Projet 2.3 – Détection de fraude bancaire

Ce projet aborde un problème critique en cybersécurité financière : la détection de transactions frauduleuses dans un environnement hautement déséquilibré, où moins de 0,2 % des opérations sont malveillantes. L'objectif n'est pas seulement la précision, mais surtout de minimiser les faux négatifs (fraudes non détectées).

- **Contexte réel** : Les systèmes bancaires traitent des millions de transactions par jour. Une fraude non détectée peut coûter des dizaines de milliers d'euros. Cependant, marquer une transaction légitime comme frauduleuse (faux positif) peut bloquer un client — ce qui est problématique, mais moins grave qu'une fraude passée inaperçue.
- **Mission précise** : À partir du dataset **Credit Card Fraud Detection** (Kaggle), vous devrez entraîner un modèle capable de détecter les fraudes avec un **rappel (recall) $>$ 90 %** tout en maintenant un taux de faux positifs raisonnable.
- **Méthode utilisée** : Vous testerez plusieurs approches :
  - **Isolation Forest** : détecte les anomalies sans supervision.
  - **XGBoost avec seuil optimisé** : modèle supervisé robuste au déséquilibre.
  - **SMOTE** (Synthetic Minority Oversampling Technique) : pour équilibrer artificiellement les classes avant l'entraînement.
- **Dataset initial** : **Credit Card Fraud Detection** (284 807 transactions, 492 fraudes). Lien : https ://www.kaggle.com/datasets/mlg-ulb/creditcardfraud → Les features sont déjà anonymisées et normalisées (PCA), sauf le montant et la classe.
- **Résultat attendu** :
  - Un modèle avec **ROC-AUC $>$ 0.95** et **recall $>$ 0.90**.
  - Une courbe ROC tracée avec `matplotlib`.
  - Un **seuil de décision optimisé** (plutôt que 0.5 par défaut) pour maximiser le rappel.
  - Une analyse des erreurs : quelles fraudes sont manquées ? Pourquoi ?
- **Guide de réalisation pas à pas** :
  1. Téléchargez le CSV depuis Kaggle.
  2. Chargez-le avec `pandas.read_csv()`.
  3. Normalisez la colonne `Amount` avec `StandardScaler`.
  4. Divisez en train/test (`stratify=y` pour préserver la proportion de fraudes).
  5. Testez d'abord `IsolationForest` (sans échantillonnage).
  6. Puis testez `XGBClassifier` avec et sans SMOTE (`imblearn.over_sampling.SMOTE`).
  7. Comparez les matrices de confusion.
  8. Optimisez le seuil à l'aide de `precision_recall_curve`.
  9. Produisez un rapport clair avec vos observations.
- **Compétences visées** :
  - Gestion de données déséquilibrées.
  - Sélection et ajustement de seuils en classification binaire.
  - Évaluation multi-métrique (accuracy, precision, recall, F1, AUC).
  - Utilisation de bibliothèques spécialisées (`imblearn`, `xgboost`).