Rapport final

Ayman Amarouche 24009536

Enseignant : Mr **Abdelkader BERROUACHEDI**

Cours : Outils libres et développement logiciel

Lien Git du Projet **:** https://github.com/salutos93/finance

Sommaire

1. Introduction

Le projet est réalisé dans le cadre du module "Outils libres pour le développement logiciel" de mon Master 1 Informatique. L'objectif principal de ce travail est de démontrer la maîtrise et l'intégration d'un ensemble d'outils de développement open-source dans la résolution d'une problématique en science des données. J'ai choisi un sujet de **finance**, plus précisément sur la **détection de fraude par carte bancaire**.

À travers ce projet, nous allons parcourir les étapes clés d'un processus d'analyse de données : Nous allons commencer par acquérir et explorer les données, puis ensuite nous allons les nettoyer et les préparer avant de pouvoir les analyser.

Les outils que j'ai utilisés dans le cadre du projet sont :

- **Jupyter Notebook**
- **GitHub**
- **Python (pandas, matplotlib, seaborn...)**
- **PostgreSQL (avec terminal ubuntu)**
- **Kaggle (dataset)**

Le rapport détaillera les choix technologiques et méthodologiques, les analyses réalisées et les conclusions tirées sur les caractéristiques des transactions frauduleuses.

2. Acquisition et Exploration des Données

2.1. Présentation du dataset

La première étape du projet a consisté à acquérir et à se familiariser avec le dataset. J'ai téléchargé le jeu de données "Credit Card Fraud Detection" depuis la plateforme Kaggle. Ce dataset contient des informations sur 284 807 transactions par carte de crédit.

Tout d'abord j'ai essayé de comprendre le CSV téléchargé via Kaggle, tout en lisant la documentation, on voit des colonnes Vn, Time, Class... : Que peuvent-elles bien signifier ?

En lisant la documentation, j'ai pu comprendre que les Vn sont des variables anonymisées par PCA (date de transaction, lieu de transaction etc). Le PCA est une technique d'analyse de données qui permet de transformer des données corrélées entre elles en variables décorrélées. Ici, c'est surtout pour la confidentialité des données bancaires. Elles ne sont par conséquent pas compréhensibles pour l'humain, seulement utilisables par les algorithmes.

"Time" est le temps écoulé pour chaque transaction depuis la toute première transaction dans le dataset.

2.2. Exploration des données

Après cette première prise de connaissance, j'ai procédé à une exploration initiale des données à l'aide de la bibliothèque Python `pandas` et de Jupyter Notebook. Les étapes suivantes ont été réalisées :

- **Chargement du dataset :** Le fichier `creditcard.csv` a été chargé dans un DataFrame `pandas` à l'aide de la fonction `pd.read_csv()`.
- **Aperçu de la structure :** Les premières lignes du DataFrame (`df.head()`) ont été affichées pour visualiser la structure des colonnes et les types de données.

- **Informations générales et valeurs manquantes :** La méthode `df.info()` a fourni un résumé précisdu DataFrame en révélant **l'absence de toute valeur manquante**, ce qui simplifie la phase de nettoyage.
- **Statistiques descriptives :** `df.describe()` a généré des statistiques descriptives clés (moyenne, écart-type, min/max, quartiles) pour les colonnes numériques, offrant une première vue sur la distribution des valeurs.

3. Nettoyage et Préparation des Données

3.1. Transformation de "Time”

Bien qu’il n’y avait pas de valeurs manquantes dans le dataset (ce qui est très pratique), je me suis rendu compte que la colonne time n’était pas pratique (secondes, pour des millions de valeurs ça fait beaucoup trop) donc il fallait se ramener en heures. Par exemple pour pouvoir observer des tendances, il est mieux de regrouper toutes les heures par plages horaires cycliques.
C’est obligatoire à terme pour pouvoir voir les heures dans lesquelles les fraudes se produisent le plus.

3.2. Stantardisation de “Amount”

De plus, on remarque que toutes les variables Vn sont déjà centrées autour de 0 avec PCA, on remarque que Amount a des valeurs brutes trop élevées qui risquent de créer du biais en dominant les autres variables, et par conséquent rendre la visualisation obsolète (en gros on cherche juste à mettre Amount à l’échelle avec les autres variables)

Pour remédier à cela, la colonne `Amount` a été standardisée à l'aide de `StandardScaler` de la bibliothèque `scikit-learn`. Cette opération a créé une nouvelle colonne, `Scaled_Amount`, dont la moyenne est désormais très proche de 0 et l'écart-type très proche de 1. Cette standardisation garantit que toutes les caractéristiques (`Vn`, `Hour`,

`Scaled_Amount`) contribuent de la même manière aux analyses, sans une qui prend le dessus sur les autres.

4. Intégration de la Base de Données (PostgreSQL)

4.1. Configuration de l'environnement

L'intégration d'une base de données relationnelle était une exigence de ce projet. J'ai choisi d'utiliser **PostgreSQL** pour sa robustesse. J'ai déployé l'infrastructure (PostgreSQL, Python, Jupyter) a été déployé dans mon environnement **Ubuntu sous Windows Subsystem for Linux (WSL)**. Un **environnement virtuel Python (`venv`)** a également été créé et activé pour isoler les dépendances spécifiques à ce projet

- **Ensuite, mise en place de PostgreSQL J'ai utilisé Ubuntu avec WSL, venv pour python** : Très bien, c'est la justification de l'environnement.

Pour faciliter l'insertion du DataFrame `pandas` dans PostgreSQL, j'ai utilisé une combinaison de bibliothèques Python :

- `psycopg2` : Ce pilote est l'interface directe et de bas niveau pour communiquer avec PostgreSQL depuis Python. Il gère la connexion au serveur et l'exécution des requêtes SQL brutes.
- `SQLAlchemy` : Cette bibliothèque permet à `pandas` (notamment sa méthode `to_sql()`) d'interagir de manière avec différentes bases de données. `SQLAlchemy` utilise ensuite `psycopg2` en arrière-plan pour les communications spécifiques à PostgreSQL.

La chaîne de communication peut être visualisée comme suit :
`DataFrame Pandas` ---(utilise)---> `SQLAlchemy` ---(utilise)---> `psycopg2` ---(communique avec)---> `Serveur PostgreSQL`

L'insertion a été validée en exécutant des requêtes SQL pour compter et afficher les premières lignes de la table, confirmant que l'ensemble des 284 807 transactions étaient bien stockées de manière persistante.

4.2. **Pourquoi PostgreSQL ?**

On peut se demander pourquoi j'ai utilisé PostgreSQL et le langage SQL, vu que les opérations de chargement, nettoyage et analyse exploratoire ont été toutes réalisées avec Python et la bibliothèque Pandas. Pour ce dataset de taille moyenne et le type d'analyse effectué, une approche purement Python aurait pu sembler suffisante, les deux points les plus importants sont les suivant :

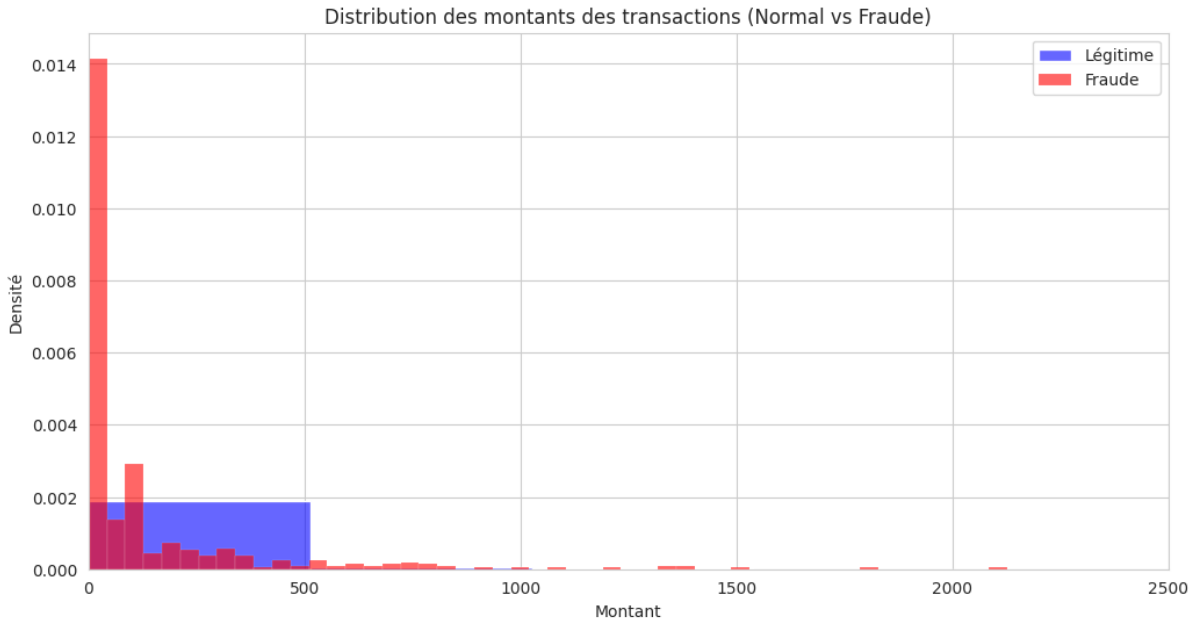
1. **Persistance et Fiabilité des Données :**
    - **Limitation de Pandas :** Un DataFrame Pandas est une structure de données qui réside en mémoire vive (RAM). Lorsque ton programme Python s'arrête ou que Jupyter est fermé, le DataFrame disparaît.
    - **Avantage de la BDD :** Une base de données comme PostgreSQL stocke les données de manière **persistante et sécurisée** sur le disque dur. Les données restent accessibles même après l'arrêt de l'ordinateur ou de l'application. C'est crucial pour tout système où les données ne doivent pas être perdues.
2. **Gestion de Volumes de Données Plus Importants (Scalabilité) :**
    - Le dataset de détection de fraude (environ 285 000 lignes) est de taille moyenne. Cependant, les bases de données sont conçues pour gérer des **téravélions, voire des pétaoctets de données**. (1 téra = 1000 GO, 1 péta = 1000 TERA =

- Si ton dataset contenait des dizaines de millions ou des milliards de transactions, Pandas seul ne pourrait pas les charger entièrement en mémoire sur un ordinateur standard. Une base de données permettrait d'interroger et de traiter ces données de manière efficace, en ne chargeant en mémoire que les portions nécessaires. C'est un principe du "Big Data".

En conclusion :
L'utilisation de PostgreSQL et SQL dans ce projet permet de maîtriser des outils fondamentaux pour la gestion de données, leur persistance, leur scalabilité, ce que Pandas seul ne peut pas couvrir.
C’est indispensable si je veux me projeter dans un projet de plus grande envergure.

5. Analyses Statistiques et Visualisations

5.1. Déséquilibre des Classes



- **Pic Massif Près de Zéro :** Les deux distributions (transactions légitimes en bleu, transactions frauduleuses en rouge) présentent un pic extrêmement prononcé tout à gauche de l'axe des "Montant", c'est-à-dire pour des montants très faibles, proches de zéro.
- **Montant de 0 pour les fraudes ? :** Sur ce type d'histogramme un pic à 0 signifie que la très grande majorité des transactions concernées ont un montant compris dans l'intervalle le plus bas (par exemple, entre 0 et 50 unités monétaires, selon la largeur de la barre). Pour les fraudes (rouge), ce pic initial est assez élevé.
- **Forme de la Distribution Légitime (Bleu) :** Après le pic initial, la densité des transactions légitimes diminue rapidement mais s'étend sur une plage beaucoup plus large (même si le graphique est coupé à 2500 sur l'axe X pour améliorer la lisibilité, les statistiques montrent des montants légitimes allant jusqu'à 25 691).
- **Forme de la Distribution Frauduleuse (Rouge) :** La distribution des fraudes est très concentrée autour des très faibles montants. Bien qu'elle s'étende aussi vers des montants plus élevés, cette extension est bien moins prononcée que pour les transactions légitimes (le maximum des fraudes est de 2125, contre 25691 pour les légitimes).

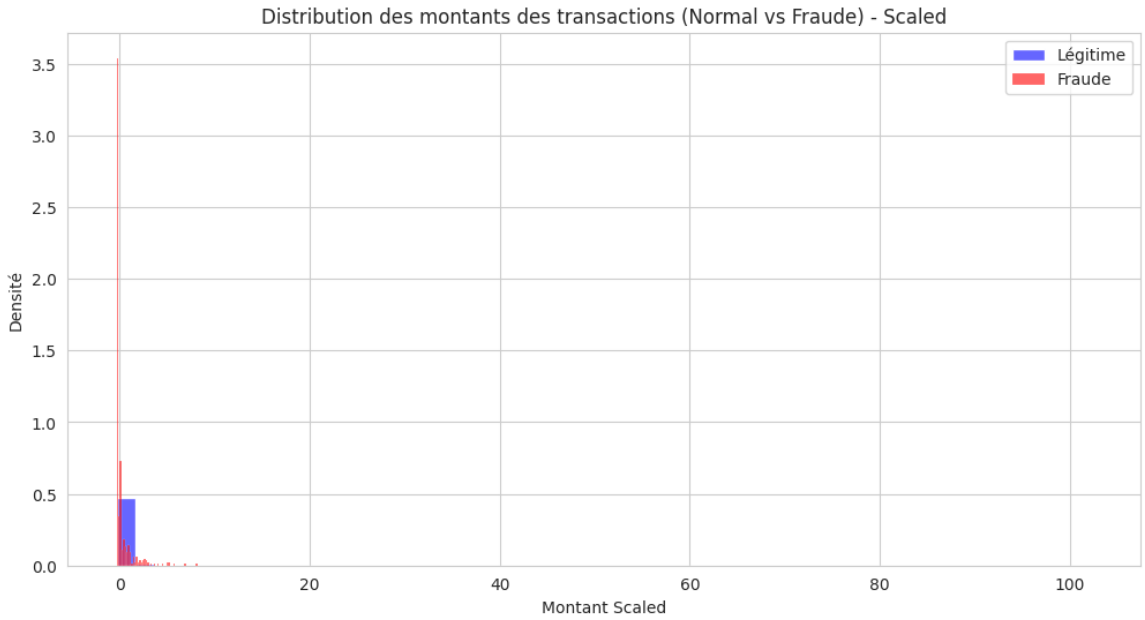
Interprétation :

Ce graphique est très révélateur des stratégies de fraude. Le fait que la **majorité des transactions frauduleuses soient de très faibles montants** (le pic rouge écrasant près de zéro) suggère que les fraudeurs tentent d'éviter la détection en réalisant de nombreuses petites transactions, susceptibles de passer sous les seuils d'alerte automatisés ou de ne pas attirer l'attention des victimes.

Bien que les transactions légitimes présentent également un grand nombre de faibles montants (achats quotidiens), leur distribution s'étale sur une gamme de montants beaucoup plus vaste,

incluant des transactions très importantes, ce qui n'est pas le cas des fraudes. Cela confirme que le **montant de la transaction est un facteur discriminant** significatif pour différencier les fraudes des transactions légitimes.

5.2. Analyse du Montant

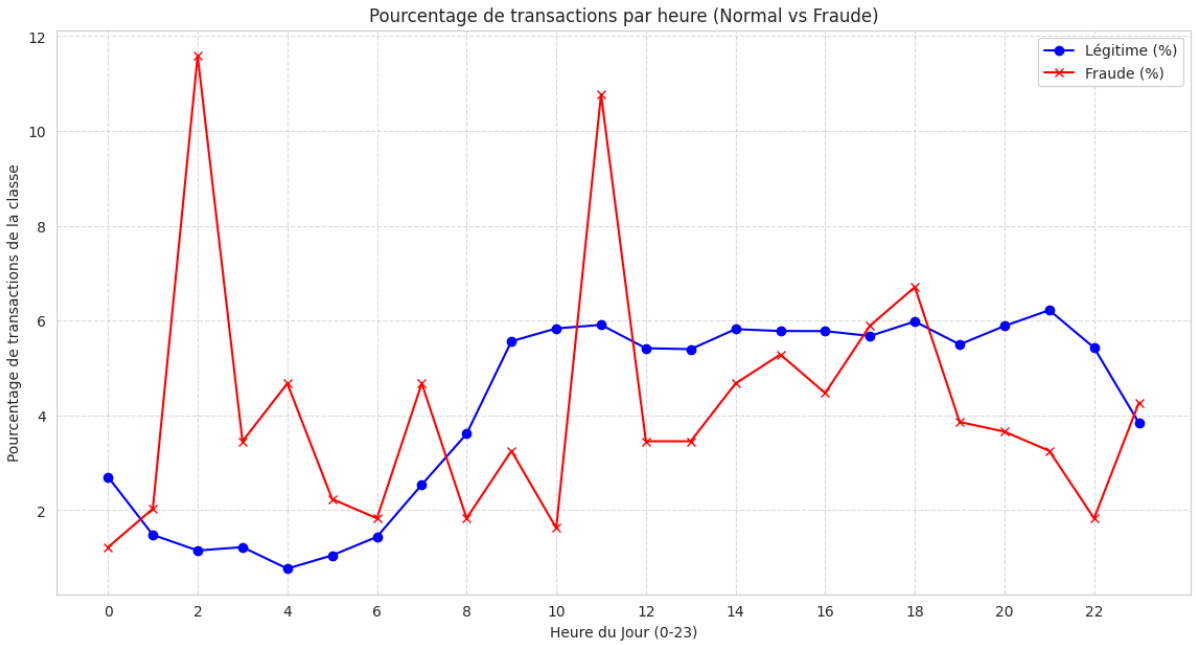


- **Similitude :** Ce graphique présente une forme générale très similaire au premier histogramme des montants bruts. On observe toujours des pics très élevés et concentrés pour les deux catégories de transactions (légitimes en bleu, frauduleuses en rouge) près de la valeur zéro sur l'axe des "Montant Scaled".
- **Changement d'Échelle :** L'axe X, "Montant Scaled", montre des valeurs différentes de celles du montant brut. Comme nous l'avons vu, la standardisation a transformé les montants originaux pour qu'ils aient une moyenne très proche de 0 et un écart-type de 1.
- **Concentration des Fraudes :** Le pic de densité pour les transactions frauduleuses (rouge) reste extrêmement élevé et focalisé autour de zéro sur l'échelle standardisée, indiquant une forte concentration des fraudes sur les valeurs faibles de `Scaled_Amount`.

Interprétation :

- Ce graphique a pour principal objectif de **visualiser l'effet de la standardisation** et de montrer comment la distribution des montants se présente sur une échelle comparable aux autres caractéristiques anonymisées (Vn).
- Malgré la transformation des valeurs brutes en valeurs standardisées, l'interprétation reste la même : une proportion très significative des transactions frauduleuses se situe dans la plage des très faibles montants (qui, une fois standardisés, se regroupent autour de zéro).

5.1.3. Analyse de l'Heure



Observations Clés :

- **Courbe Légitime (Bleu) :** Cette courbe représente le pourcentage de transactions légitimes qui se produisent à chaque heure du jour. On observe un schéma très prévisible et logique :
    - Activité très faible pendant les heures de nuit (de 0h à environ 6h du matin).
    - Augmentation progressive de l'activité à partir de 6h-7h du matin.
    - Une forte activité en journée, avec des plateaux ou de légers pics pendant les heures ouvrables (par exemple, autour de 9h-12h et 14h-18h).
    - Diminution de l'activité en soirée et durant la nuit.
      Ce comportement est typique des activités humaines et commerciales classiques.
- **Courbe Frauduleuse (Rouge) :** Cette courbe montre un schéma d'activité frauduleuse très différent :

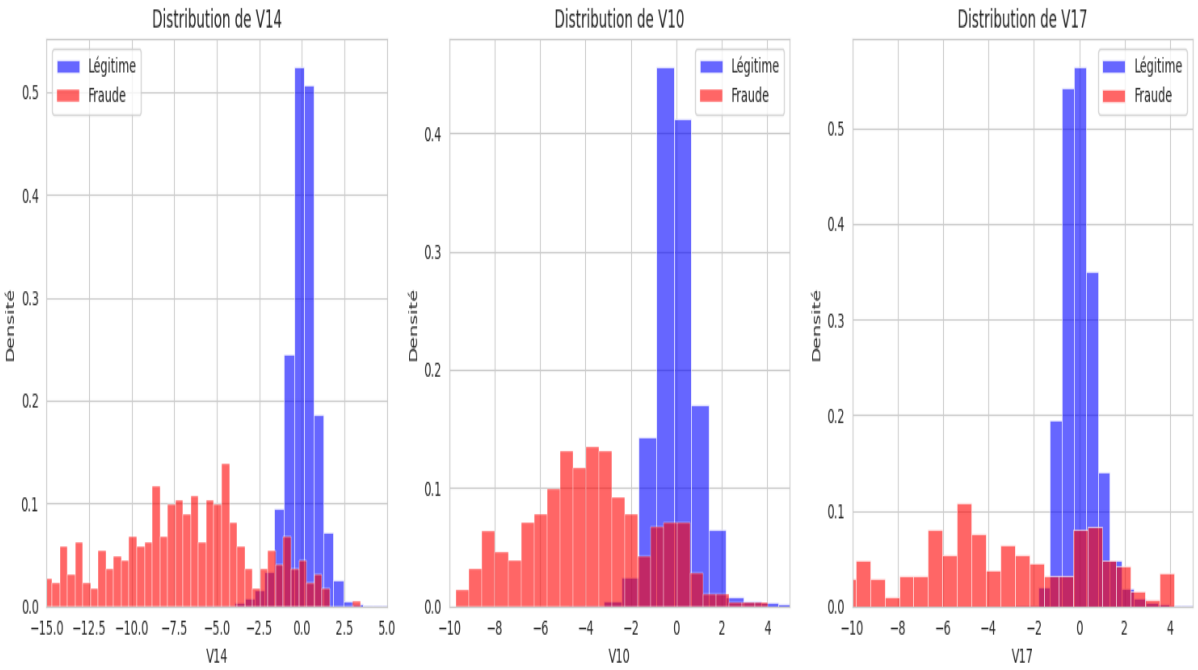
- Un **pic très prononcé aux alentours de 2h du matin**, où le pourcentage de fraudes est le plus élevé par rapport à l'ensemble des fraudes (près de 12%). À cette heure, l'activité légitime est à son minimum.
- Un autre **pic significatif en fin de matinée (environ 11h)**.
- Contrairement aux transactions légitimes, l'activité frauduleuse proportionnelle ne diminue pas autant la nuit, et présente des variations plus brusques.
- Un autre pic moins grand est observable vers 18h.

Interprétation :

Ce graphique met en évidence que le **moment de la journée est un facteur clé** pour identifier une transaction frauduleuse.

- La **forte concentration des fraudes durant les heures de nuit** (notamment le pic de 2h du matin) est particulièrement révélatrice. Cela suggère que les fraudeurs peuvent choisir ces moments pour opérer, sachant que les systèmes de surveillance manuels sont moins actifs, ou que les utilisateurs dorment et sont moins susceptibles de détecter des activités suspectes immédiatement.
- Les autres pics (ex: 11h, 18h) montrent que les fraudeurs peuvent aussi essayer de se fondre dans les périodes de forte activité légitime, mais leur distribution globale reste distincte, ne suivant pas le même "rythme" que les transactions normales.

5.1.4. Analyse des Caractéristiques Vn



Observations Clés :

Ce groupe de graphiques, présentant les distributions de variables Vn sélectionnées (V14, V10, V17), est le plus révélateur.

- **Distributions Légitimes (Bleu) :** Pour les trois variables, les transactions légitimes (en bleu) montrent une distribution fortement concentrée autour de zéro (ou d'une valeur très proche de zéro), formant des pics nets. C'est le comportement attendu pour des caractéristiques après une transformation PCA.
- **Distributions Frauduleuses (Rouge) :** En contraste, les transactions frauduleuses (en rouge) présentent des distributions radicalement différentes et **nettement décalées** par rapport aux légitimes :
    - Pour V14, la distribution des fraudes est étalée et centrée sur des valeurs négatives significatives (autour de -5 à -10), très loin du pic des transactions légitimes.
    - Pour V10, une séparation similaire est visible, avec les fraudes ayant des valeurs plus négatives que les légitimes.
    - Pour V17, la même tendance de décalage vers des valeurs négatives est observée pour les fraudes par rapport aux transactions légitimes.
- **Absence de Chevauchement :** Pour ces variables, il y a très peu de chevauchement entre les distributions des transactions légitimes et frauduleuses. Elles occupent des plages de valeurs distinctes.

Interprétation :

Ces graphiques sont la preuve visuelle que les **transactions frauduleuses possèdent une "signature" numérique bien différente** de celle des transactions légitimes.

Malgré l'anonymisation des variables Vn (qui nous empêche de comprendre ce que V14 représente concrètement), leur analyse démontre que :

- Les transactions légitimes se regroupent dans un "nuage" dans l'espace défini par les Vn.
- Les transactions frauduleuses, étant des anomalies, se situent dans des **zones très éloignées et différentes** de ce nuage . Elles se comportent de manière anormale le long de ces dimensions.

6. Gestion de Version avec Git/GitHub

J'ai utilisé **Git** localement et **GitHub** comme plateforme de dépôt distant. Git m'a permis de :

- **Suivre l'historique des modifications :** Chaque étape de développement, de la configuration initiale à l'analyse finale, a été enregistrée via des `commits`, permettant de documenter la progression et de revenir à des versions antérieures si nécessaire.

- **Assurer la collaboration et la reproductibilité :** L'utilisation de Git et GitHub prépare à des environnements de travail collaboratifs et facilite la reproduction du projet par des tiers.
- **Présenter le travail :** Le dépôt GitHub sert de portfolio et de point d'accès centralisé au code et aux ressources du projet.

Le dépôt Git a été initialisé (`git init`) au début du projet. J'ai utilisé `.gitignore` a été mis en place très tôt pour exclure certains fichiers et répertoires :

- Le dossier de l'environnement virtuel (`venv/`) contenant toutes les bibliothèques Python (car ces dépendances peuvent être réinstallées via `pip` et alourdissent inutilement le dépôt).
- Les fichiers de checkpoints Jupyter (`.ipynb_checkpoints/`).
- Les fichiers de données brutes (`creditcard.csv` et `creditcardfraud.zip`), car leur taille (plus de 100 Mo pour le CSV) dépasse les limites de GitHub pour les fichiers standards. Ces fichiers sont téléchargeables séparément via Kaggle, dont le lien est fourni dans le `README.md`."

Après chaque étape majeure, les modifications ont été ajoutées à l'index (`git add .`) puis sauvegardées dans l'historique local (`git commit`). Enfin, le dépôt local a été lié à un dépôt distant sur GitHub (`git remote add origin ...`), et toutes les modifications ont été poussées (`git push`) pour rendre le projet accessible en ligne.

7. Conclusion et Perspectives

Ce projet a été une opportunité d'appliquer et de consolider mes compétences sur les outils libres pour le développement logiciel dans un contexte financier. J'ai pu maîtriser l'ensemble d'un pipeline d'analyse de données, de l'acquisition à la restitution, en passant par des étapes cruciales de préparation et d'intégration avec une base de données."

Les apprentissages clés incluent :

- La gestion d'un projet avec **Git et GitHub**, essentielle pour le versioning et la collaboration.
- La manipulation et l'analyse de grands volumes de données avec **Python et Pandas** dans l'environnement **Jupyter Notebook**.
- L'intégration et l'interaction avec une base de données relationnelle **PostgreSQL**, justifiée par la persistance, la scalabilité (adaptable avec de plus en plus de données) et la puissance des requêtes SQL.
- La réalisation d'une **analyse exploratoire de données** pour caractériser un phénomène complexe comme la fraude bancaire, en extrayant des schémas à l'aide de données brutes et transformées.

Les analyses ont clairement démontré que les transactions frauduleuses possèdent des 'signatures' distinctes en termes de montant, de timing, et surtout des caractéristiques anonymisées Vn, prouvant ainsi que la détection d'anomalies est statistiquement possible.

Pour aller plus loin, plusieurs perspectives pourraient être explorées :

- L'implémentation d'un **modèle de Machine Learning** (par exemple, un algorithme de classification) pour prédire les transactions frauduleuses, en gérant le déséquilibre important des classes.
- L'exploration de techniques d'optimisation des requêtes SQL pour des volumes de données encore plus importants.
- L'intégration d'outils de visualisation interactifs avancés pour une exploration plus dynamique des données.

8. Références / Bibliographie

- **Dataset :** Credit Card Fraud Detection via Kaggle: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
- **Dépôt GitHub du Projet :** https://github.com/salutos93/finance