Federated Learning Drift Detection: An Empirical Study on the Impact of Concept and Data Drift

Leyla Rahimli
Institute of Computer Science
University of Tartu
Tartu, Estonia
Email: leyla.rahimli@ut.ee

Feras M. Awaysheh, Sawsan Al Zubi
Institute of Computer Science
Tartu University
Tartu, Estonia
feras.awaysheh@ut.ee

Sadi Alawadi
Department of Computer Science
Blekinge Institute of Technology
Karlskrona, Sweden
sadi.alawadi@bth.se

***Abstract*—Federated Learning (FL) has emerged as a transformative paradigm in machine learning, enabling decentralized model training while preserving data privacy across multiple clients. 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 paper proposes a novel drift detection and management methodology within federated environments. Our experimental analysis demonstrates the effectiveness of the proposed drift detection framework. 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.**

***Index Terms*—Federated Learning, Drift Detection, Model Drift, Concept Drift, Supervised Machine Learning**

1. Introduction

The need for distributed machine learning (ML) has emerged in this big data era to meet application demands and training models where data is allocated across multiple devices or servers [1]. 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 in many domains [2].

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 [3], [4]. 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. Model drift is a phenomenon where an ML model's performance declines over time when the assumptions underlying the model's training phase no longer align with real-world conditions.

This issue 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 [5], [6]. A fundamental assumption in building distributed 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.

Nevertheless, this issue is becoming more prevalent in an FL environment since data is distributed across numerous devices, each with potentially different data distributions [7]; the aggregated model must contend with various non-IID (independent and identically distributed) [8] 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 FL 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.

In this paper, we introduce an integrated approach using the Flower FL framework alongside Alibi Detect to handle the challenges of model drift in FL 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 FL settings.

The rest of this paper is organized as follows. Section

2. Background

2.1. Model Drift & FL

Model Drift [9], [10], 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 1 represents different types of model drift in federated learning compared to the scenario (Figure 1a) 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 1b). 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 [10] 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.
- **Data Drift**: a change in input data distribution or features while the relationship between the input data and the target variable remains unchanged (Figure 1c). Data drift [9], [10] 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 ML 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 their effects on the model.

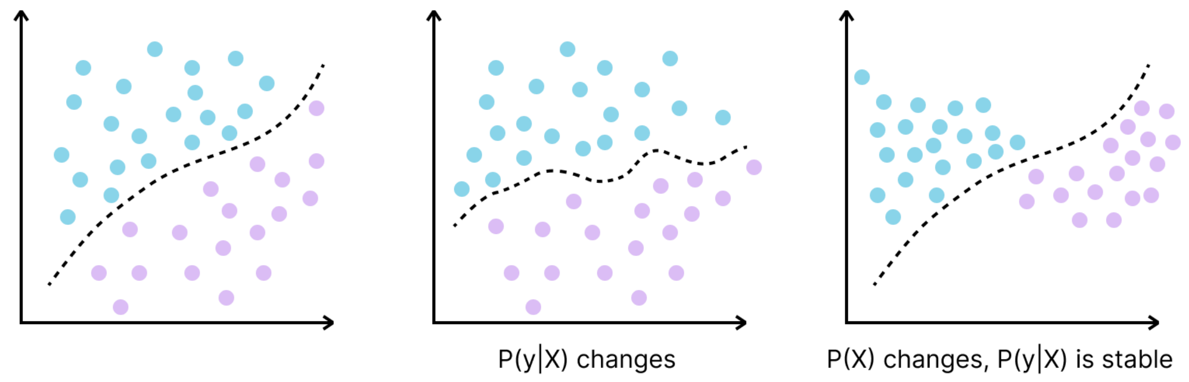
2.2. Existing model drift detection methods

Detecting model drift is an essential aspect of ML, 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. The impact of this effect is more pronounced in the edge-to-cloud continuum [11], [12].

Among the first methods for monitoring model performance are **statistical process control** (SPC) [5] 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 [5] 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 [6] are designed for settings with continuous data streams. These techniques include incremental learning and adaptive windowing



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

Figure 1. Model drift types

[5], 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, like IoT [13] and metaverse [14].

With **ensemble techniques** [5], 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 [15], 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.

2.3. Related work

In classical ML, considerable research has been published on understanding and handling model drift. Studies such as those by Gama et al. [10] and Žliobaitė [16] 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 or code-smell detection [17].

However, adapting these well-established methods to federated learning brings several 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. [18] and Li et al. [19], 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.

3. Methodology

This 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.

3.1. Model drift detection method

The proposed method uses the *Alibi-Detect* [20] 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 [21], PyTorch [22], and KeOps(where applicable) [23] 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) [24] 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. [24]

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.

3.2. Data collection and preparation process

We used the palmerpenguins dataset [26] 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. 8 features 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.

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 2-a 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.

Ultimately, we have a dataset with 4,832 examples, which was subsequently shuffled to distribute properly between clients. Figure 2-b 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.

3.3. 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 3 shows the decision boundaries and represents model accuracies evaluated over different settings:

- Figure 3-a represents the reference data, where the model achieves 100% accuracy. 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 with the same distribution, it classifies all points correctly.
- Figure 3-b displays the model's performance on a test dataset without any drift. The decision boundaries are still aligned with the test data because they come from the same distribution as the training data. The natural variability in the data may have caused the slight drop in accuracy, but overall, the model managed to classify the new but similar data very well.
- 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. Figure 3-c represent the scenario where the data drift represented in the test data, In this case, the different locations of the data points in the plot show that the distribution of the test data has changed compared to the training data. This shift causes some points to fall into areas where the model's decision boundaries are not optimal, leading to an accuracy decline. Misclassifications increase when the model struggles with data that is different from its training set.
- Lastly, 3-d introduces concept drift into the test data, where the input data distribution remains unchanged, but the underlying relationship between the input features and the target label evolves over time, indicating concept drift. This type of drift is typically more challenging to address, as the model's decision boundaries, which were optimal for the training data, no longer accurately separate the classes in the test data. In this plot, it is evident that the decision boundaries are less effective in correctly classifying the data points, resulting in 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.

4. Experiment Setup & Results

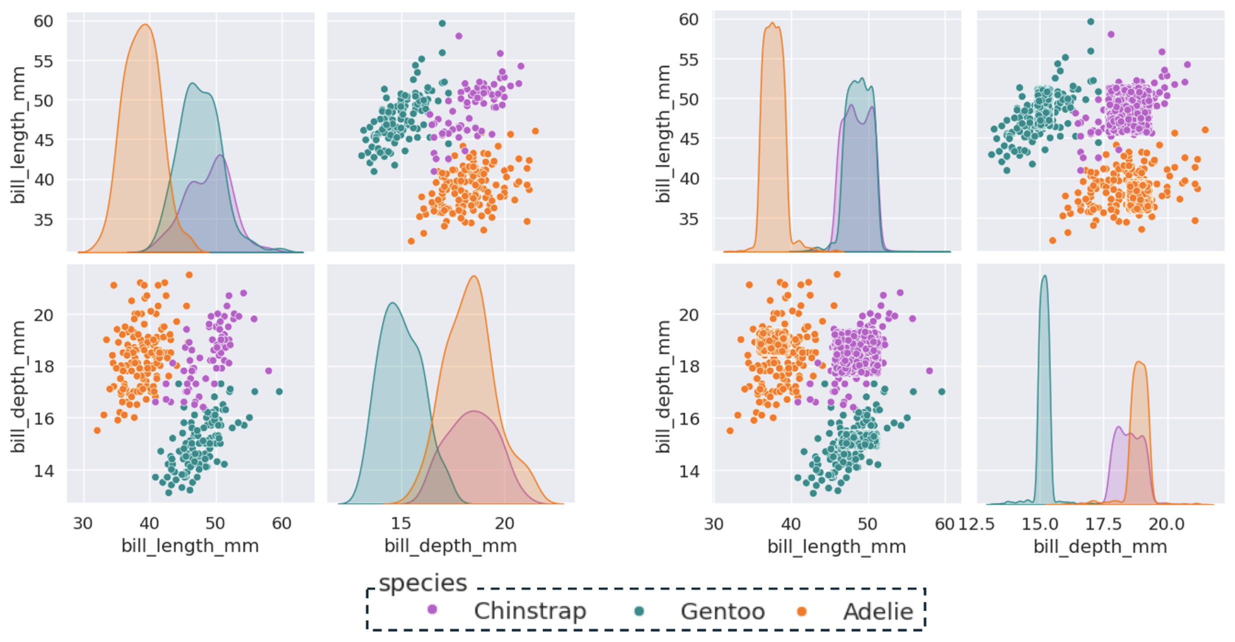
The Experimental Setup and Results section describes the experimental framework for implementing and testing the proposed model drift detection method. It details the experiment's setup, creating the federated learning environment with Flower and integrating Alibi Detect for drift detection. This section 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.

4.1. 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.

TABLE 1. Original Palmer Penguins dataset

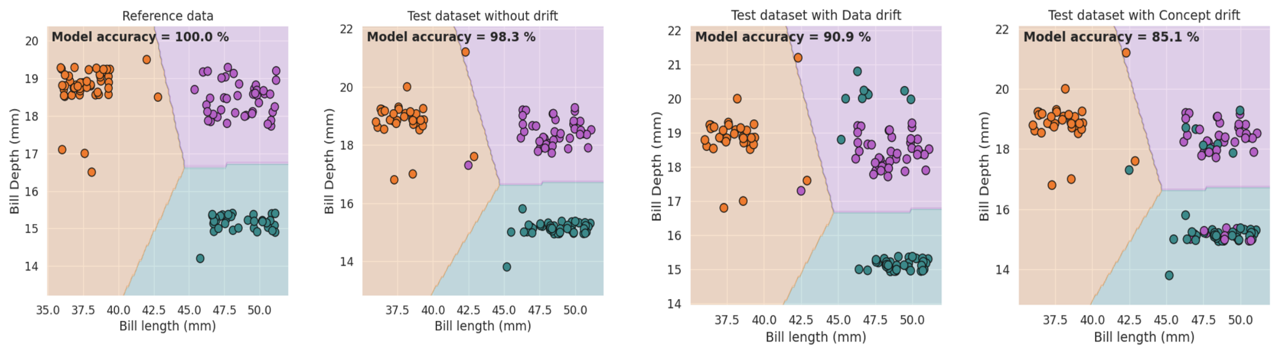
species	island	bill length mm	bill depth mm	flipper length mm	body mass g	sex	year
Adelie	Torgersen	39.1	18.7	181.0	3750.0	male	2007
Adelie	Biscoe	39.0	17.5	186.0	3550.0	female	2008
Adelie	Torgersen	40.3	18.0	195.0	3250.0	female	2007
Gentoo	Biscoe	46.7	15.3	219.0	5200.0	male	2007
Gentoo	Biscoe	43.3	13.4	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



(a) Classification plot of initial dataset

(b) Classification plot of generated synthetic dataset

Figure 2. Data distribution and classification plots for both (a) initial dataset and (b) agumented dataset



(a) (b) (c) (d)

Figure 3. The Federated Learning global model achieved varying levels of accuracy and decision boundaries under different configurations with 5 clients and 8 communication rounds.

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, mat-

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

plotlib, scikit-learn, seaborn, palmerpenguins dataset.

In this experiment, we had 10 clients and 1 server using 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 increasing drifted clients, and the performance was compared against the benchmarks. Finally, the differences in these metrics are visualized using Matplotlib, which clearly represents the impact of drift on the model.

4.2. Experiment 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. This experiment introduced concept drift 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

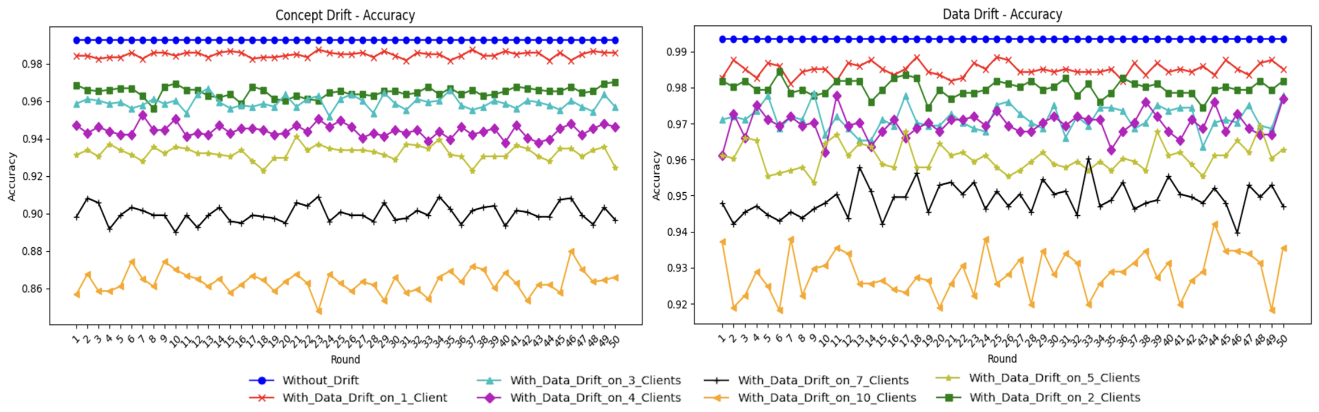
Drift detection for client ID 5 and 9th round: Concept Drift? Yes! p-value: 0.0271689497716895

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.

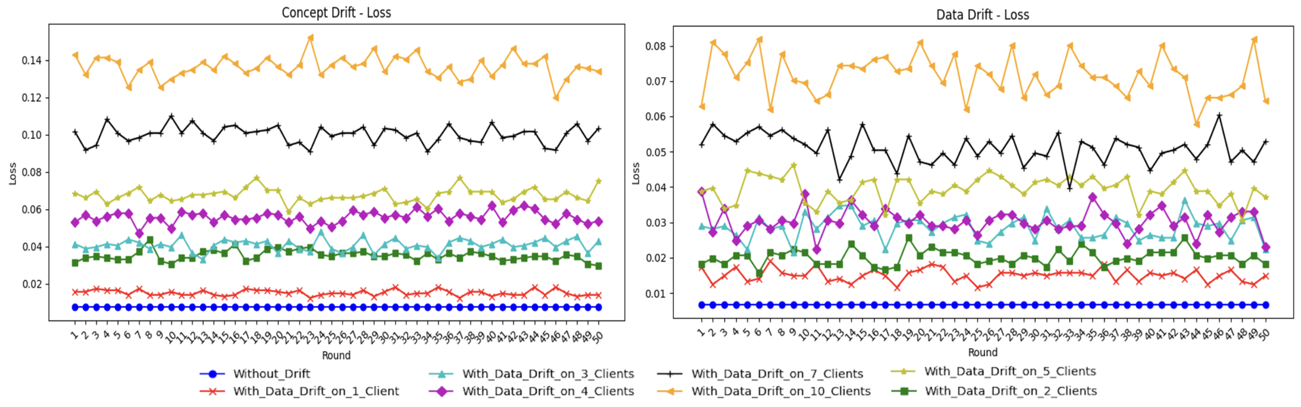
We got similar results when we repeated the whole experiment with 10 clients and 50 rounds to detect the data drift introduced to a randomly chosen client.

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 4-a, we saw a slight decrease in accuracy and a small rise in a loss in Figure 5-a when concept drift was introduced in



(a) (b)

Figure 4. Federated Learning global model obtained accuracy using 10 clients and 50 rounds with different settings. (a) illustrates the model's accuracy for each round when concept drift is introduced. (b) presents the model's accuracy when data drift is presented.



(a) (b)

Figure 5. Federated Learning global model obtained loss using 10 clients and 50 rounds with different settings. (a) illustrates the model's loss for each round when concept drift is introduced. (b) presents the model's loss when data drift is presented.

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 4-a shows that for every additional client experiencing concept drift, the global model's accuracy declined systematically. Similarly, Figure 5-a 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 4-b shows a progressive decrease in accuracy when data drift was introduced to more clients, while Figure 5-b 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.

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,

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 across different 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

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.

5. Conclusion

This study extensively investigated drift detection mechanisms in federated learning environments, employing the Flower framework and the Alibi Detect library. Our research revealed significant insights into how concept and data drift impact the performance of federated models, verified through a series of comprehensive experiments. A key discovery was the system's capacity to detect early signs of drift, which is crucial for maintaining the reliability and robustness of machine learning models over time. The experiments demonstrated that even minimal drift, such as that introduced by a single client, could lead to observable performance degradation, highlighting the proposed method's sensitivity and efficacy in identifying potential issues before they escalate.

The analysis of performance metrics—accuracy, loss, Kappa, F1 score, recall, and ROC AUC—provided a deeper understanding of how drift influences model outcomes. These metrics were instrumental in showcasing the method's effectiveness in diagnosing drift and its impact on model behavior. Overall, the results validate the proposed drift detection method's effectiveness in identifying concept and data drift in federated learning settings. Its adaptability across various measurements and scenarios underscores its potential for real-world federated learning applications, offering a valuable tool for enhancing model resilience and integrity in dynamic environments.

References

[1] F. M. Awaysheh, M. Alazab, S. Garg, D. Niyato, and C. Verikoukis, "Big data resource management & networks: Taxonomy, survey, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2098–2130, 2021.

[2] H. Kaminaga, F. M. Awaysheh, S. Alawadi, and L. 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*, 2023, pp. 1–10.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023. [Online]. Available: https://arxiv.org/abs/1602.05629

[4] F. M. Awaysheh, M. N. Aladwan, M. Alazab, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by design for big data frameworks over cloud computing," *IEEE Transactions on Engineering Management*, vol. 69, no. 6, pp. 3676–3693, 2021.

[5] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, p. 108632, 2022.

[6] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE transactions on knowledge and data engineering*, vol. 31, no. 12, pp. 2346–2363, 2018.

[7] F. M. Awaysheh, M. Alazab, M. Gupta, T. F. Pena, and J. C. Cabaleiro, "Next-generation big data federation access control: A reference model," *Future Generation Computer Systems*, vol. 108, pp. 726–741, 2020.

[8] M. Haller, C. Lenz, R. Nachtigall, F. M. Awayshehl, and S. 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)*. IEEE, 2023, pp. 0762–0770.

[9] S. Ackerman, E. Farchi, O. Raz, M. Zalmanovici, and P. Dube, "Detection of data drift and outliers affecting machine learning model performance over time," *arXiv preprint arXiv:2012.09258*, 2020.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[11] F. M. Awaysheh, "From the cloud to the edge towards a distributed and light weight secure big data pipelines for iot applications," in *Trust, security and privacy for big data*. CRC Press, 2022, pp. 50–68.

[12] F. M. Awaysheh, R. Tommasini, and A. Awad, "Big data analytics from the rich cloud to the frugal edge," in *2023 IEEE international conference on edge computing and communications (EDGE)*. IEEE, 2023, pp. 319–329.

[13] F. M. Awaysheh, S. Alawadi, and S. 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)*. IEEE, 2022, pp. 1–6.

[14] G. Yenduri, D. Reddy, G. Srivastava, Y. Supriya, M. Ramalingam, T. R. Gadekallu, and F. M. Awaysheh, "Federated learning for the metaverse: A short survey," in *2023 International Conference on Intelligent Metaverse Technologies & Applications (iMETA)*. IEEE, 2023, pp. 1–10.

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

[16] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[17] S. Alawadi, K. Alkharabsheh, F. Alkhabbas, V. R. Kebande, F. M. Awaysheh, F. Palomba, and M. Awad, “Fedcsd: A federated learning based approach for code-smell detection,” *IEEE Access*, 2024.

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

[19] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” *IEEE signal processing magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[20] 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.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[23] B. Charlier, J. Feydy, J. A. Glaunès, F.-D. Collin, and G. Durif, “Kernel operations on the gpu, with autodiff, without memory overflows,” *Journal of Machine Learning Research*, vol. 22, no. 74, pp. 1–6, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-275.html

[24] Alibi Detect, “Fisher’s Exact Test,” https://docs.seldon.io/projects/alibi-detect/en/stable/cd/methods/fetdrift.html, 2024, accessed: 2024-08-04.

[25] S. Lydersen, M. W. Fagerland, and P. Laake, “Recommended tests for association in 2$\times$ 2 tables,” *Statistics in medicine*, vol. 28, no. 7, pp. 1159–1175, 2009.

[26] A. M. Horst, A. P. Hill, and K. B. Gorman, *palmerpenguins: Palmer Archipelago (Antarctica) penguin data*, 2020, r package version 0.1.0. [Online]. Available: https://allisonhorst.github.io/palmerpenguins/