Système de Détection de Fraude Bancaire Distribué

Architecture Edge Computing et Apprentissage Fédéré

Département Informatique

Cheikhani : C20114

Résumé

Ce rapport présente la mise en œuvre technique et les résultats expérimentaux d'un système de détection de fraude distribué, conçu pour le secteur bancaire mauritanien. Contrairement aux approches centralisées traditionnelles, ce projet adopte une architecture **Edge Computing** couplée à l'**Apprentissage Fédéré (Federated Learning)**. Le système permet l'entraînement de modèles d'IA locaux au sein des agences (Edge Nodes) tout en préservant la confidentialité absolue des données clients. Les résultats démontrent une précision globale de **61.48%** après agrégation, validant la robustesse de l'architecture basée sur Apache Kafka et Docker.

1 Introduction et Contexte

Les institutions financières modernes (telles que Bankily ou Masrivi) traitent des millions de transactions quotidiennes. Le défi majeur consiste à détecter les fraudes en temps réel sans compromettre la confidentialité des données (Réglementations BCM) ni surcharger le réseau.

Notre solution propose un changement de paradigme : passer d'une architecture *"Code-to-Data"* (centralisée) à une architecture *"Data-to-Code"* (distribuée), où les données ne quittent jamais l'agence locale.

2 Architecture Technique

Le système est structuré en trois couches logiques, assurant scalabilité et sécurité :

2.1 Couche 1 : Edge Computing (Les Agences)

Chaque agence agit comme un nœud indépendant (`edge_node.py`).

— **Confidentialité :** Les fichiers CSV contenant les transactions restent locaux.
— **Traitement :** Un modèle `LogisticRegression` est entraîné localement.
— **Extraction :** Seuls les coefficients mathématiques (Poids W et Biais b) sont extraits. Aucune donnée personnelle n'est exposée.

2.2 Couche 2 : Fog Computing (Communication)

L'utilisation d'**Apache Kafka** garantit le découplage entre les agences et le serveur.

— Les mises à jour des modèles sont envoyées vers le topic `fraud-model-updates`.
— Kafka assure une *Haute Disponibilité* : si le serveur central est hors ligne, les mises à jour sont mises en file d'attente.

2.3 Couche 3 : Cloud Computing (Agrégation)

Le serveur central (`cloud_server.py`) récupère les poids et applique l'algorithme **FedAvg** (Moyenne Pondérée) pour générer un modèle global unifié.

3 Rapport d'Exécution et Résultats

Les tests ont été effectués sur une infrastructure simulée comprenant 3 nœuds (Agences) et un serveur d'agrégation.

3.1 Performance des Nœuds Locaux (Edge Nodes)

Chaque agence a entraîné son modèle sur un jeu de données massif d'environ 1.4 million de transactions. Les logs d'exécution montrent une stabilité remarquable des modèles locaux.

Nœud	Volume de Transactions	Précision Locale	Statut
Agency_1	1,408,653	61.45%	Succès
Agency_2	1,408,653	61.43%	Succès
Agency_3	1,408,652	61.53%	Succès

TABLE 1 – Résultats de l'entraînement local par agence

3.2 Processus d'Agrégation (Serveur Central)

Le serveur a reçu les mises à jour via Kafka et a procédé à la fusion des modèles en temps réel. Voici la séquence des événements extraite des logs :

Listing 1 – Extrait des logs du serveur

```
(Round 1) Fusion Agency_3 + Agency_1 : Pr cision Globale -> 61.49%
(Round 2) Fusion Agency_2 + Agency_1 : Pr cision Globale -> 61.44%
(Round 3) Fusion Agency_2 + Agency_3 : Pr cision Globale -> 61.48%
```

4 Analyse du Tableau de Bord

L'analyse des graphiques générés par le système de monitoring confirme plusieurs points clés :

1. **Convergence et Stabilité :** La précision globale oscille très peu (entre 61.44% et 61.49%). Cela indique que les données sont distribuées de manière homogène (IID) entre les agences, facilitant l'apprentissage.
2. **Robustesse :** Le modèle global maintient une précision élevée (61.48%) même lorsque seulement deux agences participent à un round, prouvant la résilience du système face aux pannes potentielles d'une agence.

5 Conclusion

Ce projet démontre avec succès la faisabilité technique de l'Apprentissage Fédéré pour le secteur bancaire.

— **Objectif atteint :** Nous avons un modèle intelligent unifié sans jamais centraliser les données brutes.
— **Performance :** Le traitement distribué réduit la charge serveur.
— **Sécurité :** L'utilisation de conteneurs isolés (Docker) et de canaux sécurisés (Kafka) répond aux exigences de sécurité modernes.

Les perspectives futures incluent le déploiement de ce système sur un cluster Kubernetes pour gérer dynamiquement l'ajout de nouvelles agences bancaires.