Explainable Federated Learning: Detecting Concept Drift with Feature-Level Attribution Using SAGE*

Rohit Jagtani[1], Timo Ropinski[2], and Gjergji Kasneci[1]

[1] Technical University of Munich, Munich, Germany
[2] University of Ulm, Ulm, Germany

Abstract. Loss curves alone provide limited insights into feature attributions and model behavior in Federated Learning (FL), making it challenging to diagnose performance degradation. To address this, we propose a novel framework that leverages Shapley Additive Global Explanations (SAGE) to attribute model performance to individual features by analyzing their impact on the loss. By extending SAGE to a federated setting, we gain a more detailed understanding of feature attributions across decentralized clients. Building on this capability, our approach detects concept drift by analyzing shifts over time in the distributions of loss and feature attributions contributed by each client. Our approach allows us to pinpoint which features exhibit significant drift and identify those most contributing to model degradation, thereby facilitating targeted interventions. Across four datasets, our framework achieves detection accuracy better than standard drift detection methods, while reducing average detection delay by roughly 25% and retaining fine-grained, feature-level drift interpretability. Furthermore, we approximate centralized SAGE with a mean error of roughly 8.5%, enabling accurate drift explanation in federated settings.

Keywords: Explainable AI · Federated Learning · Feature Attribution · Concept Drift · Shapley Values

1 Introduction

Federated Learning has emerged as a pivotal machine learning paradigm due to its unique capability to train models collaboratively across decentralized devices without centralizing sensitive user data [6, 29]. This approach is particularly significant in privacy-sensitive sectors such as healthcare and finance, where data security and regulatory compliance are paramount [35, 42]. By maintaining data privacy, FL inherently reduces risks of data breaches and supports compliance with stringent privacy regulations [19, 20]. Additionally, FL capitalizes on the computational power of edge devices, enhancing scalability and enabling efficient real-time applications [43].

* Code available at `https://github.com/rjagtani/fl_xai_drift`

FL can be broadly categorized into horizontal FL, where clients share the same feature space but possess different data samples, and vertical FL, where clients have different feature spaces but share overlapping user sets [42].

While FL enables collaborative model training without data centralization, it also introduces a key challenge: data heterogeneity due to the non-IID nature of distributed client data. Unlike centralized learning, where data can be homogenized, FL operates over diverse client datasets that vary in distribution, size, and quality. One key manifestation of data heterogeneity is distribution shift across clients, including concept shift, label distribution shift, and feature distribution shift [23, 27]. Among these, *concept drift*, i.e., temporal changes in the conditional distribution of the target given the features, poses a significant threat to the stability and performance of FL models over time. Consequently, the crucial task of understanding and interpreting concept drift at the feature level remains largely unaddressed in the FL setting.

Detecting and interpreting concept drift in FL is a difficult problem for several reasons. First, privacy constraints prevent direct access to client-level data, limiting the ability to perform classical drift detection analyses. Second, the heterogeneity across clients makes it hard to distinguish between normal client-specific variation and true distribution shifts. Third, feature importance is typically not observable at the server level, preventing straightforward identification of features responsible for drift. Finally, concept drift may emerge gradually or abruptly, requiring flexible methods that can detect both subtle and severe changes.

To address this, we propose a two-part approach. First, we introduce *Federated SAGE*, an extension of the model agnostic SAGE [15] method to FL, that provides system-level feature attributions by aggregating client-wise SAGE values. This allows us to measure how individual features contribute to global model performance in a federated system. Second, we propose a two-stage, distribution-based drift detection framework, to the best of our knowledge, the first to enable feature-level drift understanding in FL, that tracks the distribution of client-level losses and feature attributions over time and is sensitive to the severity of drift. By analyzing the distribution of client metrics rather than simple aggregates, our framework captures subtle drifts and reveals client heterogeneity and disagreement in feature attributions.

1.1 Main Contributions

Our paper introduces a novel feature attribution-based framework for detecting and interpreting concept drift in federated learning environments. The primary contributions of this work are:

- Federated SAGE, an extension of SAGE to federated learning that explains global model performance in terms of feature contributions by aggregating client feature attributions, while preserving the decomposition of loss reduction at the system level.

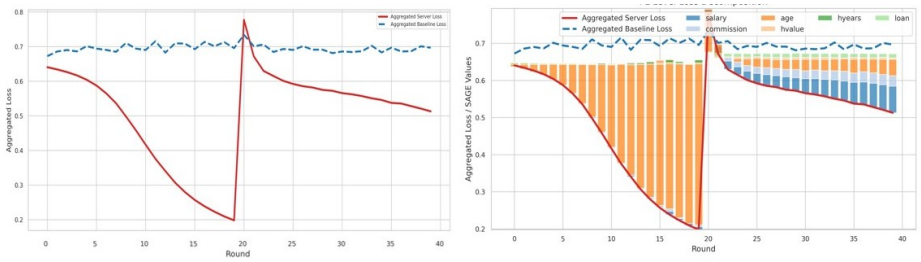

Fig. 1: FL on synthetic data based on the Agrawal stream. Initially, the feature salary is not relevant to the classification task. After round 20, a concept drift alters the decision boundary, making salary important. Loss curves (left) reflect performance changes but do not reveal the underlying cause, while feature attributions in FL (right) expose the drivers of drift.

- A two-stage, distribution-based drift detection framework, the first approach to enable feature-level drift understanding in FL, that tracks the distribution of client-level losses and feature attributions over time and is sensitive to the severity of drift.
- Comprehensive validation of our framework through experiments on synthetic and real-world datasets under a variety of drift scenarios, demonstrating its ability to effectively detect and explain concept drift in FL systems, while consistently matching or outperforming standard drift detection baselines.

2 Related Work

Explainable Artificial Intelligence (XAI) and FL have increasingly intersected as researchers aim to make privacy-preserving models interpretable. Efforts in explainable FL range from by-design models like fuzzy systems and rule-based classifiers [7, 16], to post-hoc explanations using SHAP and feature importance methods adapted for FL [11, 17]. More recent studies examine explanation consistency and privacy trade-offs in distributed settings [36, 40], alongside emerging surveys that attempt to consolidate this growing field [25].

Concept drift in FL has also gained attention, focusing on detection and adaptation to non-stationary data. Adaptive optimization methods like Adaptive-FedAvg and CDA-FedAvg tackle drift through learning rate adjustments and continual adaptation [8, 9, 33], while frameworks such as FedConD address asynchronous client drift via regularization [10]. Client-focused methods aim to identify drifted nodes through clustering or adaptive aggregation [28, 22]. However, these approaches typically lack feature-level insights and interpretability. Broader empirical studies on FL under data and concept drift highlight the impact on performance but do not provide fine-grained explanations [34].

Beyond FL, classical drift detection techniques like DDM, ADWIN, and Page-Hinkley [18, 5, 32] remain widely used but focus on performance metrics rather

than interpretability. Recent efforts to adapt explanation methods for online and evolving data streams [30, 31], and domain-specific explainable drift frameworks [1] demonstrate the potential of feature-level drift understanding but are yet to be integrated into federated learning scenarios. Our work bridges this gap by introducing a feature attribution-based framework that detects and explains concept drift in FL, providing actionable insights into which features contribute to model degradation.

3 Methodology

In this section, we present our federated framework for explainability and concept drift detection, which provides fine-grained, feature-level insight into model degradation under federated learning. Our approach is centered on extending SAGE [15], a global, model-agnostic method for feature attribution, to operate efficiently and accurately in the federated setting. We formally define Federated SAGE, detail its aggregation protocol, and state key theoretical guarantees, including unbiased estimation and valid loss decomposition at the system level. All theoretical results are proved in Appendix C.

3.1 Federated SAGE: Global Feature Attribution in FL

Notation and setup. Let $f_\theta \colon \mathbb{R}^d \to \mathcal{Y}$ be a predictive model parameterized by θ, operating on d-dimensional inputs $x = (x_1, \ldots, x_d)$ with labels $y \in \mathcal{Y}$. We write $\ell(f_\theta(x), y)$ for a pointwise loss (e.g. cross-entropy) and denote a data distribution by $\mathcal{D}$.

SAGE: feature-level loss decomposition. SAGE (Shapley Additive Global importancE) [15] attributes model performance to individual features by treating each feature as a player in a cooperative game. For a subset $S \subseteq \{1, \ldots, d\}$ of "known" features, the *value function* measures the expected loss when only the features in S are available to the model:

$$v(S) = \mathbb{E}\big[\ell\big(f_\theta(x_S, \tilde{x}_{\bar{S}}),\, y\big)\big], \tag{1}$$

where x_S denotes the observed features in S and $\tilde{x}_{\bar{S}}$ denotes the remaining features, replaced by draws from a reference distribution.

Ideally, the absent features $\bar{S}$ would be sampled from the conditional distribution $p(x_{\bar{S}} \mid x_S)$, but estimating high-dimensional conditionals is generally intractable (cite). SAGE therefore adopts the common *marginal* approximation (cite this paper in parenthesis): absent features are drawn independently from a *background dataset* $\mathcal{B}$, which represents the empirical marginal distribution of the data. This yields the tractable value function

$$v_{\mathcal{B}}(S) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\, \mathbb{E}_{\tilde{x}\sim\mathcal{B}}\big[\ell\big(f_\theta(x_S, \tilde{x}_{\bar{S}}),\, y\big)\big]. \tag{2}$$

The SAGE value of feature j is then its Shapley value in this cooperative game: the average marginal contribution of j across all possible coalitions of

other features (cite this paper in parenthesis). By the *efficiency* (or completeness) property of Shapley values, the attributions decompose the full loss reduction:

$$\sum_{j=1}^{d} \phi_j = v_{\mathcal{B}}(\emptyset) - v_{\mathcal{B}}(\{1,\dots,d\}) = \mathcal{L}^{\text{baseline}} - \mathcal{L}^{\text{model}}, \tag{3}$$

where $\mathcal{L}^{\text{baseline}} = v_{\mathcal{B}}(\emptyset)$ is the loss when all features are marginalized out (i.e. the model receives only background noise), and $\mathcal{L}^{\text{model}} = v_{\mathcal{B}}(\{1,\dots,d\})$ is the loss on the actual data. Each $\phi_j \geq 0$ thus quantifies how much feature j contributes to reducing the loss from the uninformative baseline. In practice, SAGE values are estimated via unbiased KernelSHAP [14], which provides consistent estimates with controlled variance.

Federated SAGE. Consider a federated learning (FL) system with K clients, indexed by $i = 1,\dots,K$. Each client i holds local data drawn from a (potentially distinct) distribution $\mathcal{D}_i$ and maintains a local test set $\mathcal{D}_i^{\text{test}}$ with n_i^{test} points. At communication round t, all clients receive a shared global model $f_{\theta^{(t)}}$. Let $w_i \geq 0$ denote the aggregation weight for client i (e.g. proportional to dataset size), with $\sum_i w_i = 1$. Each client i evaluates the global model on its local test set to obtain three quantities (all population expectations are estimated empirically; we reuse the same notation for brevity): the server model loss $\mathcal{L}_i^{\text{server}}(t)$, the baseline loss $\mathcal{L}_i^{\text{baseline}}(t)$ obtained by marginalizing all features via the local background $\mathcal{B}_i$, and per-feature SAGE values $\phi_{i,j}^{(t)}$ satisfying the efficiency property (3). The server then aggregates across the federation:

$$\mathcal{L}^{\text{server}}(t) = \sum_{i=1}^{K} w_i \, \mathcal{L}_i^{\text{server}}(t), \qquad \mathcal{L}^{\text{baseline}}(t) = \sum_{i=1}^{K} w_i \, \mathcal{L}_i^{\text{baseline}}(t), \tag{4}$$

$$\Phi_j(t) = \sum_{i=1}^{K} w_i \, \phi_{i,j}^{(t)} \qquad \text{for } j = 1,\dots,d. \tag{5}$$

Proposition 1 (Federated SAGE Loss Decomposition). *For any round t, the aggregated feature attributions preserve the global loss decomposition:*

$$\sum_{j=1}^{d} \Phi_j(t) = \mathcal{L}^{baseline}(t) - \mathcal{L}^{server}(t).$$

Proposition 2 (Unbiasedness and Variance of Federated SAGE). *Let $w_i \geq 0$ be deterministic aggregation weights with $\sum_i w_i = 1$ (e.g., $w_i = \frac{n_i}{\sum_k n_k}$). The federated SAGE estimator*

$$\hat{\Phi}_j(t) = \sum_{i=1}^{K} w_i \hat{\phi}_{i,j}(t)$$

is unbiased, and, assuming independent estimation noise across clients, its variance is

$$\text{Var}[\hat{\Phi}_j(t)] = \sum_{i=1}^{K} w_i^2 \, \text{Var}[\hat{\phi}_{i,j}(t)].$$

This follows from the fact that unbiased KernelSHAP yields unbiased SAGE estimates [14] for each client, so no additional assumption is needed.

The complete protocol is presented in Algorithm 1. At each round, clients compute server model loss, marginalized baseline loss, and feature-level SAGE values; the server then aggregates these to produce system-level losses and federated attributions.

Algorithm 1 Federated SAGE: System-Level Attribution in Federated Learning

Input: Global model $f_{\theta^{(t)}}$, test sets $\{\mathcal{D}_i^{\text{test}}\}_{i=1}^K$, local backgrounds $\{\mathcal{B}_i\}_{i=1}^K$, aggregation weights $\{w_i\}$
for each round t in which attribution is computed **do**
  for each client $i = 1, \ldots, K$ **(in parallel) do**
    Compute $\mathcal{L}_i^{\text{server}}(t)$, $\mathcal{L}_i^{\text{baseline}}(t)$ (using background $\mathcal{B}_i$), and SAGE values $\{\phi_{i,j}^{(t)}\}_{j=1}^d$
  end for
  Aggregate losses and attributions via (4)–(5)
end for
Output: Global losses and system-level feature attributions for each selected round.

Complexity. The primary computational cost arises from estimating SAGE attributions on each client, which requires multiple model evaluations per feature per data point. The overall runtime is $\mathcal{O}(K \cdot M \cdot d)$ model passes per attribution round, where M is the number of Monte Carlo samples per client. Efficient sampling strategies can help control this cost in practice (see Appendix D).

3.2 Approximation Quality: IID and Non-IID Data

We next clarify the conditions under which Federated SAGE recovers the true global feature importance that would be obtained by running SAGE on the union of all client data pooled centrally, i.e., the "centralized SAGE" computed with access to the complete dataset as in a conventional (non-federated) setting, and quantify the approximation error when client data distributions differ. Recall that each client i uses its local distribution $\mathcal{D}_i$ as the marginalization background when computing SAGE attributions. Let $\phi_{i,j}(t)$ be the true SAGE value for feature j given distribution $\mathcal{D}_i$ at round t. The system-wide or pooled (mixture) distribution is defined as

$$\mathcal{D}_{\text{cent}} = \sum_{i=1}^{K} w_i \mathcal{D}_i$$

and the *Centralized SAGE* for feature j at round t is

$$\phi_j^{\text{cent}}(t) := \text{SAGE for } f_{\theta^{(t)}} \text{ on } \mathcal{D}_{\text{cent}}.$$

Theorem 1 (IID Consistency of Federated SAGE). *Suppose all clients draw data from the same distribution, i.e., $\mathcal{D}_i = \mathcal{D}$ for all i. Then, as each client's SAGE estimator is computed with increasingly large datasets ($n_i \to \infty$ for all i) and a sufficient number of Monte Carlo samples, the federated SAGE estimate converges in probability to the centralized SAGE:*

$$\hat{\Phi}_j(t) \xrightarrow{p} \phi_j^{\text{cent}}(t) \quad as \quad \min_i n_i \to \infty,$$

for all j, with no additional aggregation or partitioning bias.

This result confirms that when client datasets are IID, Federated SAGE converges to the same solution as centralized SAGE: the only error is due to finite-sample variance (as characterized in Theorem 2).

Theorem 2 (Non-IID Heterogeneity Gap Bound). *Assume the SAGE attribution functional for feature j is L_j-Lipschitz with respect to a distance metric $d(\cdot,\cdot)$ on distributions; that is, for all distributions $\mathcal{P}, \mathcal{Q}$,*

$$|SAGE_j(f; \textit{background } \mathcal{P}) - SAGE_j(f; \textit{background } \mathcal{Q})| \leq L_j \, d(\mathcal{P}, \mathcal{Q})$$

for some sensitivity constant $L_j > 0$. Here, $SAGE_j(f; \textit{background } \mathcal{P})$ denotes the SAGE value for feature j of model f computed using marginalization background $\mathcal{P}$. Let $\Phi_j(t)$ denote the Federated SAGE value, where each client computes SAGE for feature j using its local background distribution $\mathcal{D}_i$, and $\phi_j^{\text{cent}}(t)$ denote the Centralized SAGE value, computed using the global mixture background $\mathcal{D}_{\text{cent}} = \sum_i w_i \mathcal{D}_i$, both evaluated on the same estimation set. *Then:*

$$|\Phi_j(t) - \phi_j^{\text{cent}}(t)| \leq L_j \sum_{i=1}^{K} w_i \, d(\mathcal{D}_i, \, \mathcal{D}_{\text{cent}})$$

Here, L_j reflects how sensitive the SAGE attribution for feature j is to changes in the marginalization background, and $d(\mathcal{D}_i, \mathcal{D}_{\text{cent}})$ quantifies the heterogeneity between each client's background and the system mixture. We analyze in Appendix D.1 how this approximation error depends on the statistical distance between client distributions and their global mixture.

3.3 Drift Detection and Diagnosis

Drift detection algorithms broadly fall into three categories: (a) error rate-based, (b) data distribution-based, and (c) multiple hypothesis testing-based methods [26]. Among these, error rate-based approaches, centered on monitoring changes in online error rates, are widely used for their simplicity and model-agnostic nature. In FL, they translate to tracking global loss or accuracy, but such aggregate signals can mask client-specific drifts due to client heterogeneity. To highlight the limitations of aggregate monitoring in FL, we simulate a concept drift scenario on the *Hyperplane* dataset [21], where the importance of feature x_1 increases for a subset of clients after round 20. Despite this induced drift,

Figure 2 (bottom-left) shows that the aggregated SAGE value of x_1 remains stable, creating a misleading impression that the feature importance is unchanged. However, a closer examination of client-level SAGE values (right panels) reveals a different story: while a non-drifted client (top-right) exhibits a sharp *decrease* in feature x_1 attribution post-drift, a drifted client (bottom-right) shows a clear *increase*.

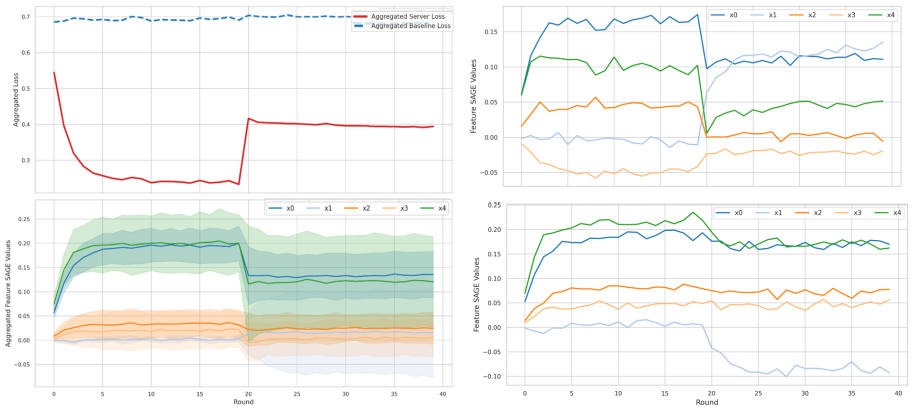

Fig. 2: **System-level vs. client-level SAGE analysis on the Hyperplane dataset.** *Top-left*: Aggregated server and baseline losses. *Bottom-left*: Aggregated SAGE values with variance (shaded); although a drift occurs at round 20, system-level SAGE values for x_1 remain stable, while variance reveals client disagreement. *Right*: Client-level SAGE values for a non-drifted client (top) and a drifted client (bottom), showing opposing shifts in x_1 attribution that cancel out in the aggregate mean.

To address this, we propose a distribution-based drift detection and diagnosis framework (Algorithm 2) that leverages Wasserstein distance to monitor client-level metrics in FL. Instead of relying on aggregate statistics or tracking individual clients across rounds, our approach pools per-client values at each round and examines how their overall distribution changes over time. This approach allows us to detect distributional shifts in client behavior without needing to link or identify clients. Wasserstein distance is well-suited for this purpose, as it handles small sample sizes and non-overlapping distributions more robustly than divergence-based alternatives such as KL or Jensen–Shannon divergence [13].

Relative Drift Score (RDS). Our detection statistic measures how much the current round's cross-client distribution deviates from recent history. For a per-client quantity $X_i(t)$ at round t (e.g., server loss or a SAGE attribution), we collect the values from the past w rounds into a flat reference multiset $\mathcal{R}(t) = \{X_i(r) : i \in [K], r = t-w, \ldots, t-1\}$ of size wK, discarding client identity. We

then compare this reference against the K values observed in the current round, $\mathcal{T}(t) = \{X_1(t), \ldots, X_K(t)\}$, via the *relative drift score*:

$$\mathrm{RDS}_X(t) = \frac{W_1^{\mathbf{w}}\big(\mathcal{R}(t),\, \mathcal{T}(t)\big)}{\mathrm{Std}_{\mathbf{w}}\big(\mathcal{R}(t)\big) + \epsilon}, \tag{6}$$

where $W_1^{\mathbf{w}}$ denotes the 1-Wasserstein distance computed over client-weighted empirical distributions, with weights $w_i = n_i / \sum_{k=1}^{K} n_k$ proportional to each client's dataset size n_i. The denominator uses the correspondingly weighted standard deviation $\mathrm{Std}_{\mathbf{w}}$, and $\epsilon > 0$ is a small constant for numerical stability. Normalizing by the reference standard deviation renders the score scale-free and comparable across quantities with different magnitudes. The weighting ensures that clients with more data contribute proportionally more to the distributional comparison, reflecting their greater statistical reliability.

Calibration and detection (RDS-Detect). Detection follows a standard *Statistical Process Control* (SPC) protocol with two phases:

- **Phase I (calibration).** During a warm-up and calibration window $[t^0_{\mathrm{cal}}, t^1_{\mathrm{cal}}]$, the detector computes RDS at each round to estimate the null-distribution parameters $\hat{\mu}_0$ and $\hat{\sigma}_0$. No alarms are raised.
- **Phase II (monitoring).** For all subsequent rounds $t > t^1_{\mathrm{cal}}$, the threshold is fixed at

$$\tau = \hat{\mu}_0 + k\,\hat{\sigma}_0, \qquad k = \Phi^{-1}\big(1 - \tfrac{p}{M}\big), \tag{7}$$

  where Φ^{-1} is the standard normal quantile function, p is the family-wise error rate (FWER) target (e.g. $p = 0.05$), and $M = T - t^1_{\mathrm{cal}}$ is the number of monitoring rounds. This Bonferroni correction ensures that the probability of *any* false alarm across the entire monitoring horizon is at most p.

Two additional guards reduce false positives. First, a detection streak may only begin at a round where the aggregated server loss increased, i.e. $\mathcal{L}^{\mathrm{server}}(t) > \mathcal{L}^{\mathrm{server}}(t-1)$, preventing alarms during natural model improvement. Second, we require k_{cons} *consecutive* threshold exceedances before triggering; at the start of a streak the reference window is *frozen* at its pre-drift state so that subsequent RDS values are always compared against the same baseline. If any round falls below τ, the streak resets and the window unfreezes.

Theorem 3 (Bonferroni Control of False Positives). *Let M be the number of monitoring rounds and set the RDS threshold as $\tau = \hat{\mu}_0 + k\,\hat{\sigma}_0$, where $k = \Phi^{-1}(1 - p/M)$. Under the null hypothesis (no drift), the probability of at least one false positive across all M rounds satisfies $\mathbb{P}(\text{any false alarm}) \le p$.*

The proof is provided in Appendix C, using a union bound (Bonferroni inequality) over the M monitoring rounds.

Algorithm 2 RDS-Detect: Distributional Drift Detection and Diagnosis

Input: Per-client losses $\{\mathcal{L}_i^{\text{server}}(t)\}$, SAGE values $\{\phi_{i,j}(t)\}$; window w; calibration interval $[t_{\text{cal}}^0, t_{\text{cal}}^1]$; FWER target p; consecutive count k_{cons}
Phase I: Compute $\text{RDS}_{\mathcal{L}^{\text{server}}}(t)$ for $t \in [t_{\text{cal}}^0, t_{\text{cal}}^1]$; estimate $\hat{\mu}_0, \hat{\sigma}_0$; set τ via (7)
for each monitoring round $t > t_{\text{cal}}^1$ **do** ▷ Phase II
    Compute $\text{RDS}_{\mathcal{L}^{\text{server}}}(t)$ via (6)
    if RDS $\geq \tau$ **and** $\mathcal{L}^{\text{server}}(t) > \mathcal{L}^{\text{server}}(t-1)$ (or streak active) **then**
        Increment consecutive counter; freeze reference window if new streak
        if k_{cons} consecutive exceedances reached **then**
            Trigger drift alarm
            Diagnose: Compute $\text{RDS}_{\phi_j}(t)$ for each feature j; rank by descending RDS
        end if
    else
        Reset consecutive counter; unfreeze reference window
    end if
end for

Drift diagnosis. Since the RDS statistic is defined for any per-client quantity, it naturally extends from loss-based detection to feature-level diagnosis. Once a drift alarm is triggered, we compute $\text{RDS}_{\phi_j}(t)$ for each feature j using the SAGE attributions $\{\phi_{i,j}(t)\}_{i=1}^K$ and rank features by descending RDS. Features with the largest distributional shift in their attributions are identified as the primary drivers of the detected drift.

Deferred SAGE computation. To reduce runtime and communication overhead, we defer full SAGE attribution computation until drift detection is triggered. Each client maintains a lightweight cache for the past w rounds, storing only the sufficient statistics required for SAGE estimation. When $\text{RDS}_{\mathcal{L}^{\text{server}}}(t) \geq \tau$, clients reconstruct SAGE values for the cached rounds on demand, avoiding redundant forward passes. We provide a theoretical and empirical analysis of runtime and memory usage in Appendix B.

4 Experiments and Results

We evaluate Federated SAGE and our drift detection framework across six datasets spanning synthetic and real-world domains. The synthetic datasets are Hyperplane [21] and Agrawal Stream [2], which offer controllable drift mechanisms (coefficient changes and classification function switches, respectively). The real-world datasets are PIMA Indian Diabetes [39], Wine Quality [12], FedHeart *(cite this paper in parenthesis)*, and Credit-G *(cite this paper in parenthesis)*, where drift is introduced through feature noise injection or label manipulation. Together, these datasets allow us to evaluate detection and diagnosis under *sudden* and *gradual* drift patterns with varying severity and proportions of affected clients. Detailed dataset descriptions, drift configurations, and model hyperparameters are provided in Appendix A.

4.1 Federated SAGE Approximation of Centralized SAGE

We evaluate how well Federated SAGE (aggregated from client-level SAGE values) approximates Centralized SAGE (computed on pooled data) across six datasets. We measure Mean Absolute Error (MAE) against centralized SAGE, Spearman rank correlation (ρ), and the maximum per-client runtime, which determines the wall-clock cost of each federated round since the server must wait for the slowest client.

Efficient SAGE via background compression. Computing SAGE requires marginalizing unobserved features using a *background distribution* and evaluating the model on an *estimation (foreground) set*. Since SAGE's cost scales with the number of background samples, naively using the full local dataset as the background can be prohibitively expensive on resource-constrained edge devices. To reduce this cost, we compress the background distribution using coreset methods [3] like compress++ [38], which reduce n local samples to $\mathcal{O}(\sqrt{n})$ representative points while preserving distributional structure.

We compare three strategies: (1) **IID**: IID-sampled background (4× the compressed size) with the full estimation set, representing the standard approach; (2) **Comp bg**: compressed background with the full estimation set, reducing background cost while retaining full evaluation fidelity; and (3) **Comp fg**: compressed background *and* compressed foreground (estimation) set, the most lightweight variant.

Results. Table 1 and Figure 3 summarize the results. Across all six datasets, IID and Comp bg achieve comparable MAE (typically < 0.01) and high Spearman correlations ($\rho > 0.9$) on Credit, FedHeart, and Hyperplane, confirming that compressed backgrounds closely match IID sampling in approximation quality. Runtimes for IID and Comp bg are also similar on most datasets, with the exception of Wine Quality where Comp bg achieves a notable speedup (13 s vs. 28 s). The fully compressed variant (Comp fg) is consistently the fastest (e.g., Diabetes: 0.3 s vs. 3.5 s for IID) but incurs substantially higher MAE and lower rank correlations, particularly on datasets with many features or small client samples (e.g., Wine: $\rho = 0.06$; Credit: $\rho = 0.33$). A likely explanation is that aggressive foreground compression discards samples that are important for accurate SAGE estimation, and this effect may be amplified in non-IID federated settings where client-level background sets are already biased, making small compressed summaries less representative of the true data distribution. Overall, **Comp bg offers the best accuracy–efficiency tradeoff**: it preserves nearly the same approximation quality as IID sampling while reducing the background set size, making it well-suited for federated deployments on resource-constrained devices.

4.2 Drift Detection Evaluation

Several works have explored drift detection in federated learning, but primarily at the **client level** [9, 28, 34], whereas our approach operates at the *system*

Table 1: Federated SAGE approximation quality (IID setting, no drift): MAE vs. centralized SAGE, Spearman ρ, and max per-client runtime (s). IID: IID-sampled background + full estimation set; Comp bg: compressed background + full estimation set; Comp fg: compressed background + compressed foreground set. Std in parentheses; best MAE and ρ per row in bold. Mean over 5 seeds.

	MAE			ρ			Max time (s)		
Dataset	IID	Comp bg	Comp fg	IID	Comp bg	Comp fg	IID	Comp bg	Comp fg
Hyperplane	**0.005 (0.002)**	0.006 (0.002)	0.021 (0.007)	0.96 (0.05)	**0.98 (0.04)**	0.84 (0.08)	1.1	**0.6**	5.7
Wine	**0.011 (0.001)**	0.011 (0.001)	0.020 (0.004)	**0.51 (0.19)**	0.43 (0.13)	0.06 (0.29)	28	13	**1.1**
Diabetes	**0.007 (0.001)**	0.008 (0.001)	0.020 (0.002)	**0.59 (0.31)**	0.53 (0.26)	0.35 (0.19)	3.5	3.6	**0.3**
Credit	**0.002 (0.000)**	0.002 (0.000)	0.012 (0.003)	0.92 (0.02)	**0.92 (0.01)**	0.33 (0.10)	6.6	4.5	**1.1**
FedHeart	0.007 (0.001)	**0.007 (0.000)**	0.026 (0.009)	0.92 (0.03)	**0.94 (0.04)**	0.33 (0.27)	7.3	7.0	**0.6**
Agrawal	**0.002 (0.000)**	0.003 (0.001)	0.010 (0.003)	0.84 (0.08)	**0.85 (0.02)**	0.75 (0.17)	2.5	**1.4**	2.3

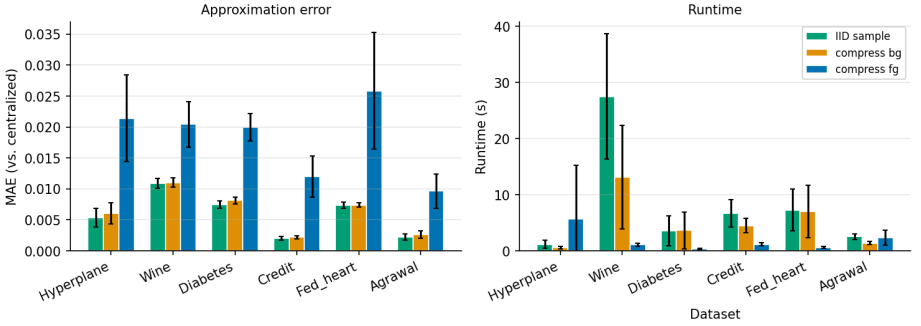

Fig. 3: **Federated SAGE approximation error (left) and per-client runtime (right)** across six datasets under IID conditions. Comp bg closely matches IID sampling in MAE while reducing runtime, whereas the fully compressed variant (Comp fg) is fastest but incurs notably higher error.

level. Other studies propose adaptation mechanisms to mitigate drift [8, 33, 24, 41, 22], but do not provide explicit drift detection methods. To benchmark our framework, we compare RDS-based detection against four widely used change detection methods applied to the aggregated system loss: CUSUM [32], Page-Hinkley (PH) [32, 37], ADWIN *(cite this paper in parenthesis)*, and KSWIN *(cite this paper in parenthesis)*. All methods are evaluated across six dataset–drift combinations spanning sudden and gradual drift patterns.

Table 2 reports F1-score and average detection delay. RDS achieves near-perfect detection (avg. F1 $= 0.97$) across 5 seeds per dataset with the lowest delay (1.9 rounds) and is the only method with F1 ≥ 0.80 on every dataset. The strongest baseline, CUSUM (avg. F1 $= 0.78$), matches RDS on three datasets but drops to 0.40 on Credit. The window-based methods ADWIN and KSWIN are least consistent (avg. F1 of 0.28 and 0.14) and incur delays exceeding 10 rounds on average. Overall, RDS provides robust system-level detection regardless of

Table 2: Drift detection performance: F1-score and average detection delay (rounds) across datasets ($n = 5$ seeds). Best F1 and delay per row in **bold**; "–" indicates no true positives (delay undefined).

	F1-score					Avg. Detection Delay				
Dataset	RDS	CUSUM	PH	ADWIN	KSWIN	RDS	CUSUM	PH	ADWIN	KSWIN
Agrawal (Sudden)	**1.00**	**1.00**	0.80	0.50	0.00	**0.0**	**0.0**	**0.0**	10.0	—
Wine (Sudden)	**1.00**	**1.00**	0.60	0.67	0.50	**0.0**	0.2	0.3	2.0	6.0
FedHeart (Sudden)	**1.00**	0.67	0.44	0.00	0.00	3.0	**2.7**	3.5	—	—
Hyperplane (Gradual)	**1.00**	**1.00**	0.60	0.50	0.00	**0.4**	1.0	1.3	22.5	—
Diabetes (Gradual)	**1.00**	0.60	0.60	0.00	0.33	**2.0**	5.0	5.0	—	14.0
Credit (Sudden)	**0.80**	0.40	0.40	0.00	0.00	**5.8**	9.0	9.0	—	—
Avg.	**0.97**	0.78	0.57	0.28	0.14	**1.9**	3.0	3.2	11.5	10.0

drift type, whereas classical methods are sensitive to hyperparameters and often miss subtle distributional shifts.

Figure 4 illustrates RDS-based detection on four datasets. The RDS score stays near zero during calibration and spikes after drift onset: immediately for sudden drift (FedHeart, Wine) and gradually over the transition window for gradual drift (Diabetes, Hyperplane). Even when the loss change is subtle (e.g., Diabetes), RDS amplifies the distributional shift, highlighting its sensitivity over point-estimate methods.

4.3 Drift Diagnosis Evaluation

Once drift is detected, the diagnosis module ranks features by their contribution to the shift. We evaluate three attribution methods—SAGE (global model-loss decomposition), PFI (permutation feature importance) *(cite this paper in parenthesis)*, and SHAP (prediction value function) *(cite this paper in parenthesis)*—combined with two ranking strategies: (i) **RDS-based**, which applies the Wasserstein-based RDS statistic (Eq. 6) to per-client feature attributions, and (ii) **Delta-based**, which ranks features by the absolute change in mean attribution before and after the detected drift point.

Table 3 reports Mean Reciprocal Rank (MRR) for identifying the ground-truth drifted feature(s) across six datasets. Delta-based methods consistently outperform their RDS-based counterparts, with **Delta-SHAP** (avg. MRR = 0.85) and **Delta-SAGE** (avg. MRR = 0.81) achieving the highest scores overall. Both attain perfect or near-perfect rankings on most datasets, with the exception of Credit-G, which remains challenging for all methods. Among RDS-based methods, RDS-PFI is the strongest (avg. MRR = 0.70). These results suggest that comparing pre- and post-drift attribution means provides a more reliable ranking signal than distribution-based testing at the feature level, likely because delta ranking is less sensitive to noise in individual client attributions. Importantly, both top-performing methods (Delta-SAGE and Delta-SHAP) benefit from the federated SAGE infrastructure proposed in this work, demonstrating that our

Table 3: Drift diagnosis performance (MRR) across datasets ($n = 5$ seeds). Higher is better; best per row in **bold**. RDS-based methods rank features by Wasserstein-based distributional shift; Delta-based methods rank by absolute change in mean attribution.

	RDS-based			Delta-based		
Dataset	SAGE	PFI	SHAP	SAGE	PFI	SHAP
Agrawal (Sudden)	**1.00**	0.80	0.70	**1.00**	**1.00**	0.70
Wine (Sudden)	**1.00**	**1.00**	**1.00**	**1.00**	0.42	**1.00**
FedHeart (Sudden)	0.50	0.60	0.67	**1.00**	0.87	**1.00**
Hyperplane (Gradual)	0.47	0.57	0.71	0.64	0.67	**1.00**
Diabetes (Gradual)	0.54	0.90	0.47	0.87	**1.00**	0.90
Credit (Sudden)	0.46	0.35	0.44	0.33	0.15	**0.52**
Avg.	0.66	0.70	0.67	0.81	0.69	**0.85**

framework supports effective diagnosis regardless of which specific method is chosen.

Limitations While our proposed framework advances the understanding of concept drift in FL, several limitations remain and open avenues for future research. Calculating feature importance scores such as SAGE can be computationally intensive, especially for large feature spaces. This poses practical challenges for deployment on resource-constrained edge devices. Furthermore, while our method focuses on drift detection and interpretation, future extensions could explore its integration with drift adaptation mechanisms, enabling not only detection but also automated responses to mitigate performance degradation in FL systems. Finally, although our approach avoids linking attributions to specific clients, sharing distributional information may still risk exposing sensitive data patterns, highlighting the need for privacy-preserving extensions such as differential privacy or secure aggregation.

5 Conclusion

We present a novel framework for detecting and interpreting concept drift in Federated Learning by extending SAGE to a federated setting and combining it with distribution-based drift detection. Our approach provides feature-level attributions of global model performance and identifies drifted features through analysis of client-level distributions, enabling fine-grained understanding of drift without compromising data privacy. Through extensive evaluations on synthetic and real-world datasets, we demonstrated that our method effectively detects and explains various types of concept drift. This work paves the way for more interpretable and robust FL systems, with future extensions focusing on privacy-preserving mechanisms and integration with drift adaptation strategies.

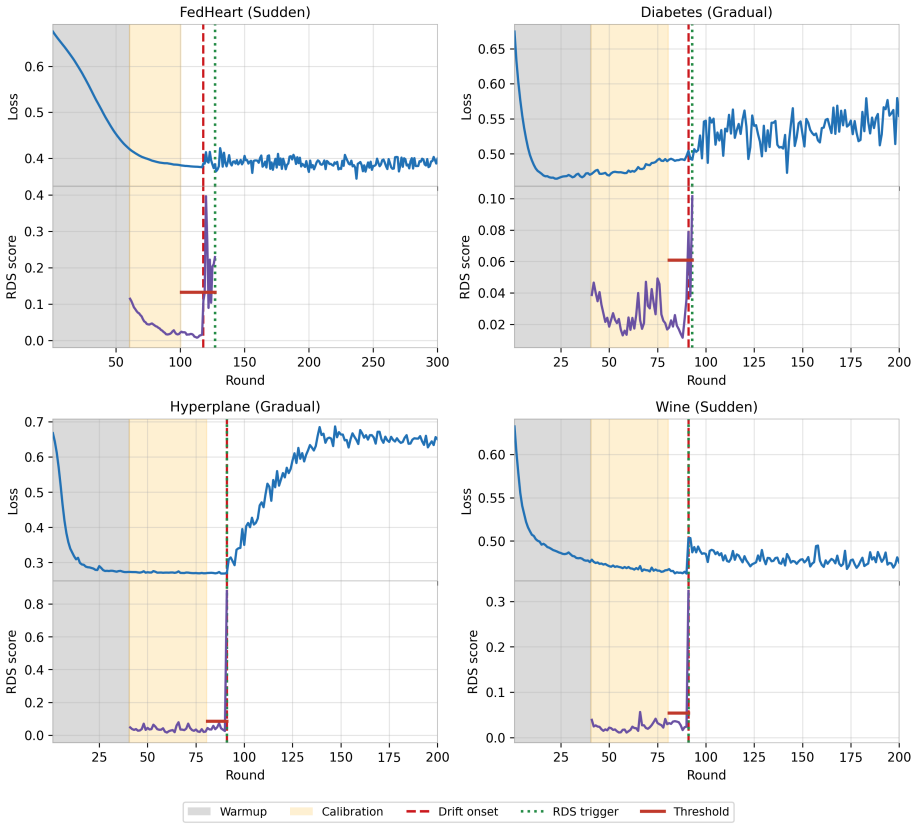

Fig. 4: **System loss and RDS score over federated rounds** for four representative datasets. Each panel shows the aggregated loss (top) and the corresponding RDS score (bottom). Shaded regions indicate warmup (gray) and calibration (yellow) phases. The dashed red line marks drift onset; the dotted green line marks the round at which RDS triggers detection; the solid horizontal line shows the detection threshold. RDS responds promptly after drift onset across both sudden (FedHeart, Wine) and gradual (Diabetes, Hyperplane) drift patterns.

Supplementary Material for: Explainable Federated Learning: Detecting Concept Drift with Feature-Level Attribution Using SAGE

The supplementary material is structured as follows. Section A provides additional details on our experimental setup, including dataset descriptions, experiment configurations, and model and explainer hyperparameters. In Section B, we analyze the computational complexity of Federated SAGE and RDS-Detect. Section C presents the proofs of the theorems stated in the main paper.

A Experiment Details

A.1 Datasets

We use six datasets spanning synthetic and real-world domains. The **Hyperplane** [21] dataset is a synthetic data stream with controllable feature weights, commonly used in streaming and online learning research. The **Agrawal Stream** [2] dataset produces tabular data for binary classification based on complex logical functions of numerical and categorical features. The **PIMA Indian Diabetes** [39] dataset contains medical diagnostic records for predicting diabetes onset. The **Wine Quality** [12] dataset consists of physicochemical properties of wine samples with expert-rated quality scores. The **FedHeart** dataset *(cite this paper in parenthesis)* provides clinical features for heart disease prediction. The **Credit-G** dataset *(cite this paper in parenthesis)* contains financial attributes for credit risk classification.

A.2 Experiment Configurations

Table 4 summarizes the experimental configurations used across our detection and diagnosis experiments. Each experiment targets a specific drift pattern (sudden or gradual) with a dataset-specific drift mechanism and known ground-truth drifted features, enabling controlled evaluation of both detection and feature-level diagnosis.

Table 4: Experiment configurations for drift detection and diagnosis. Each row defines a dataset–drift-type combination with the corresponding drift mechanism, drifted feature(s), number of clients, and FL training rounds.

Dataset	Drift Type	Drift Mechanism	Drifted Feature(s)	# Clients	# Rounds	Seeds
Agrawal	Sudden	Concept switch (fn 0→1)	salary, commission, age	10	200	5
Wine Quality	Sudden	Feature noise (alcohol)	alcohol	5	200	5
FedHeart	Sudden	Feature noise (cp)	cp	4	300	5
Credit-G	Sudden	Feature noise (duration)	duration	5	200	5
Hyperplane	Gradual	Coefficient change (x_0)	x_0	10	200	5
Diabetes	Gradual	Feature noise (Glucose)	Glucose	6	200	5

Drift is injected at a randomized onset round t_0 (drawn uniformly within a predefined range after a calibration period). For gradual drift, a transition window is used over which the drift is incrementally introduced. All results are aggregated over 5 random seeds.

A.3 Model and Explainer Hyperparameters

The model used in all experiments is a lightweight fully connected neural network consisting of three linear layers with ReLU activations: an input layer with 128 units, a hidden layer with 64 units, and an output layer matching the number of classes. This architecture balances expressiveness and computational efficiency for federated FL experiments. Each client trains this model using stochastic gradient descent (SGD) with a learning rate of 0.05 and momentum of 0.5, running for 40 communication rounds with a batch size of 32. Clients split their local data into 80% for training and 20% for testing. Local training is performed for a single epoch per communication round, reflecting typical low-resource FL scenarios.

For feature attribution, we adopt unbiased KernelSHAP [14] with marginal sampling for imputing masked features as the approximation method to estimate SAGE values [15]. We selected Unbiased KernelSHAP instead of PermutationSHAP because it offers stronger statistical guarantees and demonstrated more reliable empirical convergence in our federated learning experiments. In preliminary experiments, we observed that PermutationSHAP failed to converge in certain settings, whereas KernelSHAP consistently provided reliable estimates across all tested configurations. For each client, SAGE values are computed using an estimation set comprising at least 30% of the client's test data, and the background set for unbiased KernelSHAP is drawn from train data, with size equal to the estimation set. This configuration ensures stable estimation while maintaining reasonable computational overhead in federated settings.

All federated learning experiments were implemented using the Python library `Flower` [4], and SAGE values were estimated using the `sage-importance` package [15].

B Runtime Complexity

Runtime Complexity of Federated SAGE The runtime of Federated SAGE is dominated by the cost of computing local SAGE values on each client before aggregation. Given K clients, each with an estimation dataset of size n_i, a background dataset of size m_i, and a model with forward pass cost C_f, evaluating a single coalition costs:

$$O(n_i \cdot m_i \cdot C_f).$$

The algorithm samples k coalitions, leading to a per-client runtime of:

$$T_{\text{client}} = O(k \cdot n_i \cdot m_i \cdot C_f) + O(d^3),$$

where the $O(d^3)$ term arises from solving a dense $d \times d$ linear system at the final regression step. Overall, Federated SAGE scales across clients as:

$$T_{\text{Federated SAGE}} = O\left(\sum_{i=1}^{K} \left(k \cdot n_i \cdot m_i \cdot C_f + d^3\right)\right),$$

with linear dependence on the number of coalition samples k, dataset sizes n_i and m_i, the number of features d, and the number of clients K, while being easily parallelizable across participants.

Runtime Complexity of RDS-Detect RDS-Detect operates in two stages. In Stage 1, it monitors the aggregated loss values reported by all K clients and computes the 1D Wasserstein distance between their current and previous loss distributions. This calculation costs:

$$O(K \log K),$$

Since computing the Wasserstein distance in one dimension is dominated by a sorting step over the K client loss values. Stage 2 is triggered only when Stage 1 detects a potential drift and performs feature-level drift detection across all d features, adding:

$$p_{\text{trigger}} \cdot O(d \cdot K \log K)$$

to the runtime, where p_{trigger} is the probability that Stage 2 is activated.

In addition, RDS-Detect requires SAGE feature attribution values for all clients over the last $w+1$ rounds. These SAGE values are computed only when Stage 1 triggers, leading to an expected computational cost of:

$$p_{\text{trigger}} \cdot (w+1) \cdot T_{\text{Federated SAGE}},$$

where $T_{\text{Federated SAGE}}$ is defined in the previous section.

Thus, the overall expected runtime of RDS-Detect is:

$$T_{\text{RDS,total}} = p_{\text{trigger}} \cdot \Big[(w+1) \cdot T_{\text{Federated SAGE}} + O(d \cdot K \log K)\Big] + O(K \log K).$$

Figure 5 shows that runtime grows approximately linearly with window size w, as predicted by the complexity analysis.

Note that RDS-Detect also reads the per-client loss values in each round to compute Stage 1 statistics. However, these losses are already computed as part of the standard federated training process and do not require additional inference or data transfer. Therefore, their computation cost is not counted as an extra overhead in the runtime complexity.

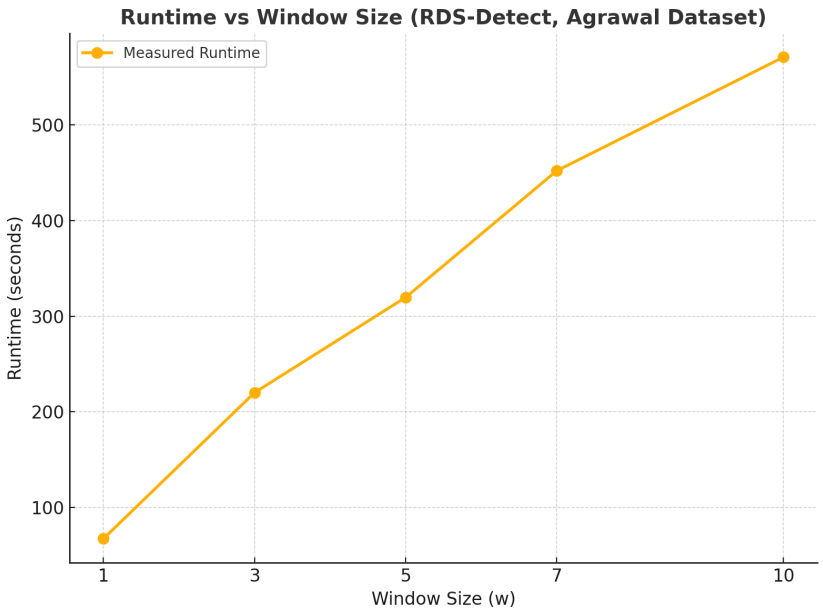

Fig. 5: Measured runtime of RDS-Detect (in seconds) as a function of the sliding window size w on the Agrawal dataset. The results show that runtime increases approximately linearly with w

Memory Complexity of RDS-Detect RDS-Detect requires each client to maintain a sliding buffer of information from the past w rounds to enable recomputation of feature attribution values when Stage 1 drift detection is triggered. This buffer consists of:

- **Estimation datasets:** $O(n \cdot w)$, where n is the number of samples used to estimate SAGE values per round.
- **Background datasets:** $O(m \cdot w)$, where m is the number of background samples drawn for imputations in KernelSHAP.
- **Model snapshots:** $O(S_{\text{model}} \cdot w)$, where S_{model} denotes the memory required to store one instance of the global model (weights and optimizer state).

The total client-side memory footprint is therefore:

$$M_{\text{client}} = O\big(w(n + m + S_{\text{model}})\big).$$

This design is consistent with the federated learning paradigm, where computation and temporary storage are handled by client devices to minimize server-side resource demands and preserve data locality. Memory usage grows linearly with the sliding window size w, and is dominated by model checkpoint storage when models are large relative to cached sample data.

C Proofs

C.1 Proof of Federated SAGE Loss Decomposition (Proposition 1)

Proof. Assume that all client weights satisfy $w_i \geq 0$ and $\sum_{i=1}^{K} w_i = 1$. Recall the definition of the aggregated feature attributions:

$$\sum_{j=1}^{d} \Phi_j(t) = \sum_{j=1}^{d} \sum_{i=1}^{K} w_i \phi_{i,j}^{(t)} = \sum_{i=1}^{K} w_i \sum_{j=1}^{d} \phi_{i,j}^{(t)}.$$

By the local SAGE decomposition for each client i (Eq. (3) in the main text),

$$\sum_{j=1}^{d} \phi_{i,j}^{(t)} = \mathcal{L}_i^{\text{baseline}}(t) - \mathcal{L}_i^{\text{server}}(t).$$

Substituting this into the previous expression yields:

$$\sum_{i=1}^{K} w_i \big(\mathcal{L}_i^{\text{baseline}}(t) - \mathcal{L}_i^{\text{server}}(t)\big) = \mathcal{L}^{\text{baseline}}(t) - \mathcal{L}^{\text{server}}(t),$$

where the last step follows from the definition of system-level aggregation (Eq. (4) and (5) in the main text). Thus, the decomposition holds under the stated weight assumptions.

C.2 Proof of Unbiasedness and Variance of Federated SAGE (Proposition 2)

Proof. Each client i computes $\hat{\phi}_{i,j}(t)$, the unbiased SAGE estimator for feature j at round t, so $\mathbb{E}[\hat{\phi}_{i,j}(t)] = \phi_{i,j}(t)$. The system aggregates as

$$\hat{\Phi}_j(t) = \sum_{i=1}^{K} w_i \hat{\phi}_{i,j}(t).$$

By linearity of expectation,

$$\mathbb{E}\left[\hat{\Phi}_j(t)\right] = \sum_{i=1}^{K} w_i \mathbb{E}\left[\hat{\phi}_{i,j}(t)\right] = \sum_{i=1}^{K} w_i \phi_{i,j}(t) = \Phi_j(t).$$

For variance:

$$\text{Var}[\hat{\Phi}_j(t)] = \text{Var}\left(\sum_{i=1}^{K} w_i \hat{\phi}_{i,j}(t)\right)$$

Assuming independence,

$$= \sum_{i=1}^{K} w_i^2 \ \text{Var}[\hat{\phi}_{i,j}(t)].$$

C.3 Proof of IID Consistency of Federated SAGE (Theorem 1)

Proof. If all clients have the same data distribution $\mathcal{D}_i = \mathcal{D}$, and all feature attributions are computed (with enough data and MC samples), then by the law of large numbers (for $n_i \to \infty$), local SAGE estimators converge to the population value: $\phi_{i,j}(t) \to \phi_j(\mathcal{D}, t)$ for all i. The aggregation thus yields:

$$\lim_{n_i \to \infty} \hat{\Phi}_j(t) = \sum_{i=1}^{K} w_i \phi_j(\mathcal{D}, t) = \phi_j(\mathcal{D}, t),$$

since $\sum_{i=1}^{K} w_i = 1$. At the same time, the centralized SAGE $\phi_j^{\text{cent}}(t)$ is calculated on the pooled data, i.e., on $\mathcal{D}$, so

$$\lim_{n \to \infty} \phi_j^{\text{cent}}(t) = \phi_j(\mathcal{D}, t).$$

Therefore, both limits match, and federated and centralized SAGE are equivalent in the IID regime.

C.4 Proof of Non-IID Heterogeneity Gap Bound (Theorem 2)

Proof. Let $\mathcal{D}_{\text{cent}} = \sum_i w_i \mathcal{D}_i$ denote the weighted mixture distribution, where $w_i \geq 0$ and $\sum_i w_i = 1$. Define $F(\mathcal{D}) = \phi_j(\mathcal{D}, t)$. We can write:

$$|\Phi_j(t) - \phi_j^{\text{cent}}(t)| = \Big| \sum_i w_i F(\mathcal{D}_i) - F(\mathcal{D}_{\text{cent}}) \Big| = \Big| \sum_i w_i \big(F(\mathcal{D}_i) - F(\mathcal{D}_{\text{cent}}) \big) \Big|.$$

Applying the triangle inequality gives:

$$\leq \sum_i w_i \, |F(\mathcal{D}_i) - F(\mathcal{D}_{\text{cent}})|.$$

Assumption (Lipschitz Continuity): We assume that F is Lipschitz continuous with respect to a distance metric $d(\cdot, \cdot)$ on probability measures, i.e.,

$$|F(\mathcal{D}_1) - F(\mathcal{D}_2)| \leq L_j \, d(\mathcal{D}_1, \mathcal{D}_2) \quad \forall \mathcal{D}_1, \mathcal{D}_2.$$

Using this property for each term:

$$|\Phi_j(t) - \phi_j^{\text{cent}}(t)| \leq L_j \sum_i w_i \, d(\mathcal{D}_i, \mathcal{D}_{\text{cent}}).$$

This establishes the bound.

C.5 Proof of Bonferroni Control of False Positives (Theorem 3)

Proof. Let M denote the number of monitoring rounds and let $\text{RDS}(t)$ be the score at round t. Under the null hypothesis (no drift), we model $\text{RDS}(t)$ as approximately Gaussian with mean μ_0 and standard deviation σ_0, estimated

during calibration as $\hat{\mu}_0$ and $\hat{\sigma}_0$. The threshold is set as $\tau = \hat{\mu}_0 + k\,\hat{\sigma}_0$ with $k = \Phi^{-1}(1 - p/M)$.

For each individual round, the per-round false alarm probability is:

$$\mathbb{P}\big(\mathrm{RDS}(t) \geq \tau \mid H_0\big) = 1 - \Phi(k) = \frac{p}{M}.$$

Applying the union bound (Bonferroni inequality) over the M monitoring rounds:

$$\mathbb{P}\big(\exists\, t : \mathrm{RDS}(t) \geq \tau \mid H_0\big) \leq \sum_{t=1}^{M} \frac{p}{M} = p.$$

Thus the family-wise error rate (FWER) is controlled at level p.

Bibliography

[1] Adams, J.N., van Zelst, S.J., Quack, L., Hausmann, K., van der Aalst, W.M., Rose, T.: A framework for explainable concept drift detection in process mining. In: Business Process Management: 19th International Conference, BPM 2021, Rome, Italy, September 06–10, 2021, Proceedings 19. pp. 400–416. Springer (2021)

[2] Agrawal, R., Imielinski, T., Swami, A.: Database mining: a performance perspective. IEEE Transactions on Knowledge and Data Engineering **5**(6), 914–925 (1993)

[3] Baniecki, H., Casalicchio, G., Bischl, B., Biecek, P.: Efficient and accurate explanation estimation with distribution compression. arXiv preprint arXiv:2406.18334 (2024)

[4] Beutel, D.J., Topal, T., Mathur, A., Qiu, X., Fernandez-Marques, J., Gao, Y., Sani, L., Li, K.H., Parcollet, T., de Gusmão, P.P.B., et al.: Flower: A friendly federated learning research framework. arXiv preprint arXiv:2007.14390 (2020)

[5] Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 2007 SIAM international conference on data mining. pp. 443–448. SIAM (2007). https://doi.org/10.1137/1.9781611972771.42

[6] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečnỳ, J., Mazzocchi, S., McMahan, B., et al.: Towards federated learning at scale: System design. Proceedings of machine learning and systems **1**, 374–388 (2019)

[7] Bárcena, J.L.C., Daole, M., Ducange, P., Marcelloni, F., Renda, A., Ruffini, F., Schiavo, A.: Fed-xai: Federated learning of explainable artificial intelligence models. In: Proceedings of XAI. it 2022 Italian Workshop on Explainable Artificial Intelligence 2022. pp. 1–14 (2022)

[8] Canonaco, G., Bergamasco, A., Mongelluzzo, A., Roveri, M.: Adaptive federated learning in presence of concept drift. In: Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–7 (2021). https://doi.org/10.1109/IJCNN52387.2021.9533710

[9] Casado, F.E., Lema, D., Criado, M.F., Iglesias, R., Regueiro, C.V., Barro, S.: Concept drift detection and adaptation for federated and continual learning. Multimedia Tools and Applications pp. 1–23 (2022)

[10] Chen, Y., Chai, Z., Cheng, Y., Rangwala, H.: Asynchronous federated learning for sensor data with concept drift. In: 2021 IEEE International Conference on Big Data (Big Data). pp. 4822–4831. IEEE (2021)

[11] Corbucci, L., Guidotti, R., Monreale, A.: Explaining black-boxes in federated learning. In: World Conference on Explainable Artificial Intelligence. pp. 151–163. Springer (2023)

[12] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decision support systems **47**(4), 547–553 (2009)
[13] Courty, N., Flamary, R., Tuia, D., Rakotomamonjy, A.: Optimal transport for domain adaptation. IEEE transactions on pattern analysis and machine intelligence **39**(9), 1853–1865 (2016)
[14] Covert, I., Lee, S.I.: Improving kernelshap: Practical shapley value estimation via linear regression. arXiv preprint arXiv:2012.01536 (2020)
[15] Covert, I., Lundberg, S.M., Lee, S.I.: Understanding Global Feature Contributions With Additive Importance Measures. In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020). pp. 17212–17223 (2020)
[16] Ducange, P., Marcelloni, F., Renda, A., Ruffini, F.: Federated learning of xai models in healthcare: A case study on parkinson's disease. Cognitive Computation **16**(6), 3051–3076 (2024). https://doi.org/10.1007/s12559-024-10332-x
[17] Fiosina, J.: Interpretable privacy-preserving collaborative deep learning for taxi trip duration forecasting. Communications in Computer and Information Science **1612** (2022)
[18] Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17. pp. 286–295. Springer (2004)
[19] Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client-level perspective. arXiv preprint arXiv:1712.07557 (2017)
[20] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., Ramage, D.: Federated learning for mobile keyboard prediction. arXiv preprint arXiv:1811.03604 (2018)
[21] Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 97–106. KDD '01, Association for Computing Machinery, New York, NY, USA (2001). https://doi.org/10.1145/502512.502529, https://doi.org/10.1145/502512.502529
[22] Jothimurugesan, E., Hsieh, K., Wang, J., Joshi, G., Gibbons, P.B.: Federated learning under distributed concept drift. In: International Conference on Artificial Intelligence and Statistics. pp. 5834–5853. PMLR (2023)
[23] Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al.: Advances and open problems in federated learning. Foundations and trends® in machine learning **14**(1–2), 1–210 (2021)
[24] Kang, M., Kim, S., Jin, K.H., Adeli, E., Pohl, K.M., Park, S.H.: Fednn: Federated learning on concept drift data using weight and adaptive group normalizations. Pattern Recognition **149**, 110230 (2024)
[25] Lopez-Ramos, L.M., Leiser, F., Rastogi, A., Hicks, S., Strümke, I., Madai, V.I., Budig, T., Sunyaev, A., Hilbert, A.: Interplay between federated learning and explainable artificial intelligence: a scoping review. arXiv preprint arXiv:2411.05874 (2024)

[26] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2019). https://doi.org/10.1109/TKDE.2018.2876857

[27] Lu, W., Wang, J., Chen, Y., Qin, X., Xu, R., Dimitriadis, D., Qin, T.: Personalized federated learning with adaptive batchnorm for healthcare. IEEE Transactions on Big Data (2022)

[28] Manias, D.M., Shaer, I., Yang, L., Shami, A.: Concept drift detection in federated networked systems. In: Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM). pp. 1–6 (2021). https://doi.org/10.1109/GLOBECOM46510.2021.9685083

[29] McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics. pp. 1273–1282. PMLR (2017)

[30] Muschalik, M., Fumagalli, F., Hammer, B., Hüllermeier, E.: isage: An incremental version of sage for online explanation on data streams. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 428–445. Springer (2023)

[31] Muschalik, M., Fumagalli, F., Jagtani, R., Hammer, B., Hüllermeier, E.: ipdp: on partial dependence plots in dynamic modeling scenarios. In: World Conference on Explainable Artificial Intelligence. pp. 177–194. Springer (2023)

[32] Page, E.S.: Continuous inspection schemes. Biometrika **41**(1/2), 100–115 (1954). https://doi.org/10.2307/2333009

[33] Panchal, K., Choudhary, S., Mitra, S., Mukherjee, K., Sarkhel, S., Mitra, S., Guan, H.: Flash: Concept drift adaptation in federated learning. In: International Conference on Machine Learning. pp. 26931–26962. PMLR (2023)

[34] Rahimli, L., Awaysheh, F.M., Al Zubi, S., Alawadi, S.: Federated learning drift detection: An empirical study on the impact of concept and data drift. In: Proceedings of the 2024 International Conference on Federated Learning Technologies and Applications (FLTA). pp. 241–250 (2024). https://doi.org/10.1109/FLTA63145.2024.10839814

[35] Rieke, N., Hancox, J., Li, W., Milletari, F., Roth, H.R., Albarqouni, S., Bakas, S., Galtier, M.N., Landman, B.A., Maier-Hein, K., et al.: The future of digital health with federated learning. NPJ digital medicine **3**(1), 119 (2020)

[36] Saifullah, S., Mercier, D., Lucieri, A., Dengel, A., Ahmed, S.: The privacy-explainability trade-off: unraveling the impacts of differential privacy and federated learning on attribution methods. Frontiers in Artificial Intelligence **7**, 1236947 (2024). https://doi.org/10.3389/frai.2024.1236947

[37] Sebastião, R., Fernandes, J.M.: Supporting the page-hinkley test with empirical mode decomposition for change detection. In: International symposium on methodologies for intelligent systems. pp. 492–498. Springer (2017)

[38] Shetty, A., Dwivedi, R., Mackey, L.: Distribution compression in near-linear time. arXiv preprint arXiv:2111.07941 (2021)

[39] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., Johannes, R.S.: Using the adap learning algorithm to forecast the onset of diabetes mellitus. In: Proceedings of the annual symposium on computer application in medical care. p. 261 (1988)
[40] Teixeira, R., Almeida, L., Rodrigues, P., Corona, J., Antunes, M., Aguiar, R.L.: Balancing privacy and explainability in federated learning. PREPRINT (Version 1) available at Research Square [https://doi.org/10.21203/rs.3.rs-3714454/v1] (2023)
[41] Varno, F., Saghayi, M., Rafiee Sevyeri, L., Gupta, S., Matwin, S., Havaei, M.: Adabest: Minimizing client drift in federated learning via adaptive bias estimation. In: Computer Vision – ECCV 2022, pp. 710–726. Springer Nature Switzerland (2022). https://doi.org/10.1007/978-3-031-20050-2_41
[42] Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: Concept and applications. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(2), 1–19 (2019)
[43] Zhu, H., Xu, J., Liu, S., Jin, Y.: Federated learning on non-iid data: A survey. Neurocomputing **465**, 371–390 (2021)