

Compte Rendu d'Analyse et de Modélisation pour la Détection de Fraude

1. Introduction : Contexte, Problématique et Objectifs

Contexte

Cette étude porte sur l'analyse et la modélisation d'un jeu de données de transactions par carte de crédit afin d'identifier les opérations frauduleuses. L'objectif final est de développer un système automatisé capable d'évaluer le risque de chaque transaction en temps réel.[1]

Problématique

Le problème central est une tâche de classification binaire (Fraude vs. Non-Fraude). La difficulté majeure réside dans l'asymétrie extrême des classes, avec un ratio d'environ 99 transactions légitimes pour 1 transaction frauduleuse, rendant l'entraînement des modèles difficile car ils privilégient la classe majoritaire.[2]

Objectifs

L'objectif principal est de développer un modèle mesuré par sa capacité à identifier la classe minoritaire (fraude). Les objectifs spécifiques incluent l'implémentation de techniques de pré-traitement pour données asymétriques, l'évaluation de plusieurs algorithmes de classification, et l'optimisation des hyperparamètres pour maximiser le score ROC AUC.[2]

2. Méthodologie : Justification des Choix Techniques

Pré-traitement et Ingénierie de Caractéristiques

Choix Technique	Justification
Ingénierie de Caractéristiques Temporelles	La colonne TransactionDate a été transformée pour extraire Year, Month, Day, DayOfWeek, Hour, car la fraude est souvent cyclique ou liée à des plages horaires spécifiques [1].

Choix Technique	Justification
Encodage One-Hot (OHE)	Les variables catégorielles comme TransactionType et Location ont été encodées pour éviter d'attribuer un ordre aux catégories nominales [1].
Mise à l'Échelle (StandardScaler)	La variable Amount a été standardisée pour éviter que sa magnitude domine les modèles basés sur la distance ou le gradient [1].

Algorithmes de Modélisation

Deux familles de modèles ont été choisies : Régression Logistique comme baseline pour les relations linéaires, et Random Forest pour sa robustesse face aux relations non-linéaires et interactions.[2]

Gestion du Déséquilibre de Classe

Le paramètre `class_weight='balanced'` a été activé pour ajuster les poids dans la fonction de coût, attribuant un poids plus élevé aux erreurs sur la classe minoritaire.[2]

3. Résultats & Discussion : Métriques et Analyse des Erreurs

Métriques d'Évaluation

Modèle Optimisé	ROC	AUC	F1-Score	Recall	Accuracy
Régression Logistique	0.4732	0.0177	0.4150	0.9890	
Random Forest	0.4978	0.0000	0.0000	0.9900	[2]

Analyse des Performances

Les résultats sont insatisfaisants : les scores ROC AUC proches ou inférieurs à 0.5 indiquent une performance aléatoire ou pire. Le Random Forest a un Recall et F1-Score de 0, signifiant zéro détection de fraude sur le test. Les caractéristiques montrent une faible corrélation avec `IsFraud`, et `class_weight='balanced'` s'avère insuffisant.[2]

4. Conclusion : Limites du Modèle et Pistes d'Amélioration

Limites du Modèle Actuel

Le modèle est inexploitable en raison de son incapacité à généraliser sur la classe minoritaire, de l'insuffisance de la pondération des classes pour un ratio 99:1, et de l'absence de caractéristiques comportementales comme la fréquence ou les montants anormaux.[2]

Pistes d'Amélioration

- Implémenter SMOTE pour suréchantillonnage synthétique de la classe minoritaire.
- Créer des features comme vélocité des transactions (group-by par utilisateur ou lieu sur 1h/24h/7j) et anomalies de dépenses.
- Tester XGBoost, LightGBM ou CatBoost pour leur efficacité sur données tabulaires déséquilibrées.
- Utiliser cost-sensitive learning pour ajuster le seuil de décision et pénaliser les faux négatifs.[2]