# 1 — Cadrage du projet :

## **Objectif du projet :**

Le projet vise à **détecter automatiquement les transactions bancaires suspectes** via un algorithme de **clustering non supervisé (K-Means)**, basé sur des caractéristiques comme le **montant**, **localisation**, **type de transaction**, **durée**, **solde**, **âge du client**, etc..

**Actions à mener :**

* 🔍 **Comprendre le dataset** : quelles colonnes sont disponibles, quelles sont les unités, les types de transactions ?
* 🧠 **Définir ce qu’on appelle "fraude" dans ton contexte** :
  + Est-ce basé uniquement sur des patterns inhabituels ?
  + As-tu des étiquettes (supervisé) ? Ici c’est non, donc on simule les fraudes par l’éloignement du centre des clusters.

📌 **Output attendu** : une phrase claire du type :

“Le projet vise à identifier les transactions suspectes en appliquant un algorithme de clustering non supervisé (K-Means), basé sur des caractéristiques transactionnelles (montant, âge du client, localisation, etc.).

### Définition de la Fraude dans notre contexte :

* Une transaction est considérée comme potentiellement frauduleuse lorsqu'elle s'écarte significativement des comportements normaux identifiés par notre modèle K-means
* Les critères de détection incluent :
* Des montants de transaction anormalement élevés par rapport à l'historique du client
* Des localisations géographiques suspectes ou des changements rapides de localisation
* Des transactions effectuées à des heures inhabituelles
* Des tentatives de connexion multiples échouées
* Des durées de transaction anormales
* Des écarts significatifs par rapport aux centroïdes des clusters K-means
* Des changements brusques dans les habitudes de transaction du client

### Actions Immédiates :

* Lorsqu'une transaction est détectée comme suspecte par le modèle K-means :
* Le système bloque automatiquement la transaction en attente de vérification
* Une alerte est envoyée en temps réel via l'interface Streamlit
* Le tableau de bord Power BI est mis à jour avec les nouvelles anomalies
* Le client reçoit une notification via son canal préféré
* Les agents de sécurité peuvent visualiser la transaction suspecte dans l'interface Streamlit
* Le système collecte des informations supplémentaires pour la vérification
* Si la fraude est confirmée :
* Le compte est gelé
* Le processus de remboursement est initié
* L'incident est documenté pour améliorer le modèle K-means
* Les métriques de performance (taux d'anomalie, faux positifs) sont mises à jour dans le tableau de bord

## 2. **Pourquoi ce projet ?**

### Valeur pédagogique :

* Apprendre l'analyse exploratoire de données (EDA)
* Maîtriser les techniques de clustering (K-means)
* Comprendre la détection d'anomalies non supervisée
* Développer des compétences en visualisation de données
* Apprendre à créer des interfaces utilisateur interactives

### Valeur professionnelle :

* Démonstrer des compétences en data
* Montrer la capacité à développer des solutions business
* Acquérir de l'expérience en détection de fraude

### Compétences ciblées :

1. **Techniques :**

* Python (Pandas, NumPy, Scikit-learn)
* Machine Learning (K-means, clustering)
* Visualisation (Streamlit, Power BI)
* Développement Web (Flask, API)

1. **Analytiques :**

* Analyse de données transactionnelles
* Détection de patterns anormaux
* Interprétation des résultats
* Optimisation des modèles

1. **Business :**

* Compréhension du domaine bancaire
* Gestion des KPIs (taux d'anomalie ≥ 5%, faux positifs ≤ 10%)
* Communication des résultats
* Documentation technique

## **3- Définition des KPI :**

1. **KPI Principal - Taux de Détection d'Anomalies :**

* Objectif : ≥ 5% (126 transactions sur 2514)
* Pourquoi : C'est le cœur du projet de détection de fraude
* Mesure : Via Streamlit en temps réel
* Impact : Démontre l'efficacité du modèle K-means

1. **KPI de Qualité - Taux de Faux Positifs :**

* Objectif : ≤ 10%
* Pourquoi : Évite de perturber les clients avec de fausses alertes
* Mesure : Validation par les agents via Streamlit
* Impact : Fiable et utilisable en production

1. **KPI de Performance - Temps de Détection :**

* Objectif : < 30 secondes
* Pourquoi : La détection de fraude doit être rapide
* Mesure : Temps entre la transaction et l'alerte
* Impact : Réaction rapide pour minimiser les pertes
* Basé sur : Notre dataset de transactions en temps réel

# 2-Stack Technique du Projet de Détection de Fraude :

1. **Langage et Environnement :**

* Python 3.x
* Jupyter Notebook pour l'EDA
* VS Code comme IDE

1. **Data Science & ML :**

* Pandas : Manipulation des données (2514 transactions)
* NumPy : Calculs numériques
* Scikit-learn :
* K-means pour le clustering
* StandardScaler pour la normalisation
* Metrics pour l'évaluation

1. **Visualisation :**

* Streamlit : Interface utilisateur interactive
* Matplotlib/Seaborn : Visualisations dans le notebook
* Power BI : Tableau de bord analytique

1. **Développement :**

* Flask : API pour les prédictions
* HTML/CSS : Interface Streamlit
* Git/GitHub : Versioning du code

1. **Gestion des Données :**

* Joblib : Sauvegarde du modèle K-means
* CSV : Format des données (bank\_transactions\_data\_2.csv)

Cette stack est :

* Adaptée à notre dataset
* Suffisante pour atteindre nos KPIs
* Réalisable dans le temps imparti
* Parfaite pour un projet de détection de fraude

**Livrable Final - Application de Détection de Fraude :**

## Interface Utilisateur (Streamlit) :

**Un formulaire pour entrer les données d'une transaction :**

* Montant
* Type de transaction (Débit/Crédit)
* Localisation
* Canal (ATM/Online/Branch)
* Âge du client
* etc.

**Un bouton "Vérifier la Transaction"**

=>Un résultat qui indique :

* "Transaction Suspecte" ou "Transaction Normale"
* Score de risque
* Raisons de la suspicion

## Fonctionnalités :

* Vérification en temps réel
* Historique des vérifications
* Statistiques des détections
* Export des résultats

## Utilisation :

* L'utilisateur entre les données d'une transaction
* L'application vérifie si c'est une fraude
* Donne un résultat immédiat
* Peut vérifier plusieurs transactions

## EXPLICATION PLUS CLAIRE VIA UN USER EXPERIENCE :

### Nouvelles Transactions :

* L'utilisateur entre une NOUVELLE transaction
* Cette transaction n'existe pas dans notre dataset original (2514 transactions)
* C'est une transaction réelle à vérifier

### Comment ça marche :

* Notre modèle K-means est déjà entraîné sur les 2514 transactions
* Il a appris les patterns "normaux" de transactions
* Quand une nouvelle transaction est entrée :
* Elle est normalisée (comme les données d'entraînement)
* Elle est comparée aux clusters existants
* Si elle est trop éloignée des clusters normaux → Suspecte
* Si elle est proche des clusters normaux → Normale

### Exemple Concret :

* Dataset original : 2514 transactions (historique)
* Nouvelle transaction : "Transaction de 1000$ à 3h du matin"
* Le modèle compare cette nouvelle transaction avec les patterns appris
* Si ce pattern est rare/éloigné des clusters normaux → Alerte de fraude

🡺l’application vérifie les NOUVELLES transactions en les comparant avec les patterns appris sur notre dataset historique .

# ARBRE STRUCTURE FOLDERS :

A screen shot of a computer

AI-generated content may be incorrect.

## Explication simple de chaque dossier :

A screenshot of a computer program

AI-generated content may be incorrect.

C'est comme une recette :

* Les données sont les ingrédients
* Les notebooks sont les étapes de préparation
* L'application est le plat final
* Les utilitaires sont les outils de cuisine
* Le modèle est le chef qui décide

# — Preparationn du env :

Créer un **environnement de travail isolé** pour ton projet de détection de fraude, afin que toutes les bibliothèques que tu installes n’affectent pas les autres projets sur ton ordinateur.

1-Créer un nouvel environnement "fraud-env" avec Python 3.10

2. **Activer cet environnement (conda activate fraud-env)**

**3. Aller dans le dossier de ton projet : (**cd "C:\Users\marjane\OneDrive\Bureau\COURSES\Projet\_PFA\fraud\_detection\_app")

**4. Installer les outils utiles dans cet environnement :** (conda install pandas numpy scikit-learn matplotlib seaborn streamlit joblib)

# ÉTAPE 3 — Analyse exploratoire (EDA) insights : (voir la chronologie dans le EDA.ipynb file)

Après avoir exploré le dataset, voici les points clés et les variables à considérer pour le clustering K-means :

# ## Observations Clés :

* L'exploration des relations entre les variables numériques (comme le montant et le solde) nous a permis de voir leurs dépendances linéaires.
* Nous avons découvert que la distribution des montants de transaction est significativement influencée par le canal utilisé (Online, ATM, Branch), ce qui est une information clé pour identifier les comportements anormaux.
* L'analyse temporelle a révélé les heures et jours de pointe pour les transactions, nous aidant à repérer les activités qui sortent de ces périodes normales.
* L'étude du temps écoulé entre les transactions est essentielle pour identifier les séquences d'activités très rapides, souvent indicatives de fraude.
* **L'analyse spécifique du temps entre les transactions montre un grand nombre de transactions se produisant à de très courts intervalles (proches de 0 secondes) pour un même compte.**

# ## Variables Potentiellement Utiles pour K-Means

Basé sur les distributions et relations observées lors de l'analyse exploratoire, les variables suivantes (qui nécessiteront un prétraitement comme la normalisation ou l'encodage) semblent les plus prometteuses pour aider l'algorithme K-means à identifier les transactions anormales :

* TransactionAmount (normalisé/transformé pour gérer l'asymétrie et les outliers)
* CustomerAge (normalisé)
* TransactionDuration (normalisé)
* AccountBalance (normalisé/transformé pour gérer l'asymétrie)
* LoginAttempts (normalisé)
* Caractéristiques temporelles dérivées (ex: Hour, DayOfWeek, TimeSinceLastTransaction - après encodage ou normalisation pour les utiliser dans le modèle)
* Variables catégorielles encodées (ex: TransactionType, Channel, CustomerOccupation, et potentiellement un encodage pour Location si sa cardinalité permet une approche pertinente lors du prétraitement)

# ÉTAPE 4 — \_Data\_Preprocessing.ipynb :

## 1-gestion de null values :

In our case they do not exists, but if they do exist we need to choose:

**Gérer les Valeurs Manquantes**

* Reconfirmez la présence et la quantité de valeurs manquantes (df.isnull().sum()).
* Appliquez la stratégie que vous avez choisie pendant l'EDA pour chaque colonne affectée. Les options courantes sont :
* **Suppression :** Si très peu de lignes sont affectées (df.dropna()).
* **Imputation par une valeur statistique :** Moyenne, médiane (souvent préférable pour les données asymétriques), ou mode (SimpleImputer de scikit-learn est utile, ou df.fillna()).
* **Imputation par une valeur constante :** Par exemple, 0 ou une valeur spécifique qui indique un manque.

## 2-Gérer les Outliers (Considérations pour K-Means) :

### Kmeans ?

K-means est une méthode qui permet de **regrouper automatiquement** des données qui se ressemblent, **sans qu’on dise à l’avance ce qui est normal ou anormal**.

Example :

🡺Tu as plusieurs **transactions bancaires**, et tu veux **voir si certaines sont étranges** ou **différentes des autres**.

K-means va faire ceci :

1. **Regarder les transactions** (montant, fréquence, etc.)
2. **Essayer de les regrouper** en plusieurs groupes (par exemple 2 ou 3).
3. Chaque groupe contient des transactions qui **se ressemblent entre elles**.

A screenshot of a black screen

AI-generated content may be incorrect.

K-means va **mettre les 3 normales dans un groupe** et **les 2 bizarres dans un autre**.  
Donc, tu peux dire : "*Les transactions dans le petit groupe sont peut-être frauduleuses*".

### dans cette étape, on a fait ceci :

1. **Le problème :** Certains chiffres importants dans nos données (comme les montants ou les soldes) sont parfois **énormes** par rapport aux autres. Ces chiffres extrêmes (des "outliers") peuvent embrouiller l'outil qui va chercher les groupes normaux (K-means), car il calcule des moyennes.
2. **Pourquoi c'est délicat ici :** Souvent, ces chiffres énormes sont justement des transactions suspectes qu'on veut trouver ! Donc on ne peut pas juste les enlever.
3. **Notre solution :** On a appliqué une astuce mathématique (la "transformation log") aux colonnes avec ces chiffres extrêmes.
4. **Ce que ça fait :** Cette astuce **réduit l'écart** entre les très grands nombres et les nombres moyens. Les nombres énormes deviennent moins "puissants" et ne tirent plus autant la moyenne vers eux.
5. **Le résultat :** On a maintenant de **nouvelles versions** de ces colonnes (comme "montant\*log"). Elles contiennent les mêmes informations, mais les chiffres extrêmes sont "compressés".*

Ça aide K-means à mieux voir les vrais groupes de transactions normales et à identifier les transactions qui sont vraiment différentes, même si elles ont des valeurs très grandes au départ.

## 3-  Créer les Features Dérivées Pertinentes :

Nous avons déjà identifié les principales caractéristiques dérivées potentiellement utiles lors de votre EDA :

1. **Caractéristiques Temporelles :**

* L'heure de la journée (Hour)
* Le jour de la semaine (DayOfWeek)
* Le temps écoulé depuis la transaction précédente pour le même compte (TimeSinceLastTransaction\_s)

### Donc c’est dans cet etape ou en calcule ces colonnes de manière permanente :

Nous avons pris les dates et heures brutes de vos transactions et les avons transformées pour en sortir de nouvelles informations très utiles pour repérer la fraude :

1. **L'heure de la journée :** On a simplement extrait à quelle heure chaque transaction a eu lieu (de minuit à 23h). Savoir si une transaction se passe en pleine journée ou au milieu de la nuit peut être un indice d'anomalie.
2. **Le jour de la semaine :** On a aussi identifié si c'était un lundi, un mardi, etc. (de 0 pour Lundi à 6 pour Dimanche). Certaines fraudes pourraient cibler plus particulièrement les week-ends, par exemple.
3. **Le temps depuis la dernière transaction du même client :** C'est la plus importante. On a regardé l'historique de chaque client séparément, mis ses transactions dans l'ordre, et calculé combien de secondes se sont écoulées entre chaque transaction et celle d'avant. Si un fraudeur utilise une carte volée, il peut essayer de faire plusieurs petites transactions très vite. Repérer les transactions qui arrivent *très peu de temps* après la précédente pour le même client est un signe fort de comportement suspect. Pour la toute première transaction d'un client (qui n'a pas de "transaction précédente"), on a mis une valeur "moyenne" pour ne pas laisser de trou.

En ajoutant ces nouvelles colonnes (Hour, DayOfWeek, TimeSinceLastTransaction\_s) à notre tableau de données, on donne à notre modèle (K-means) beaucoup plus d'indices pour qu'il puisse mieux faire la différence entre un comportement normal (qui se passe à certaines heures/jours, avec des délais raisonnables entre les transactions) et un comportement potentiellement anormal (transactions en pleine nuit, le week-end, ou qui s'enchaînent à toute vitesse). C'est comme lui donner des lunettes pour mieux voir les détails temporels qui comptent pour la fraude.

## 4 : Encoder les Variables Catégorielles.

**Objectif :** Convertir les colonnes qui contiennent du texte (des catégories) en un format numérique que le modèle K-means pourra comprendre et utiliser pour calculer les distances

**La Méthode Principale : One-Hot Encoding**

Elle fonctionne comme ceci :

* Pour chaque colonne catégorielle à encoder (par exemple, Channel avec "ATM", "Online", "Branch"), on crée de nouvelles colonnes.
* Pour chaque catégorie unique (par exemple, "ATM"), on crée une nouvelle colonne nommée Channel\_ATM.
* Dans cette nouvelle colonne Channel\_ATM, la valeur sera 1 si la transaction a eu lieu via "ATM", et 0 sinon.
* On fait la même chose pour "Online" (Channel\_Online) et "Branch" (Channel\_Branch).

Ainsi, une ligne qui avait Channel = "Online" aura maintenant Channel\_ATM = 0, Channel\_Online = 1, Channel\_Branch = 0. Le modèle peut maintenant utiliser ces 0 et 1 comme des nombres.**Variables à Encoder dans votre Dataset :**Basé sur notre EDA et les variables potentiellement utiles, les colonnes catégorielles principales à considérer pour l'encodage sont :

* TransactionType (Debit/Credit)
* Channel (ATM/Online/Branch)
* CustomerOccupation (Student, Doctor, Engineer, Retired)
* Location : Comme discuté, Location a beaucoup de valeurs uniques (haute cardinalité). Faire du One-Hot Encoding direct créerait des centaines ou milliers de nouvelles colonnes, ce qui n'est pas efficace pour K-means et peut même nuire à la performance. Pour ce projet, il est généralement préférable de **ne pas encoder Location directement avec One-Hot Encoding** ou de considérer une approche simplifiée (comme juste le pays si c'était dans les données, mais ici ce sont des villes américaines, toutes dans le même pays). **Nous allons donc encoder les autres colonnes et exclure Location pour l'instant.**

## RES DE CET ETAPE :

Dans cette étape, on a rendu les informations textuelles de notre tableau de données compréhensibles pour le modèle :

* On a pris les colonnes qui contenaient du texte (comme le type de transaction - Débit/Crédit, le canal - Online/ATM/Branch, et la profession du client).
* On a utilisé une technique appelée "One-Hot Encoding" pour les transformer.
* Cette technique a créé de nouvelles colonnes pour chaque option possible (par exemple, une colonne pour "Online" avec 1 si la transaction est en ligne et 0 sinon, et pareil pour "ATM" et "Branch").
* On a fait cela pour le type de transaction, le canal, et la profession.
* À la fin, on a retiré les colonnes texte originales puisque le modèle utilise maintenant les nouvelles colonnes numériques (0s et 1s).

## Étape 5 : Sélection des Features Finales :

**Objectif :** Choisir précisément l'ensemble des colonnes qui seront utilisées comme entrée pour l'algorithme K-means.

on a fait le tri dans toutes les informations de notre tableau pour choisir celles qui vont vraiment servir à notre modèle K-means pour trouver les transactions suspectes :

* On a décidé **quelles colonnes garder** : toutes celles qui décrivent le comportement ou les caractéristiques d'une transaction (les montants, l'âge, la durée, le solde, le nombre de tentatives de connexion, les informations temporelles qu'on a créées comme l'heure/jour et le temps depuis la dernière transaction, ainsi que les catégories qu'on a transformées en nombres 0/1).
* On a aussi décidé **quelles colonnes enlever** : celles qui ne sont pas utiles ou que le modèle ne peut pas utiliser (comme les numéros d'identification uniques des transactions, des comptes, des appareils, les adresses IP, les dates brutes qu'on a déjà traitées, et la localisation parce qu'elle est trop compliquée à utiliser directement).

En résumé, on a créé un nouveau tableau de données (X) qui contient **uniquement les informations pertinentes et numériques**, prêtes pour la mise à l'échelle avant de les donner au modèle.

## Étape 6 : Normaliser/Standardiser les Features Numériques :

**Objectif :** Mettre toutes les variables numériques de X sur une échelle similaire.

**Ce que l'on a fait dans cette étape, en simple :**

* On a pris toutes nos informations numériques (les montants, l'âge, les durées, le solde, les tentatives de connexion, les informations temporelles dérivées, et les 0/1 des colonnes encodées).
* On a appliqué un processus pour **mettre toutes ces informations sur une échelle comparable**. C'est comme si, au lieu d'utiliser les centimètres et les grammes directement, on transformait les chiffres pour qu'ils soient tous "centrés" de la même manière et étalés de façon similaire.
* On a utilisé un outil appelé StandardScaler pour faire ça. Pour chaque type d'information (chaque colonne), cet outil calcule la moyenne et la "dispersion" (l'écart-type) de toutes les valeurs.
* Ensuite, pour chaque valeur (par exemple, la taille de 175 cm), il la transforme en regardant **à quel point elle est éloignée de la moyenne de sa colonne, mesuré en unités de "dispersion" (écart-type)**. Si quelqu'un a une taille très au-dessus de la moyenne, il aura un chiffre positif important. S'il est en dessous de la moyenne, un chiffre négatif. Si sa taille est exactement la moyenne, le chiffre sera 0.
* On fait ça pour la taille *et* pour le poids. Après cette transformation, une personne qui est très grande pour sa population aura un chiffre positif important, et une personne qui pèse très lourd pour sa population aura aussi un chiffre positif important, même si les chiffres de départ (cm et grammes) étaient sur des échelles très différentes.

# ETAPE 5 : Model\_Development.ipynb

**Objectif de la Modélisation :** Utiliser l'algorithme K-means pour regrouper les transactions en fonction de leurs caractéristiques prétraitées, identifier les clusters représentant des comportements inhabituels, et marquer les transactions comme potentiellement frauduleuses

…etapess voir les cellules dans le file.

 l'**Étape 5 : Analyser les Clusters et Pré-identifier les Anomalies** est cruciale. Il s'agit de comprendre ce qui rend ces groupes différents et d'identifier lequel (ou lesquels) d'entre eux représente des transactions inhabituelles qui pourraient être frauduleuses.

**Objectif :** Caractériser chaque cluster trouvé par K-means et déterminer quel(s) cluster(s) présentent des propriétés qui correspondent à notre définition de la fraude (patterns inhabituels).

**Conclusion Simple pour le Choix des Clusters d'Anomalies :**Basé *uniquement* sur ces moyennes de centroïdes :

* Cluster 0 se distingue clairement comme le groupe des **transactions CRÉDIT**.
* Cluster 1 et 2 sont les groupes de **transactions DÉBIT**, différenciés principalement par le **solde moyen** du compte.

Si votre définition de fraude inclut des transactions CRÉDIT suspectes, ou des DÉBITS avec des soldes très bas ou très hauts, alors ces caractéristiques vous aideront à choisir les clusters d'anomalies (anomaly\_cluster\_ids).Il est possible que les vraies anomalies (des transactions individuelles *très* différentes des autres) soient juste des "outliers" au sein de ces grands clusters. K-means les a regroupées avec le groupe le plus "proche". Pour une détection non supervisée avec K-means, il est courant de considérer que les **transactions les plus ÉLOIGNÉES des centroïdes de *tous* les clusters** sont des anomalies, pas seulement celles d'un cluster spécifique. Cependant, l'approche par "cluster(s) d'anomalies" est aussi une façon de faire, surtout si un cluster a des caractéristiques moyennes très extrêmes.Pour la suite (Cellule 7), vous devrez choisir anomaly\_cluster\_ids en fonction des moyennes qui vous semblent les plus "suspectes" d'un point de vue métier dans ce tableau. Si aucune moyenne ne semble *extrêmement* anormale, vous pourriez choisir le(s) plus petit(s) cluster(s) comme potentiels clusters d'anomalies (Cluster 0 et 1 dans votre cas)

**Conclusion & Décision**

**🚩 Cluster 0 → SUSPECT (potentiellement frauduleux)**

* Proportion **élevée de transactions en ligne**
* **Uniquement des transactions de type crédit**
* Comportement typique d’**injection d’argent** ou fraude en ligne
* Profil occupation hétérogène → difficilement traçable

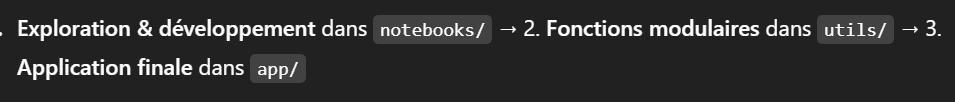
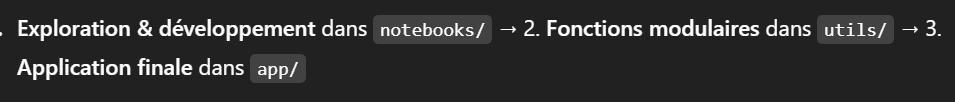
**🚩 Cluster 1 → A SURVEILLER (possiblement anormal)**

* Montants élevés avec **solde très faible**
* Forte présence d’étudiants
* Transactions uniquement de type **débit**
* **Utilisation accrue d’ATM**

**✅ Cluster 2 → Comportement régulier (non suspect)**

* Solde le plus élevé
* Répartition équilibrée des canaux
* Pas d’indicateur fortement suspect

# UTILS/ :

# REALISATION DE L’APP base sur ce model :