Rapport du Projet : Détection de Fraude Bancaire

Introduction

Ce projet a pour objectif de détecter les fraudes bancaires en combinant l'analyse de données SQL et l'intelligence artificielle. Nous avons conçu une base de données transactionnelle, effectué des analyses avancées avec SQL et développé un modèle de machine learning pour prédire les transactions frauduleuses. L'optimisation de la base de données et l'intégration avec Python ont permis d'améliorer la performance du système tout en automatisant la détection des anomalies.

Objectifs du Projet

Les principaux objectifs de ce projet étaient :

- Concevoir une base de données transactionnelle en SQL pour gérer les clients, comptes et transactions.
- Effectuer des analyses avancées sur les données avec des requêtes SQL complexes.
- Utiliser des algorithmes d'intelligence artificielle, comme les modèles de classification, pour détecter les fraudes bancaires.
- Optimiser la base de données avec des index et des partitionnements pour améliorer les performances des requêtes.
- Créer un pipeline SQL → Python → Machine Learning pour l'analyse et la prédiction des fraudes bancaires.

1. Conception de la Base de Données

Tables créées :

1. **Clients :** Contient les informations relatives aux clients bancaires (identifiant, nom, prénom, âge, sexe, pays et revenu annuel).
2. **Comptes :** Contient les informations des comptes bancaires des clients (identifiant du compte, type de compte, solde, et date de création).
3. **Transactions :** Contient les informations des transactions effectuées (identifiant de la transaction, identifiant du compte, montant, date, type de transaction, lieu et indication si la transaction est frauduleuse).

La création des tables a été réalisée en utilisant des commandes SQL de type CREATE TABLE, et des relations ont été établies entre elles via des clés étrangères (par exemple, la relation entre les tables Comptes et Clients).

2. Analyse SQL et Requêtes Avancées

Des analyses SQL ont été réalisées sur la base de données pour obtenir des informations intéressantes, comme le montant total des transactions par type de transaction. Voici un exemple de requête SQL pour obtenir l'agrégation des montants des transactions par type :

```
query = '''
    SELECT type_transaction, SUM(montant) AS total_montant
    FROM Transactions
    GROUP BY type_transaction
    ORDER BY total_montant DESC
'''

aggregated_data_df = pd.read_sql_query(query, conn)
print(aggregated_data_df.head())
```

3. Détection de Fraude avec Machine Learning

Pour détecter les fraudes bancaires, un modèle de classification a été entraîné sur les données de transactions. La variable cible était la colonne fraude, où 0 indique une transaction non frauduleuse et 1 une transaction frauduleuse.

Le modèle utilisé était un modèle de régression logistique pour la classification binaire. Après l'entraînement du modèle, une évaluation des performances a été effectuée en utilisant des métriques telles que la précision, le rappel et le score F1.

Voici un résumé des résultats d'évaluation du modèle :

	precision	recall	f1-score	support
0	0.49	0.61	0.55	155
1	0.48	0.36	0.41	154
accuracy			0.49	309
macro avg	0.49	0.49	0.48	309
weighted avg	0.49	0.49	0.48	309

Les résultats montrent que le modèle a une précision raisonnable pour prédire les transactions non frauduleuses, mais il rencontre des difficultés pour identifier correctement les transactions frauduleuses. Cela est dû au déséquilibre des classes, avec un nombre beaucoup plus élevé de transactions non frauduleuses que frauduleuses.

Améliorations possibles :

- **Utilisation de techniques de rééchantillonnage**, telles que le SMOTE (Synthetic Minority Over-sampling Technique), pour équilibrer les classes et aider le modèle à mieux identifier les transactions frauduleuses.
- **Exploration de modèles plus sophistiqués** comme XGBoost ou Random Forest, qui sont connus pour leur efficacité dans des problèmes de classification déséquilibrée et qui pourraient améliorer la détection des fraudes.

4. Optimisation de la Base de Données

Pour améliorer la performance des requêtes SQL, des index ont été ajoutés sur les colonnes les plus fréquemment utilisées dans les jointures et les recherches, comme client_id, compte_id, et date_transaction.

Voici un exemple de création d'un index sur la table Transactions :

```
conn.execute('''
    CREATE INDEX IF NOT EXISTS idx_compte_id ON Transactions (compte_id);
''')

# Créer un index sur la colonne date_transaction dans la table Transactions
conn.execute('''
    CREATE INDEX IF NOT EXISTS idx_date_transaction ON Transactions (date_transaction);
''')
```

Ces index ont permis d'accélérer les requêtes de jointures et de filtrage, surtout lors des analyses sur les transactions.

5. Conclusion

Ce projet a permis de répondre aux objectifs suivants **:**

- Conception et implémentation d'une base de données transactionnelle en SQL.
- Réalisation d'analyses avancées sur les données avec SQL pour identifier des tendances et des anomalies.
- Développement d'un modèle de machine learning pour détecter les fraudes bancaires.
- Optimisation des performances de la base de données avec des index.

Malgré des résultats prometteurs, le modèle de détection de fraude nécessite des améliorations, notamment en raison du déséquilibre des classes. De futures améliorations incluront l'application de techniques d'over-sampling et l'exploration de modèles plus avancés.