in machine learning, focusing on both concept drift and data drift, to understand the current state of research and identify significant gaps, especially within the context of federated learning. Model drift, including concept drift, where the statistical properties of the target variable change over time, and data drift, involving shifts in input data distribution, can significantly degrade the performance of machine learning models.

In classical machine learning, considerable research has been published on understanding and handling model drift. Studies such as those by Gama et al. [13], and Žliobaitė [39] have laid a robust foundation, describing various techniques for managing drift, such as adaptive sliding windows and ensemble methods. These techniques focus on dynamically updating models to adapt to new data patterns as they emerge, effectively maintaining model accuracy over time. These methods have proven effective in centralized data environments where direct access to data allows real-time analysis and immediate model adjustments.

However, adapting these well-established methods to federated learning brings a number of new challenges. Because of their strict privacy requirements and decentralized structure, federated learning environments present unique problems that have not yet been completely explored in the literature. In such settings, the implementation of traditional drift detection techniques is restricted due to the lack of direct access to centralized data. Additionally, the variability in data distribution among the networked clients in a federated system complicates the application and effectiveness of these methods. This unique structure demands novel approaches for drift detection that can operate efficiently within distributed data privacy and management regulations.

Despite the growing research on federated learning by researchers like Kairouz et al. [19] and Li et al. [23], which explore general frameworks and challenges in federated networks, there is still a need for more specific strategies for robust drift detection and management. These studies often focus more on optimizing communication efficiency and training models under privacy constraints rather than investigating and providing model robustness against evolving data conditions, which is crucial for long-term deployment.

The literature thus highlights a significant research gap in drift detection mechanisms designed for and compatible with federated learning systems. Addressing this gap is crucial for theoretical advancements in machine learning and practical applications where federated models are deployed in dynamic and data-sensitive environments.

On the other hand, data skew is a prominent challenge in Federated Learning that significantly impacts model performance, mainly when data is non-IID across clients [38, 9]. Several studies have addressed different aspects of data skew, focusing on label

skew, feature skew, and methods to mitigate these effects to enhance model convergence and accuracy [37].

One primary area of research focused on label skew, where the distribution of labels across clients varies significantly. For instance, the work by Tijani et al. proposes a data extension approach to mitigate the effects of extreme label skew by introducing external data samples as placeholders for missing classes, which enhances model generalization and reduces test error rates in highly non-IID scenarios [33]. Similarly, the study by Zhang et al. focuses on addressing label skew through the introduction of a balanced weight strategy that adjusts the contribution of each client to the global model, aiming to achieve a more representative global model even under severe label skew conditions [34].

Beyond label skew, other studies have explored broader approaches to handle data skew in FL environments. Verma et al. discuss the challenges posed by data skew in tactical coalition environments and propose two mechanisms, bounds-aware fusion, and bounds-expanding data exchange, to improve model accuracy when data is heavily imbalanced or incomplete across different coalition partners [11]. These approaches are particularly relevant when data cannot be exchanged freely due to privacy concerns, regulatory constraints, or one-shot FL [10].

While the previous approaches aim at the uneven distribution of data that can bias a model during its initial training, they do not assure degradation of a model's performance over time as the data distribution changes. This challenge requires different detection and mitigation strategies but is critical in ensuring the robustness and accuracy of FL models in varying and evolving environments. In contrast to the previously discussed literature, our study presents a novel approach by addressing model drift in FL, integrating the Alibi Detect algorithm into FL open-source frameworks. Our contributions highlight the ongoing efforts to address the complexities introduced by data drift in FL. Developing innovative techniques to manage and mitigate the effects of data heterogeneity ensures more robust and equitable model performance across diverse and distributed data environments.

4 Methodology

The Methodology section describes how Flower and Alibi Detect are used in a federated learning context to systematically detect model drift. This section describes the suggested model drift detection method and justifies the selected approach. It describes the data collection and preparation process, guaranteeing that the used data is representative and suitably preprocessed for correct analysis. Finally, the evaluation criteria and metrics used to assess model drift detection are explained, showing how several performance metrics, including accuracy, loss, Cohen's Kappa, F1 score, recall, and ROC AUC, are used to confirm the effectiveness of the proposed method.

4.1 Flower: A Friendly Federated Learning Framework

There are various frameworks for deploying federated learning. Our project uses the Flower (FLWR) framework to create a federated learning environment, allowing decentralized model training across different clients. Flower is an open-source framework for federated learning that makes it easier to train machine learning models without the need for centralized storage. Flower is designed to be highly scalable and efficiently handle federated learning environments ranging from a few devices to thousands of nodes. Flower performs effectively in both simulation and real-world applications and allows the smooth transition of experimental implementations between these two contexts as needed during research and development [7]. Its architecture supports various machine learning frameworks such as TensorFlow, PyTorch, and scikit-learn, which allows developers to implement federated learning with familiar tools and methodologies.

Figure 4 indicates the Flower core framework architecture with Virtual Client Engine. The server side has three primary components: the ClientManager, the FL loop, and a (user-customizable) Strategy. Clients are sampled from the ClientManager, which controls a set of ClientProxy objects, each representing a single client connecting to the server. They transmit and receive Flower Protocol messages to and from the client. At the center of the FL process is the FL loop, which coordinates the whole learning process. However, rather than making decisions, decision-making authority is passed on to the currently chosen Strategy. The Strategy here characterizes the algorithms that aggregate model parameters from multiple clients, such as Federated Averaging (FedAvg). The FL loop, in short, requests that the Strategy set up the upcoming FL round, notifies the clients of these configurations, gets the client updates or failures from the clients, and assigns result aggregation to the Strategy. With the further option of server-side evaluation (again via the Strategy), it follows the same process for both federated training and federated evaluation. The client role is easier; it waits for server instructions and executes training and evaluation methods [7, 27].

In our study, we configured the Flower framework to handle the specific needs of our

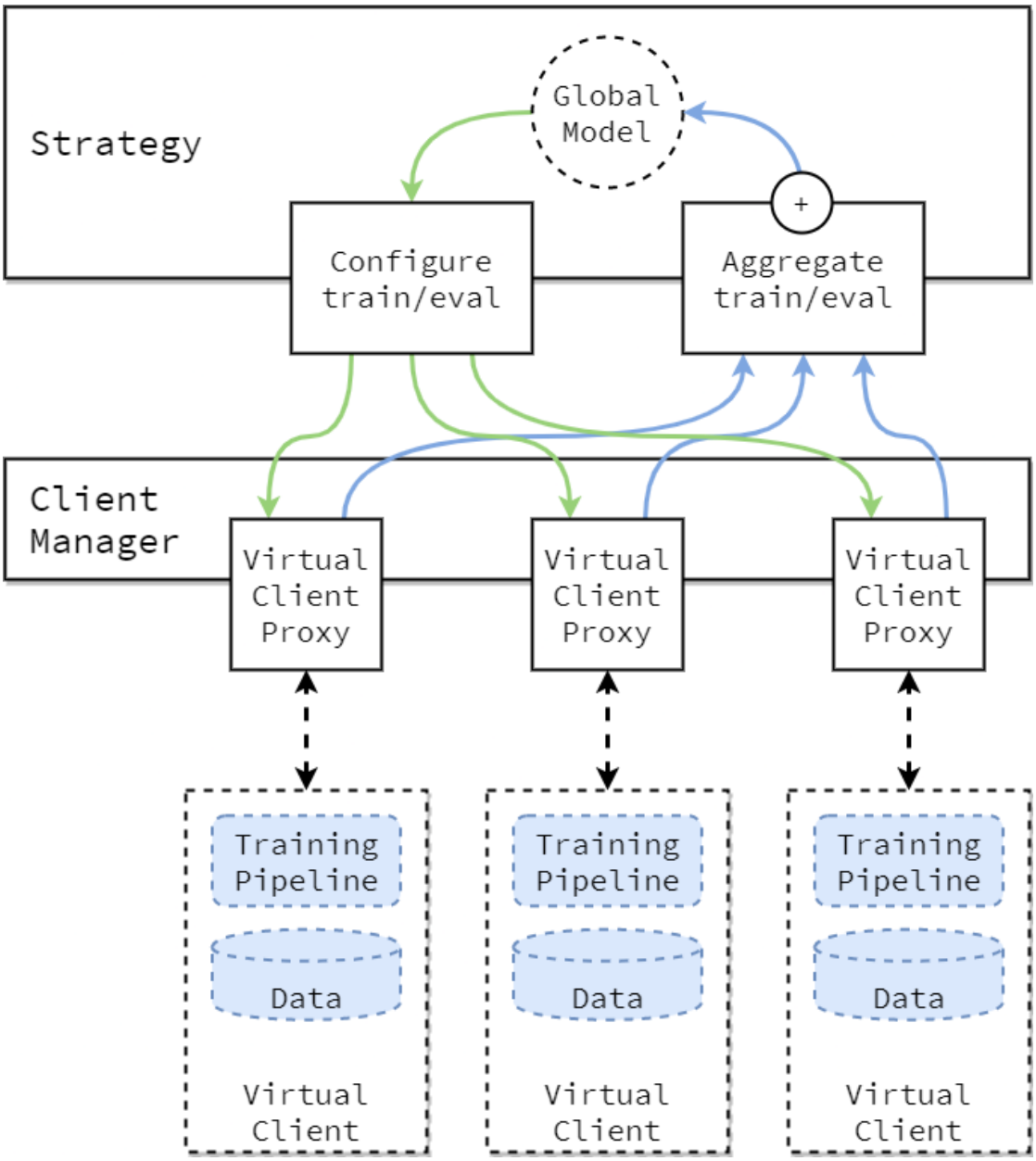


Figure 4. Flower architecture [12].

federated learning experiment, which focused on model drift detection. By using Flower's robust and flexible architecture, we carried out detailed experiments that significantly contributed to our understanding of model drift in federated learning environments, showing Flower's capability to support advanced federated learning research.

4.2 Description of the proposed model drift detection method

The proposed method uses the *Alibi-Detect* [31] library, an open-source Python library designed for detecting outliers, adversarial examples, and drift detection. It can operate with text, images, time series, and tabular data detectors both offline and online. Additionally, for drift detection, Alibi-Detect supports TensorFlow [1], PyTorch [30], and KeOps(where applicable) [8] backends and provides various algorithms for detecting concept and data drift.

As for algorithm for drift detection, we chose the *Fisher's Exact Test* (FET) [3] detector provided by Alibi Detect.

The FET drift detector is a method used to identify changes in data patterns without assuming any specific statistical model (non-parametric detector). It uses Fisher's Exact Test (FET) to analyze each feature individually. This method is specifically designed for binary data, where each feature can have one of two values: True/False or 0/1. It works best in supervised machine learning scenarios, where it tracks changes in the model's accuracy on individual predictions. Here, a correct prediction by the model is marked as 0, and an incorrect prediction is marked as 1. By continuously checking these binary results, the FET drift detector can identify if there is a significant shift in the data that might impact the model's performance. [3]

The detector is mainly meant for univariate data but can also be applied in a multivariate context. For univariate data, the detector uses Fisher's Exact Test to compare the distribution of a feature between a reference dataset and a test dataset for that feature. The 2x2 contingency table for the j^{th} feature is constructed as follows:

	True	False
x_j	N_1	N_0
x_j^{ref}	N_1^{ref}	N_0^{ref}

Where N_1 and N_1^{ref} represent the number of true instances in the test and reference data, respectively, while N_0 and N_0^{ref} represent the number of false instances for the feature j^{th} in these datasets accordingly.

The odds ratio (OR) compares the odds of observing a "1" versus a "0" in the test data to the odds of observing a "1" versus a "0" in the reference data. Mathematically, the odds ratio is defined as:

$$\widehat{OR} = \frac{\frac{N_1}{N_0}}{\frac{N_1^{ref}}{N_0^{ref}}}$$

This formula simplifies to:

$$\widehat{OR} = \frac{N_1 \times N_0^{ref}}{N_0 \times N_1^{ref}}$$

The hypothesis testing determines whether there is a significant change (drift) in the feature's distribution between the test data and the reference data, based on the odds ratio:

- Null Hypothesis (H_0): The null hypothesis states that the odds ratio ($H_0 : \widehat{OR} = 1$) is equal to 1, meaning that the proportion of 1's to 0's in the test data is the same as in the reference data. In other words, there's no drift; the feature's distribution has not changed between the two datasets.
- Alternative Hypotheses (H_a): The alternative hypothesis depends on the direction of the drift testing:
  - Greater Alternative ($H_a : \widehat{OR} > 1$): This hypothesis suggests that the odds of observing a "1" have increased in the test data compared to the reference data. In simpler terms, the feature is more likely to be 1 in the test data than in the reference data.
  - Less Alternative ($H_a : \widehat{OR} < 1$): This hypothesis suggests that the odds of observing a "1" have decreased in the test data compared to the reference data. In other words, the feature is less likely to be 1 in the test data.
  - Two-Sided Alternative ($H_a : \widehat{OR} \neq 1$): This hypothesis suggests that the odds of observing a "1" have changed (increased or decreased) in the test data compared to the reference data. It tests for any difference, not just an increase or decrease.

The p-value is a critical concept in statistical hypothesis testing, representing the probability that the observed data (or something more extreme) could have occurred under the null hypothesis.

In the context of odds ratios and Fisher's Exact Test, the p-value is defined as the probability of observing an odds ratio $\widehat{OR}$ as extreme as (or more extreme than) the one calculated from the observed data, assuming that the null hypothesis (H_0) is true [25]. Mathematically, this can be expressed

$$\text{p-value} = P(\widehat{OR} \geq \widehat{OR}_{\text{obs}} \mid H_0)$$

Here, $\widehat{OR}_{\text{obs}}$ represents the odds ratio calculated from the observed data, and the p-value reflects the likelihood of obtaining this observed odds ratio or one even further from the null hypothesis expectation of ($H_a : \widehat{OR} = 1$), given that the null hypothesis is true. In general, H_0 is rejected if the p-value does not exceed α, the significance level of the FET test.

4.3 Justification of the chosen methodology

After carefully considering tools that would fit in with the goals of this study, we decided to choose the Alibi Detect library for drift detection with Fisher's Exact Test (FET) detector and Flower for federated learning environment simulation.

We chose *Flower* due to its ability to integrate within a federated learning environment. Flower is an effective framework that coordinates the training of models across multiple clients while ensuring that raw data remains decentralized and providing various aggregation algorithms. It is an open-source framework, and its flexible architecture allows us to integrate multiple drift detection methods seamlessly.

We used *Alibi-Detect* to broaden Flower's capabilities to detect drift in model performance. Alibi-Detect offers a variety of algorithms for monitoring changes in data distributions and supports various drift detection methods, making it adaptable to different types of data and drift scenarios. Moreover, it is an open-source library actively maintained and supported by the community. Alibi-detect can be easily integrated with existing machine learning workflows and federated learning setup by Flower, which helps us efficiently monitor changes in data distribution. It allows us to detect and address potential issues in real time, maintaining the model's accuracy over time.

For our drift detection requirements, we have specifically selected *Fisher's Exact Test (FET)* from Alibi-Detect because of its ability to handle categorical data. FET is a well-known, statistically rigorous method that provides exact p-value calculations, making it reliable for detecting drift, even with small sample sizes. This is especially helpful in federated learning environments where data sizes can vary significantly across clients. FET guarantees accurate detection of any shifts affecting model performance by focusing on categorical feature distributions.

Integrating Flower, Alibi Detect, and Fisher's Exact Test provides a robust framework for drift detection in federated learning environments. Flower supports scalable federated learning and provides mechanisms for integrating drift detection algorithms, while Alibi Detect handles various drift patterns efficiently across distributed nodes. The method is further strengthened by Fisher's Exact Test, which offers strict statistical validation—a critical component in maintaining the reliability of drift detection results.

This methodology fits well with the scalability, efficiency, and data privacy requirements of federated learning. It provides precise drift detection while maintaining privacy since it functions across decentralized nodes without centralized data collecting.

In summary, the chosen method effectively satisfies the need for a flexible, scalable, and statistically robust drift detection system that is compatible with the dynamic nature of federated learning environments.

4.4 Data collection and preparation process

We used the palmerpenguins dataset [15] for the experiment.

The initial dataset includes information on 344 penguins from three Antarctic islands in the Palmer Archipelago. In this dataset, there are three different species of penguins: Adelie, Gentoo, and Chinstrap. There are 8 features that describe the physical features of the penguins, their species, sex, the island they live on, and the year the measurements were made. Table 1 represents some samples from the dataset.

Table 1. Original palmerpenguins dataset

species	island	bill length mm	bill depth mm	flipper length mm	body mass g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Biscoe	39.0	17.5	186.0	3550.0	female	2008
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Gentoo	Biscoe	46.7	15.3	219.0	5200.0	male	2007
Gentoo	Biscoe	43.3	13.4	209.0	4400.0	female	2007
Chinstrap	Dream	53.5	19.9	205.0	4500.0	male	2008
Chinstrap	Dream	49.0	19.5	210.0	3950.0	male	2008

The task is to classify each species according to its length and depth of bill, so we remove unused features in the process. Table 2 represents some samples from the modified dataset.

Figure 5 helps with species classification by visually representing the variations in bill measurements between the three species of penguins. From the plot, we can see that the Adelie species can be identified by their bill length. Further, to distinguish Gentoo and Chinstrap species, we need to analyze their bill depth.

Our data required more samples to distribute across 10 clients. Hence, we augmented the dataset by adding 1,500 samples for each species. This augmentation is based on patterns identified for each species, considering the length and depth of their bills.

Table 2. modified palmerpenguins dataset

species	island	bill_length_mm	bill_depth_mm
Adelie	Torgersen	39.1	18.7
Adelie	Biscoe	39.0	17.5
Adelie	Torgersen	40.3	18.0
Gentoo	Biscoe	46.7	15.3
Gentoo	Biscoe	43.3	13.4
Chinstrap	Dream	53.5	19.9
Chinstrap	Dream	49.0	19.5

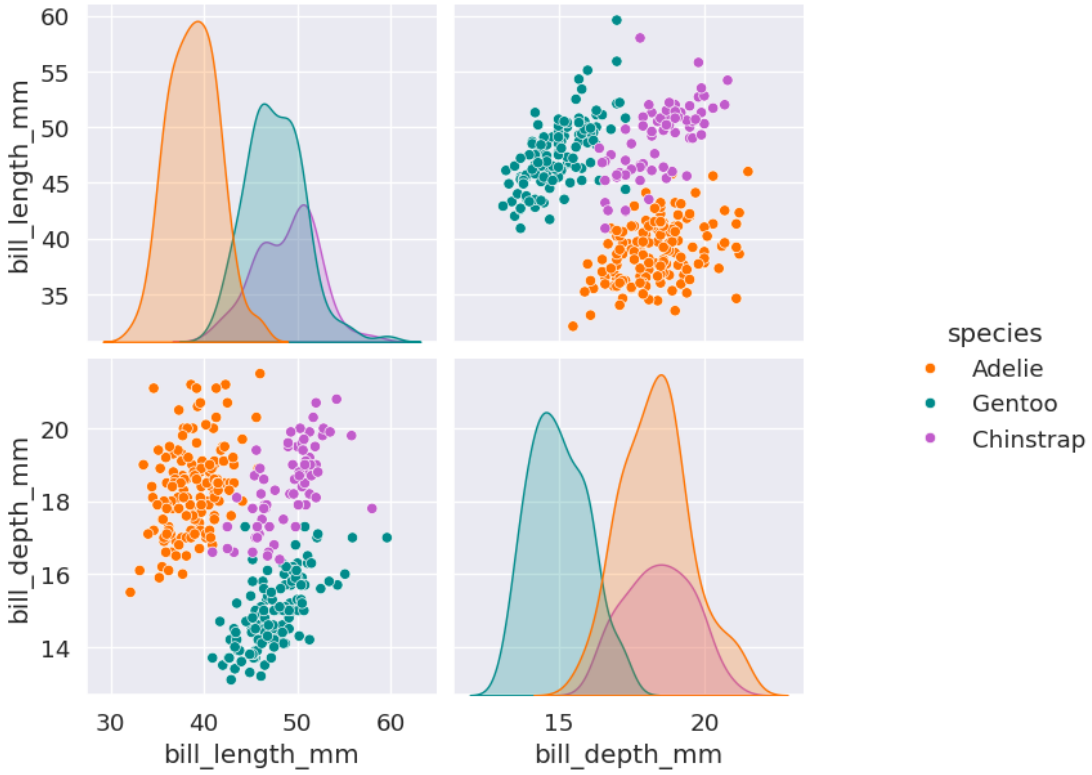


Figure 5. Classification plot of initial dataset

In the end, we have a dataset with 4,832 examples, which was subsequently shuffled to distribute properly between clients. Figure 6 represents the species classification in the generated synthetic dataset. As we can see, the plot represents the same distinguishing characteristics for the species as in the initial dataset.

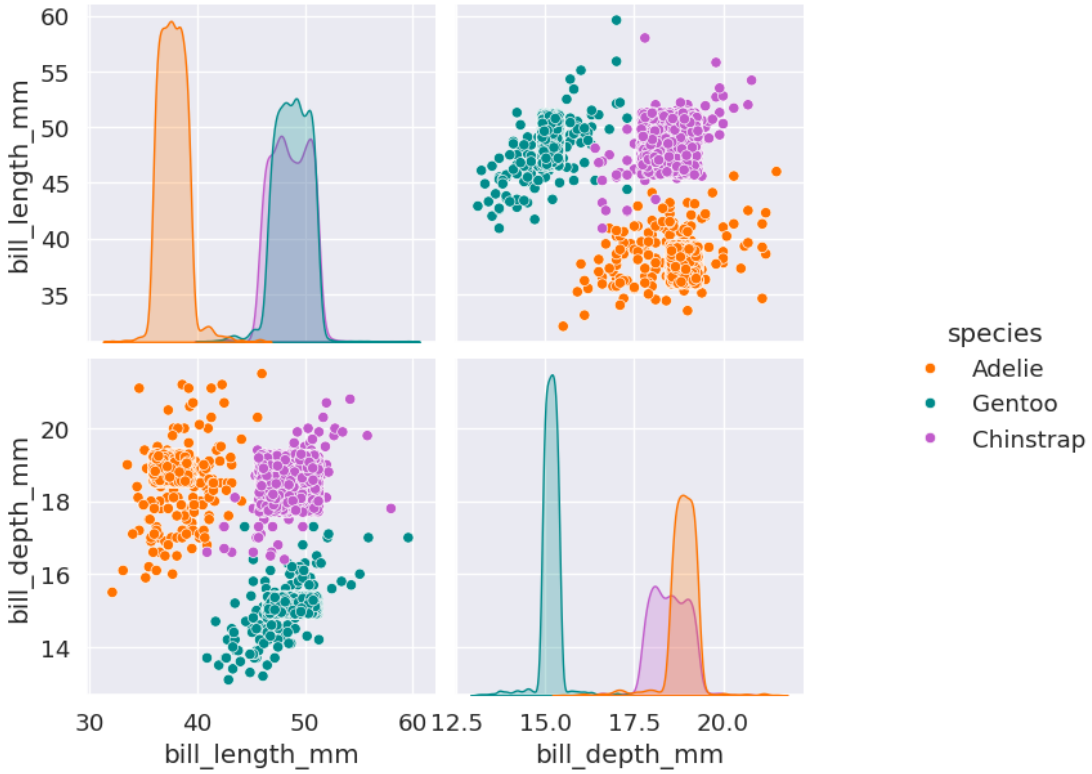


Figure 6. Classification plot of generated synthetic dataset

4.5 Evaluation criteria and metrics for model drift detection

As a metric for drift detection, we use is_drift, which we get from the prediction data where is_drift is set to 1 if the sample tested has drifted from the reference data and 0 otherwise. Another metric is p_value, which represents the significance of the FET test. When p_val is less than 0.05, we can assume that there is a drift. We can get other metrics from the prediction object, such as threshold and distance, but we don't use them in our experiment.

To evaluate the impact of detected drift, we assess how it affects model performance metrics such as accuracy, loss, kappa, recall, F1 score, roc_auc on each client. Figure 7 displays the decision boundaries and represents model accuracies evaluated on different datasets:

- Reference dataset
  - Accuracy: 100%
  - This plot shows the reference data, where the model achieves 100% accuracy.

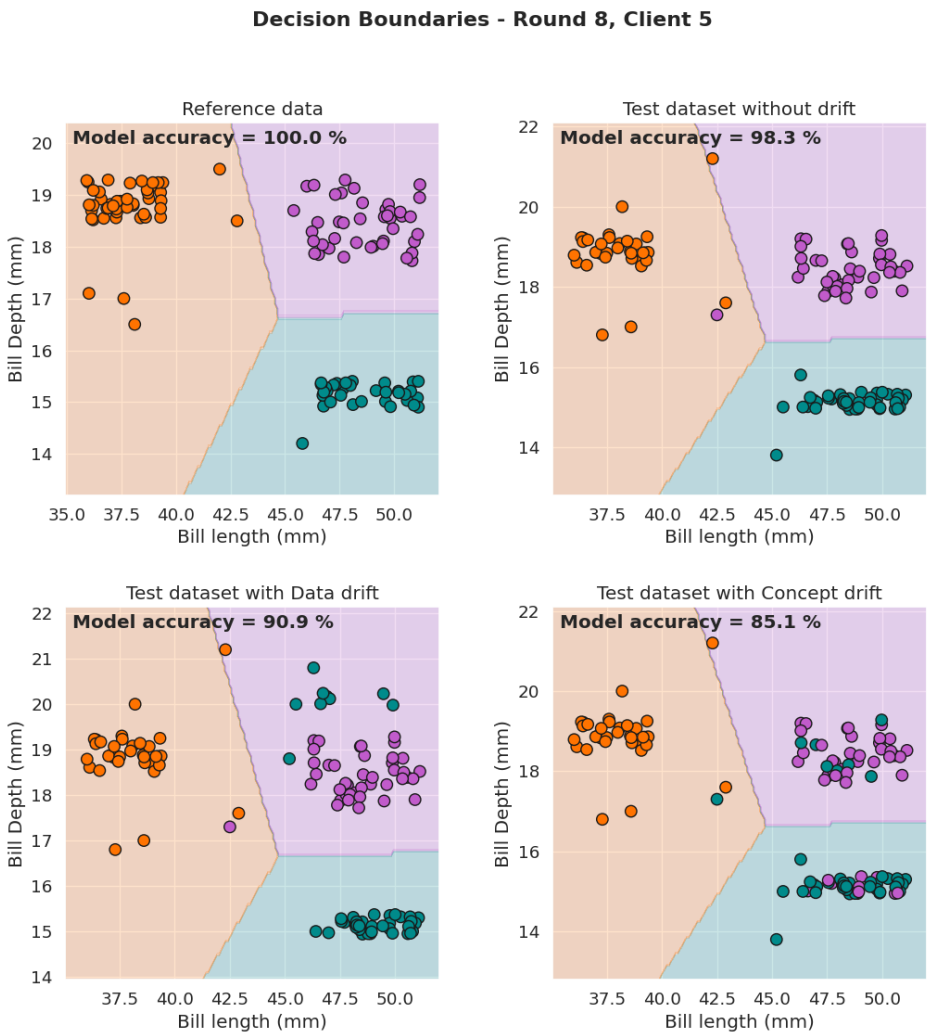


Figure 7. Client-side accuracy evaluation

This scenario shows the model's performance on the original (reference) dataset, where the decision boundaries align perfectly with the data points. Since the model is trained and tested on data from the same distribution, it classifies all points correctly.

- A test dataset without drift
  - Accuracy: 98.3%

  - This plot displays the model's performance on a test dataset without any drift. The decision boundaries are still in good alignment with the test data because they come from the same distribution as the training data. The slight drop in accuracy may have been caused by the natural variability in the data, but overall, the model managed to classify the new but similar data very well.

- A test dataset with data drift
  - Accuracy: 90.9%
  - Data drift refers to changes in input data distribution over time while the relationship between input and output (i.e., the concept) remains the same. The different locations of the data points in the plot, in this case, show that the distribution of the test data has changed in comparison to the training data. This shift causes some of the points to fall into areas where the model's decision boundaries are not optimal, which leads to an accuracy decline. Misclassifications increase when the model struggles with data that is different from its training set.

- A test dataset with concept drift
  - Accuracy: 85.1%
  - The cases when the input data distribution doesn't vary, but the underlying relationship between input features and the target label changes over time indicate concept drift. This drift is often more challenging to handle because the model's decision boundaries, which were optimal for the training data, no longer correctly separate the classes in the test data. In this plot, we can see that the decision borders in the plot are less effective in accurately classifying the data points, which leads to a further decline in accuracy.

In addition, on the server side, we aggregate metrics from all clients using the FedAvg aggregation method and evaluate the average accuracy, loss, kappa, recall, F1 score, and roc_auc metrics to see how client drift impacts the model's performance.

5 Experimental Setup and Results

The Experimental Setup and Results section provides a detailed description of the experimental framework used to implement and test the proposed model drift detection method. It goes into detail on the experiment's setup, creating the federated learning environment with Flower and integration of Alibi Detect for drift detection. This section additionally presents the experimental findings demonstrating how well the model performs in different scenarios involving concept and data drift across numerous clients. Moreover, various plots and tables illustrate the impact of drift on key performance metrics such as accuracy, loss, Kappa, F1 score, recall, and ROC AUC. The results are analyzed to demonstrate how well the suggested approach works to detect and measure the effects of drift on model performance in federated learning environments.

5.1 Experimental setup

In the experimental setup, we focus on a federated learning environment where multiple clients participate. Each client trains a machine learning model using its local dataset, and these models are then aggregated centrally. While drift can have a major impact on model performance, it is important to monitor and address it to ensure a robust and accurate model.

The experiment was carried out in a machine with an Intel 11th Gen i7-1165G7 CPU (8 vCPUs, 2.8GHz), 16GB of DDR4 RAM, and an NVIDIA GeForce GTX 1650 with Max-Q Design GPU with 4GB of dedicated VRAM.

The software used was Python3, Nvidia CUDA/CUDNN, Flower, Alibi-Detect, NumPy, pandas, matplotlib, scikit-learn, seaborn, palmerpenguins dataset.

In this experiment, we have 10 clients and 1 server with the FedAvg aggregation method over 50 rounds. We divide the dataset to guarantee that each client receives data of equal size, which is then distributed uniquely throughout the rounds. Each client uses a Logistic Regression as the model for training and evaluation. Here is a detailed explanation of the data preparation process:

- Data Distribution Among Clients: First, we divided the dataset equally among the 10 clients, ensuring that each client gets a dataset of the same size;
- Splitting Data for Training and Reference: For each client, we split the portion of data assigned into training and reference datasets in a 50% proportion;
- Splitting the Reference Data into Reference and Test Sets: We split the reference dataset into reference and test datasets equally. As a result, 25% of the data per round is used as a reference set and another 25% as a test set;

- Introducing Data Drift and Concept Drift: To simulate real-world scenarios, we introduce data drift and concept drift into 20% of the test set.

By organizing the data preparation this way, we ensure that each client is trained and evaluated on different data subsets. This method helps us to make the Federated Learning model more reliable and adaptable. Additionally, by adding data drift and concept drift to the test set, we can see how well the model performs with changing data, which makes the experiment more similar to real-world situations.

During the evaluation phase, we initially assessed the model's performance using various metrics such as accuracy, loss, Cohen's kappa, F1 score, recall, and ROC AUC to set benchmarks for the model's effectiveness and robustness. The model was then reevaluated several times with an increasing number of drifted clients, and the performance was compared against the benchmarks. Finally, the differences in these metrics are visualized using matplotlib, which provides a clear representation of the impact of drift on the model.

5.2 Implementation

Our experiment focused on detecting data and concept drift in individual clients within a federated learning system and analyzing its impact on aggregated metrics at the server level.

The proposed model drift detection method integrates the Fisher Exact Test (FET) from the Alibi Detect library to Flower to monitor for changes in data distribution in a federated learning environment. The experiment involved 10 clients and 50 rounds. It starts with preparing the data for drift detection.

Initially, we performed the federated learning process with all ten clients using their original dataset without any shifts. For each client and round, after dividing the data into training, reference, and test sets, we trained the model on the training data.

Then, we introduced artificial drift into 20% of the randomly selected client's test data. To create the concept drift, we switched the labels of the Gentoo and Chinstrap penguins. This simulates real-world scenarios where the data distribution might change over time. To introduce the data drift to the dataset, we changed feature values by adding 5mm to the bill length of all the Gentoo penguins.

The next step included creating the detector and evaluating the effectiveness of our method in detecting drift across clients.

First, after the training step, we evaluated the trained model using both reference and test datasets. It calculates the error rates for these predictions, identifying differences between predicted and true labels. The reference dataset represents the model's perfor-

mance under baseline conditions, which is without drift, while the test dataset is used to check if the model's performance has shifted.

Then, we initialize the FET drift detector on the reference dataset using the error rates:

```
cd_fet = FETDrift(loss_ref, p_val=0.05, alternative='less')
```

where

- `loss_ref` represents the error rates from the model's predictions on the reference dataset. It is a binary array where 0 indicates an incorrect classification, and 1 indicates a correct classification. For example:

  ```
  [1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0]
  ```

- `p_val` the p-value used to determine the significance of Fisher's Exact Test (FET). In this context, a p-value of 0.05 is chosen, indicating a 5% significance level for detecting drift.

- `alternative` defines the alternative hypothesis for the test. The value `'less'` means we will only flag drift only if the model's performance has degraded, i.e., the proportion of correct predictions (1s) to incorrect predictions (0s) decreases compared to the reference data. Other options include `'greater'` (testing for improvement) and `'two-sided'` (testing for any change in performance).

To detect drift, we pass the prediction losses for the test data to the FET drift detector to compare the error rates from the reference and test datasets. This test checks whether the errors in the test data significantly differ from those in the reference data. If such differences are found, it suggests a potential drift. The test returns a boolean value indicating whether drift is present and a p-value that shows the statistical significance of any observed changes. A low p-value (less than 0.05) indicates that the test data has drifted compared to the reference data:

```
preds = cd_fet.predict(np.array(loss_test))
```

where

- `cd_fet` is the drift detector created using the Fisher's Exact Test initialized with the reference loss.

- `loss_test` is the array of prediction losses for the test data. Similar to `loss_ref`, it consists of binary values where `1` represents a correct classification and `0` represents an incorrect classification.

By passing `loss_test` to the `predict` method of `cd_fet`, we compare the error rates from the reference and test datasets to detect any performance degradation (drift). The obtained result `preds` is represented as a dictionary with `meta` and `data` keys. `meta` contains the detector's metadata, while `data` is a dictionary that includes the actual predictions, which contains the following keys:

- `is_drift` represents `1` if the tested sample has drifted from the reference data and `0` otherwise.
- `p_val` shows the statistical significance of the observed changes.

In the second part of our experiment, we analyzed how the model's performance declines in cases of drifted test datasets.

We created baseline metrics for accuracy, loss, F1 score, recall, ROC AUC, and Cohen's kappa while evaluating the original dataset without any drift. It gave us a clear understanding of system performance free from drift and served as a benchmark for further evaluations.

Then, to see how drift affects the model's overall performance, we gradually introduced it to different numbers of clients. First, we started by introducing drift to the test dataset of one randomly selected client and then reran the federated learning process for 50 rounds. After completing the process, we calculated and compared metrics with the baseline to understand how much the drift of a single client affected the global model performance.

As a next step, we repeated the whole process by adding drift to two randomly selected clients and again compared the new metrics to the baseline. Then, we gradually performed the process for three, four, five, seven, and all ten clients. This incremental approach allowed us to observe how the model's performance metrics degrade or change.

We conducted the whole experiment for both data and concept drift separately. In the last step, we analyzed the results. By performing this experiment, we identified how resilient our model is towards drift.

5.3 Results

Our experiment first investigated how successfully the proposed drift detection method could identify data and concept drift in a federated learning environment. Following this, we tested and evaluated the model's performance with different levels of model drift (data and concept drift) among the clients. The main goal was to understand how these types of model drift affected the global model's performance and to evaluate the effectiveness of our drift detection method in coping with these challenges.

Our drift detection approach was based on the Fisher Exact Test, which provided accurate signals of when and where drift occurred in each client's dataset. We used this statistical test to detect and evaluate the significance of changes between the reference (baseline) and test (drifted) datasets.

The results of the concept drift detection on a randomly chosen client can be observed using:

```
print("Drift?", preds["data"]["is_drift"])
print("p-value:", preds["data"]["p_val"])
```

where

- `preds["data"]["is_drift"]` is a boolean value indicating whether drift was detected. If it is `1`, it means that the model's performance has significantly degraded based on the specified `p_val`.
- `preds["data"]["p_val"]` is the significance value of the Fisher's Exact Test. This p-value indicates the strength of the evidence against the null hypothesis (no drift). A smaller p-value suggests stronger evidence of drift.

These results are a sample from a larger experiment conducted with 10 clients over 50 rounds. In this experiment, concept drift was introduced to randomly chosen client `CID 5` (client id). Here are a few examples of the drift detection outcomes for specific clients and rounds:

```
Drift detection for client ID 8 and 4th round:
Concept Drift? No!
p-value: 1.0

Drift detection for client ID 5 and 9th round:
Concept Drift? Yes!
p-value: 0.0271689497716895

Drift detection for client ID 2 and 13th round:
Concept Drift? No!
p-value: 1.0
```

As we can see, this indicates different results for different client IDs and rounds. The p-value for client ID 8 in the fourth round, which is equal to 1.0, demonstrates that there is no decline in the model's performance. On the other hand, concept drift was detected for client ID 5 in the ninth round, with a p-value of about 0.027. Considering

that this p-value is below the 0.05 threshold, there is robust evidence of performance degradation, which means the model's performance has declined. Similarly, no drift was detected for client ID 2 in the 13th round, and the p-value of 1.0 indicates no evidence of performance decline, which means that the model's performance remained consistent with the reference data.

We got similar results when we repeated the whole experiment with 10 clients and 50 rounds to detect the data drift introduced to a randomly chosen client.

In the second part of our experiment, we observed the model's performance under typical conditions with no client drift, which we take as our baseline. In Figure 8, we saw a slight decrease in accuracy and a small rise in a loss in Figure 10 when concept drift was introduced in only one client. It demonstrates that drift can affect model performance even in small amounts.

The impact of the drift was stronger as we raised the number of clients with drift from two to ten. Figure 8 shows that for every additional client experiencing concept drift, the global model's accuracy declined systematically. Similarly, Figure 10 demonstrates that as more clients experienced drift, the loss increased, which indicates that greater drift reduces the model's ability to generalize successfully.

Data drift followed a similar pattern. Figure 9 shows a progressive decrease in accuracy when data drift was introduced to more clients, while Figure 11 illustrates a corresponding increase in loss, which in turn also emphasizes the negative impact of data drift on model performance.

To further understand the effect of drift, we examined additional metrics. The tables provided below show the average values of key performance metrics across 50 rounds under different drift scenarios.

Table 3 illustrates the results for concept drift. As the number of clients experiencing drift increased, we observed a decline in Cohen's Kappa, F1 score, Recall, and ROC AUC values. This indicates that the model's alignment with true labels, ability to classify instances correctly, and overall capability to distinguish between classes were negatively impacted by concept drift.

Similarly, Table 4 presents the performance metrics under data drift scenarios. As with concept drift, an increase in the number of clients encountering data drift led to a decrease in the metrics in the experiment.

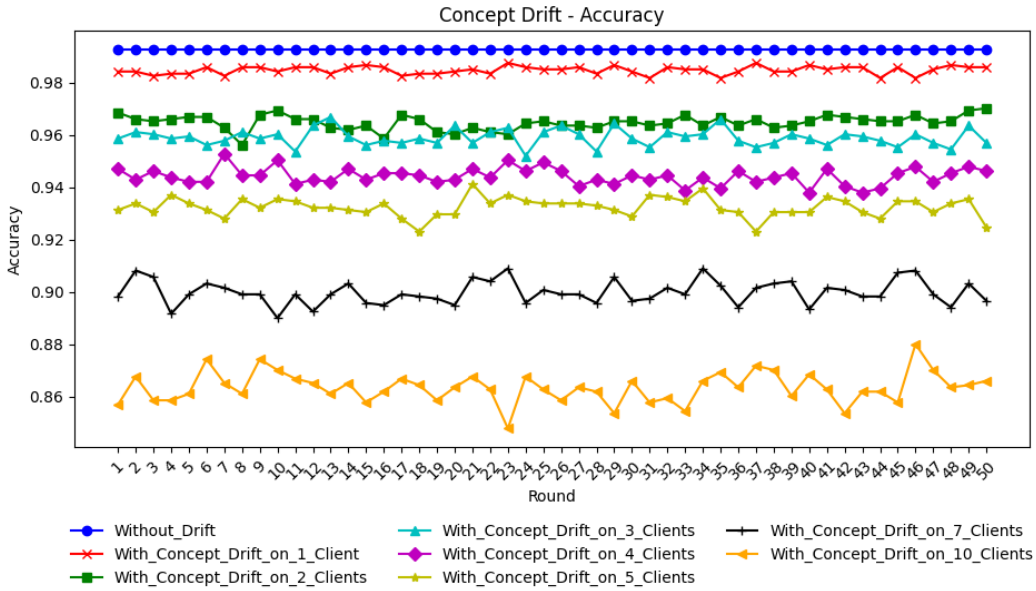


Figure 8. Accuracy analysis plot with Concept Drift

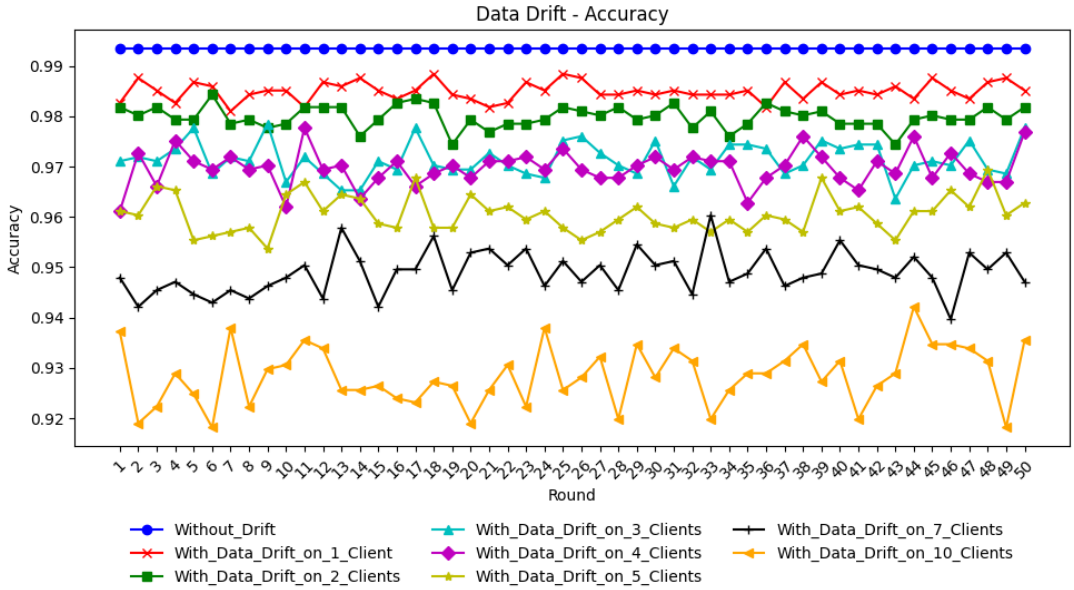


Figure 9. Accuracy analysis plot with Data drift

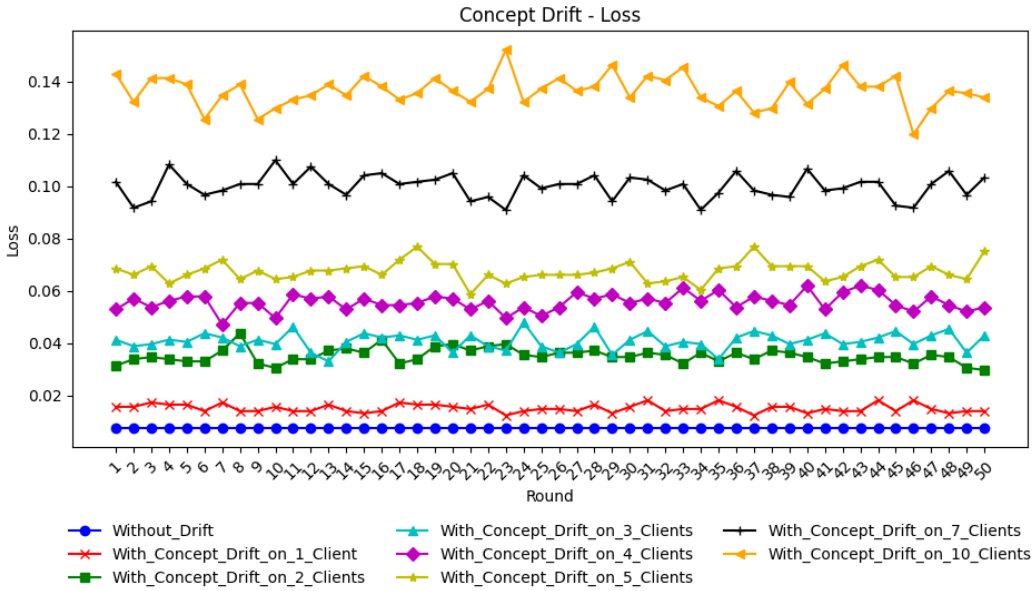


Figure 10. Loss analysis plot with Concept Drift

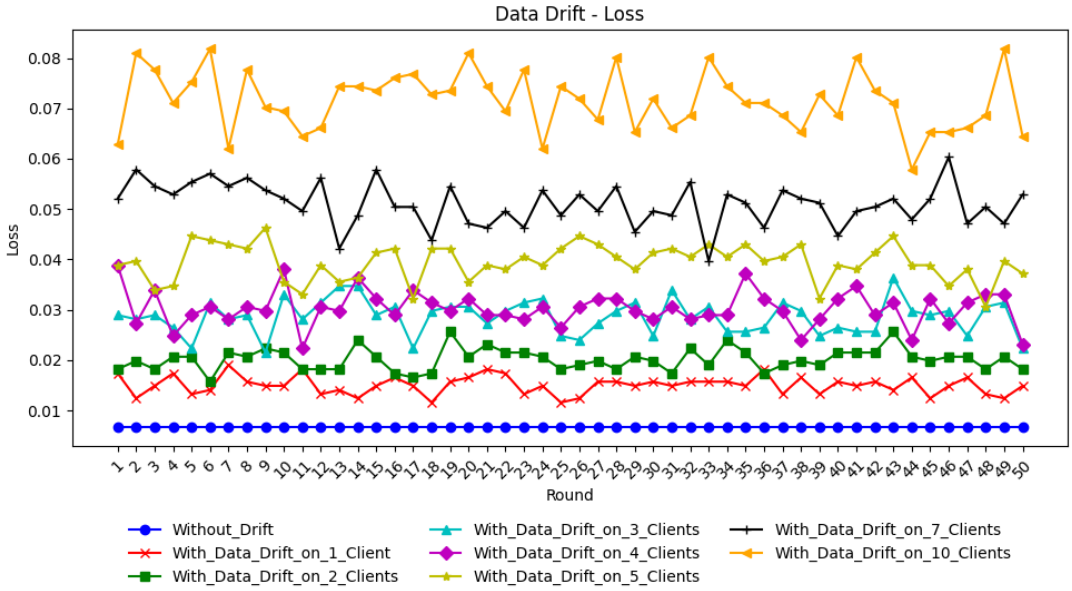


Figure 11. Loss analysis plot with Data drift

Table 3. Performance metrics across different scenarios with concept drift

Scenario	**Cohen's kappa**	**F1 score**	**Recall**	**ROC AUC**
No drift	0.989934	0.993380	0.993388	0.999948
Concept drift in 1 clients	0.977035	0.984849	0.984826	0.994534
Concept drift in 2 clients	0.946434	0.964755	0.964843	0.984434
Concept drift in 3 clients	0.938424	0.959083	0.959107	0.980938
Concept drift in 4 clients	0.915629	0.944327	0.944264	0.973032
Concept drift in 5 clients	0.897179	0.932197	0.932529	0.969291
Concept drift in 7 clients	0.848843	0.900198	0.899983	0.950248
Concept drift in 10 clients	0.793838	0.863679	0.863587	0.932654

Table 4. Performance metrics for scenarios with data drift

Scenario	**Cohen's kappa**	**F1 score**	**Recall**	**ROC AUC**
No drift	0.989934	0.993380	0.993388	0.999948
Data drift in 1 clients	0.977167	0.985040	0.984992	0.995522
Data drift in 2 clients	0.969674	0.979917	0.979868	0.986284
Data drift in 3 clients	0.957028	0.971279	0.971405	0.981211
Data drift in 4 clients	0.954245	0.969324	0.969736	0.973323
Data drift in 5 clients	0.940579	0.960250	0.960545	0.962468
Data drift in 7 clients	0.923194	0.948580	0.949008	0.955857
Data drift in 10 clients	0.892180	0.927835	0.928446	0.936839

Together, these findings confirm that model drift, whether in concept or data, significantly impacts a model's performance. As more clients experience drift, the global model's accuracy, overall agreement with true labels (Cohen's Kappa), precision (F1 score), sensitivity (Recall), and class discrimination ability (ROC AUC) all show a decline, while the loss experiences an increase. The analysis shows that the model's effectiveness declines as drift becomes more widespread, making it less reliable in real-world applications where drift is common. This underlines how crucial it is to detect drift to manage and reduce the negative impacts on model performance and maintain the reliability and effectiveness of federated learning models in dynamic environments.

6 Discussion

This discussion section thoroughly analyzes the experimental results, analyzes them in context with the formulated research questions, and considers their implications for federated machine learning. Moreover, it describes the limitations and challenges faced during the study.

6.1 Analysis of the results in the context of the research questions

The results of our experiments provide strong evidence for the effectiveness of the proposed model drift detection method in a federated learning environment. First, the systematic introduction of drift in varying client nodes helped clearly define the nature and consequences of model drift in federated learning environments. By using Flower for federated learning and Alibi Detect for drift detection, we analyzed and understood the impact of concept and data drift on various performance metrics. The method effectively detected and measured the degradation in model performance caused by drift.

The method simplified early detection of drift, allowing for timely interventions. The plots for accuracy, loss, and tables representing Cohen's kappa, F1 score, recall, and ROC AUC demonstrated a clear contrast between normal and drift conditions. By identifying even minimal drifts (as shown with one client experiencing drift), the method provides a preventative approach to drift management. This early detection is crucial for maintaining model reliability and performance over time.

The approach demonstrates reliability in various situations where different numbers of clients experienced drift. As the number of clients with drift increased, the method recognized significant performance decreases. This scalability guarantees that the method is highly flexible and can be implemented in federated learning systems of different sizes and configurations.

By evaluating multiple performance metrics, the proposed method gives a detailed understanding of model performance under drift conditions. Metrics such as accuracy, loss, Kappa, F1 score, recall, and ROC AUC provide insights into the model's performance. For example, while accuracy and loss highlighted general performance patterns, Kappa and F1 scores provided deeper insights into classification agreement and balance between precision and recall.

6.2 Implications of the findings for federated machine learning[2]

A key component of our methodology was the integration of Alibi Detect into the federated learning framework Flower to handle model drift detection. This novel architecture demonstrated its effectiveness in detecting and measuring drift among different nodes, which indicates its relevance for distributed systems where traditional centralized monitoring is impractical. The successful implementation of Alibi Detect into Flower maintained the decentralized, privacy-preserving nature of federated learning and applied a novel, precise drift detection method based on Fisher's Exact Test in the federated learning environment.

The experimental results reflect real-world scenarios where federated learning models are deployed across diverse and dynamic environments. The ability to detect and handle both concept and data drift guarantees that models remain robust and effective in practical applications, from healthcare to finance and beyond. This real-world applicability highlights the practical value of the proposed method.

The effectiveness of drift detection plays a crucial role in improving model maintenance and lifecycle management. For example, the significant impact of drift on recall and ROC AUC (as shown in Tables 3 and 4) indicates the need for increased sensitivity in model adjustments. Data scientists and engineers can take specific actions like model retraining, data augmentation, and algorithm adjustments by identifying when and where drift occurs. The method's ability to identify specific performance issues can help to build more focused and effective maintenance strategies.

In summary, the proposed drift detection method for federated learning environments has proven to be highly effective. It enables early detection, offers scalability, provides detailed performance analysis, and demonstrates real-world applicability. Moreover, the framework's performance makes it a promising solution for detecting and managing drift in complex federated learning networks for future experiments and applications.

6.3 Limitations of the study

Although this study made a significant contribution to the federated machine learning and model drift detection field, its scope and depth were limited by several issues. Understanding these limitations is essential for interpreting the findings and guiding future research efforts.

One significant restriction we faced during research was the limited timeframe available. The complexity of integrating drift detection into a federated learning environment requires proper initial planning, implementation, and analysis, which can be a time-

[2]This section was written with the assistance of OpenAI [29], which helped in structuring my ideas and observations and provided guidance on how to describe the implications of the research findings.

consuming process. Due to time limitations, we could not fully complete some of the potentially valuable experiments, such as a more profound analysis of various drift detection techniques or more detailed testing with a wider range of data distributions. This time restriction affected the depth of analysis and limited the research of alternative hypotheses or additional experiments that could have enriched the findings.

Another critical limitation was the need for more existing research related to the specific topic of model drift detection in federated learning environments. Federated learning is a relatively new field, and even less research is being done specifically analyzing the implementation of federated learning with model drift detection. Due to the lack of existing studies, comparing results or drawing conclusions from proven methodologies was challenging, which could have improved the study's planning and implementation. Since we experienced a lack of guidance from previous works, we had to take a more experimental approach. This innovative method, however, led to more uncertainties and assumptions in the research process.

During the research, we also encountered specific data distribution issues. The idea was first to divide the dataset among the clients without shuffling, which would have resulted in some clients not receiving examples of all species and creating a non-IID (independent and identically distributed) data scenario. This non-IID condition is a known challenge in federated learning, and in this study, it could have been more problematic as it could skew model training and impact drift detection since it would have required additional analysis and steps to ensure a robust drift detection method across diverse data distributions.

To focus on the study's main goal and ensure reliable results within the available timeframe, we decided to shuffle the dataset to distribute data more uniformly among all clients and maintain an IID (independent and identically distributed) scenario. It allowed us to avoid the complexities with non-IID data and provided a smoother implementation and evaluation of our drift detection methods. However, this approach left the question of how non-IID conditions affect model drift and overall performance.

Another significant challenge was the resources available on a local machine. The initial stages of the research were conducted using local computational resources, which were insufficient for the demands of training models and detecting drift in federated learning across multiple clients and many rounds. This caused longer training times and slower data processing, which was solved by experiment migration to a cloud environment.

Lastly, the experiment with different libraries and approaches that did not succeed also limited the study. During the experiment, we tested various tools and methods to integrate drift detection methods into the federated learning environment. However, many of these did not meet the project's needs or failed during the integration step. Even though this trial-and-error process was informative, it consumed valuable time and

resources that could have been spent implementing successful techniques.

As a result, even if we faced limitations and challenges during the experiment, the research significantly contributed to the fields of federated machine learning and model drift detection. By understanding and documenting these limitations, we provide a clear picture of the study context and offer a pathway for future studies based on this work to advance the abilities of federated learning in dealing with model drift.

7 Conclusion and Future Work

7.1 Summary of key findings

The study thoroughly analyzed model drift detection in federated learning environments using the Flower framework and Alibi Detect library. The research provided some important understandings of how concept drift and data drift affect the performance of federated models through several tests.

One of the key findings was the method's ability to detect early signs of drift, which is critical for maintaining the integrity and reliability of machine learning models over time. When we introduced drift to different numbers of clients, we found out that model performance changed even with very small amounts of drift, such as only in one client. This demonstrates the proposed drift detection method's sensitivity and success in identifying possible problems before they get serious.

Moreover, the analysis and visualizations of various performance metrics—accuracy, loss, Kappa, F1 score, recall, and ROC AUC- helped understand how drift impacts model performance.

Overall, the experiments proved the proposed drift detection method in detecting and identifying the impact of both concept and data drift in federated learning. The method's adaptability to many measurements and settings highlights its usefulness for federated learning applications in the real world.

7.2 Suggestions for future research directions

Although this thesis has established a solid basis for building a framework to identify model drift in federated learning environments, there is still more to investigate. The findings from this study have several exciting directions for future investigation that can further enhance the robustness and applicability of federated learning systems. The following topics indicate important ideas for further research:

- **Exploring Non-IID Data Distributions**: Although in this study, we used IID data distributions to make the initial analysis easier and allowed the study to concentrate on creating baseline drift detection techniques, experiments with non-IID data distributions should be a part of future work. Non-IID data represents real-world scenarios where data distribution and features can vary significantly between clients. Investigating model drift under these conditions will provide deeper insights into the challenges and effectiveness of drift detection methods in more complex and realistic federated learning settings.
- **Extension to Unsupervised Learning**: The current research focused on supervised federated learning environments. However, many real-world federated learning

applications may not have labeled data available, which makes unsupervised learning approaches necessary. Future research could develop and validate drift detection methods that can be applied to unsupervised federated learning. This expansion would significantly impact the applicability of federated learning models, especially in fields where it is expensive or impractical to get labeled data.

- **Implementation of Online Federated Learning**: Implementing drift detection techniques into practice and evaluating them in online federated learning environments is another exciting topic of future research. Online learning allows for continuous updates to the model as new data arrives, presenting unique opportunities and challenges for drift detection. Research in this area would focus on developing dynamic strategies that can quickly adapt to new data and changing circumstances without requiring full model retraining.

- **Management of Detected Drift**: Although this study focused on model drift detection, managing and responding to drift once it is detected is equally important. Future research will focus on developing strategies and mechanisms to adapt or update models once drift is detected. To keep federated learning models reliable and performing well over time, it will be essential to explore different management techniques.

Each of these directions has its challenges and opportunities. Exploring these areas in future research will provide a better theoretical understanding of model drift in federated learning and improve the effectiveness, applicability, and reliability of federated learning systems in real-world scenarios.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Samuel Ackerman, Eitan Farchi, Orna Raz, Marcel Zalmanovici, and Parijat Dube. Detection of data drift and outliers affecting machine learning model performance over time. *arXiv preprint arXiv:2012.09258*, 2020.

[3] Alibi Detect. Fisher's Exact Test. `https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/fetdrift.html`, 2024. Accessed: 2024-08-04.

[4] S Ancy and D Paulraj. Handling imbalanced data with concept drift by applying dynamic sampling and ensemble classification model. *Computer Communications*, 153:553–560, 2020.

[5] Feras M Awaysheh, Sadi Alawadi, and Sawsan AlZubi. Fliodt: A federated learning architecture from privacy by design to privacy by default over iot. In *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 1–6. IEEE, 2022.

[6] Firas Bayram, Bestoun S Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245:108632, 2022.

[7] Daniel J Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, et al. Flower: A friendly federated learning framework. 2022.

[8] Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021.

[9] Yuhang Chen, Wenke Huang, and Mang Ye. Fair federated learning under domain skew with local consistency and domain diversity. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 12077–12086, 2024.

[10] Yiqun Diao, Qinbin Li, and Bingsheng He. Towards addressing label skews in one-shot federated learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[11] Yiqun Diao, Qinbin Li, and Bingsheng He. Exploiting label skews in federated learning with model concatenation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11784–11792, 2024.

[12] Flower. A Friendly Federated Learning Framework. `https://flower.ai/docs/framework/contributor-explanation-architecture.html`, 2024. Accessed: 2024-08-04.

[13] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[14] Marc Haller, Christian Lenz, Robin Nachtigall, Feras M Awayshehl, and Sadi Alawadi. Handling non-iid data in federated learning: An experimental evaluation towards unified metrics. In *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 0762–0770. IEEE, 2023.

[15] Allison Marie Horst, Alison Presmanes Hill, and Kristen B Gorman. *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020. R package version 0.1.0.

[16] Chao Huang, Jianwei Huang, and Xin Liu. Cross-silo federated learning: Challenges and opportunities. *arXiv preprint arXiv:2206.12949*, 2022.

[17] Christian Janiesch, Patrick Zschech, and Kai Heinrich. Machine learning and deep learning. *Electronic Markets*, 31(3):685–695, 2021.

[18] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*, pages 5834–5853. PMLR, 2023.

[19] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[20] Hiroki Kaminaga, Feras M Awaysheh, Sadi Alawadi, and Liina Kamm. Mpcfl: Towards multi-party computation for secure federated learning aggregation. In *Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing*, pages 1–10, 2023.

[21] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[22] Yogesh Kumar, Komalpreet Kaur, and Gurpreet Singh. Machine learning aspects and its applications towards different research areas. In *2020 International conference on computation, automation and knowledge management (ICCAKM)*, pages 150–156. IEEE, 2020.

[23] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[24] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[25] Stian Lydersen, Morten W Fagerland, and Petter Laake. Recommended tests for association in $2\times$ 2 tables. *Statistics in medicine*, 28(7):1159–1175, 2009.

[26] Priyanka Mary Mammen. Federated learning: Opportunities and challenges. *arXiv preprint arXiv:2101.05428*, 2021.

[27] Akhil Mathur, Daniel J Beutel, Pedro Porto Buarque de Gusmao, Javier Fernandez-Marques, Taner Topal, Xinchi Qiu, Titouan Parcollet, Yan Gao, and Nicholas D Lane. On-device federated learning with flower. *arXiv preprint arXiv:2104.03042*, 2021.

[28] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[29] OpenAI. Chatgpt (may 13 version) [large language model]. ChatGPT, 2024. `https://chat.openai.com/`.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith

Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[31] Seldon Technologies Ltd. Alibi Detect - an open source Python library for drift detection. `https://docs.seldon.io/projects/alibi-detect/en/latest`, 2024. Accessed: 2024-08-04.

[32] TartuNLP. Neurotõlge. `https://neurotolge.ee/`, 2024. Accessed: 2024-08-13.

[33] Saheed A Tijani, Xingjun Ma, Ran Zhang, Frank Jiang, and Robin Doss. Federated learning with extreme label skew: a data extension approach. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[34] Dinesh C Verma, Graham White, Simon Julier, Stepehen Pasteris, Supriyo Chakraborty, and Greg Cirincione. Approaches to address the data skew problem in federated learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, pages 542–557. SPIE, 2019.

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

[36] Mang Ye, Xiuwen Fang, Bo Du, Pong C Yuen, and Dacheng Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys*, 56(3):1–44, 2023.

[37] Lixiang Yuan, Mingxing Duan, Guoqing Xiao, Zhuo Tang, and Kenli Li. Bm-fl: A balanced weight strategy for multi-stage federated learning against multi-client data skewing. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[38] Jie Zhang, Zhiqi Li, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, and Chao Wu. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning*, pages 26311–26329. PMLR, 2022.

[39] Indrė Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Leyla Rahimli**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Model drift in Federated Learning: an experimental analysis,
   (title of thesis)

   supervised by Feras M. Awaysheh, PhD.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Leyla Rahimli
13/08/2024