

# Introduction :

Avec l'essor des paiements en ligne et l'augmentation des transactions numériques, la fraude par carte bancaire est devenue un problème majeur pour les institutions financières. Selon des études récentes, des milliards de dollars sont perdus chaque année à cause de transactions frauduleuses, mettant en péril non seulement les finances des individus, mais également la confiance dans les systèmes de paiement électronique.

Face à cette menace croissante, les technologies de machine learning offrent des solutions efficaces pour détecter rapidement et précisément les transactions suspectes. Contrairement aux approches traditionnelles basées sur des règles prédéfinies, les algorithmes de machine learning peuvent apprendre des modèles complexes dans les données et s'adapter à l'évolution des comportements frauduleux.

Dans le cadre de ce projet, plusieurs algorithmes de machine learning ont été utilisés pour résoudre le problème de détection des fraudes par carte bancaire. Parmi les modèles explorés figurent la régression logistique, les forêts aléatoires, l’arbre de décision... Ces approches ont été testées sur un dataset issu de **Kaggle**, contenant des transactions anonymisées et réelles.

L’objectif principal est de comparer l’efficacité de ces algorithmes afin d’identifier la solution la plus performante pour détecter les transactions frauduleuses, tout en minimisant les erreurs. Ce rapport retrace les différentes étapes du projet, depuis l’analyse exploratoire des données jusqu’à l’évaluation des modèles, en passant par les techniques de prétraitement et de gestion du déséquilibre des classes.

# Prétraitement du Data Set :

## Description du data set :

Pour ce projet de détection de fraudes par carte bancaire, nous avons utilisé un dataset issu de la plateforme **Kaggle**, intitulé **"Credit Card Fraud Detection"**. Ce dataset est composé de transactions effectuées par des clients sur une période donnée et contient des informations anonymisées pour protéger la confidentialité des utilisateurs.

### Structure des Données :

Le dataset comporte :

* **Nombre de lignes** : 284,807 transactions.
* **Nombre de colonnes** : 31 caractéristiques.
  + **28 colonnes** sont des variables anonymisées (V1 à V28) issues d’une transformation PCA (Principal Component Analysis) appliquée pour réduire la dimensionnalité et préserver la confidentialité.
  + **2 colonnes supplémentaires :**
    - **Time :** Indique le temps écoulé en secondes depuis la première transaction dans l'ensemble de données.
    - **Amount :** Représente le montant de chaque transaction en dollars.
  + **Class :** Colonne cible indiquant si une transaction est frauduleuse (1) ou légitime (0).

### Visualisation Initiale des Classes

Une première analyse de la répartition des classes montre que le dataset est fortement déséquilibré :

* Sur les **284,807** transactions, seulement **492** sont frauduleuses.

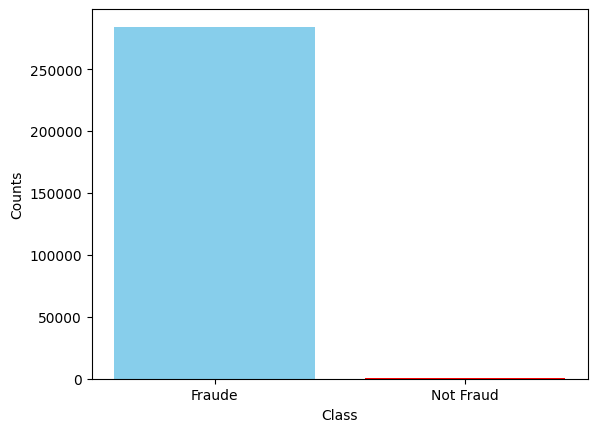
## Analyse Exploratoire des Données :

L'analyse exploratoire des données (EDA) est une étape essentielle pour mieux comprendre la structure et les propriétés du dataset. Cette analyse permet de détecter d'éventuelles anomalies, de mettre en évidence des patterns intéressants et d'orienter les choix de prétraitement et de modélisation.

### Distribution des Classes :

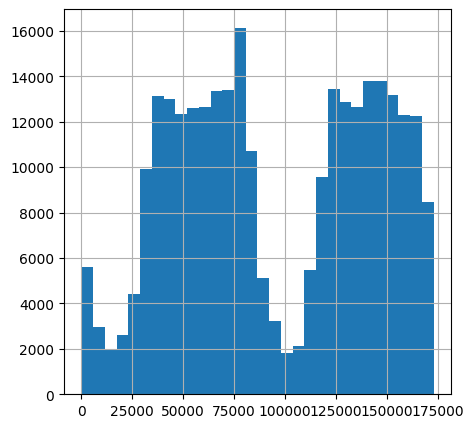
Le dataset présente un fort déséquilibre entre les classes :

* Transactions non-frauduleuses (Class = 0) : **99.83 %** (284,315 instances).
* Transactions frauduleuses (Class = 1) : **0.17 %** (492 instances).



**Figure 1:** Distribution des classes dans l'ensemble de données.

### Analyse des Variables :

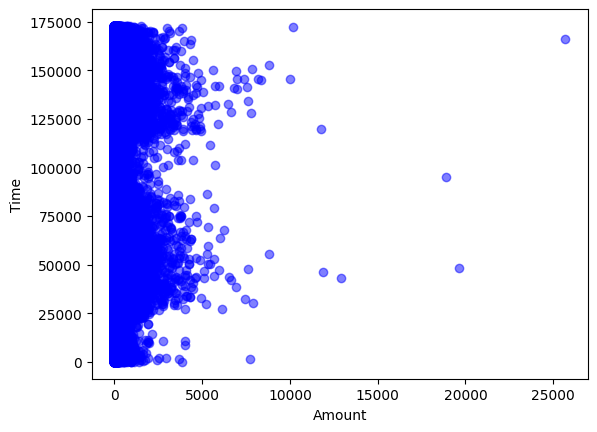


**Variable Time**:

* Cette variable indique le temps écoulé depuis la première transaction.
* La distribution de Time semble relativement uniforme, ce qui suggère que le comportement frauduleux n’est pas limité à un moment spécifique. Cependant, une analyse plus approfondie pourrait révéler des schémas temporels pertinents.

**Variable Amount**:

* Cette variable représente le montant des transactions.
* La majorité des transactions ont un montant faible ou modéré, tandis que quelques transactions présentent des montants exceptionnellement élevés (effet de queue dans la distribution).
* Les transactions frauduleuses tendent à avoir des montants légèrement différents des transactions normales, bien que cela ne soit pas systématique.



Des valeurs aberrantes (**outliers**) ont été identifiées dans les variables **Amount** et certaines variables PCA. Bien que ces valeurs puissent refléter des comportements inhabituels ou frauduleux, leur impact sur les modèles sera étudié dans les étapes ultérieures.

### Variables Anonymisées (V1 à V28) :

Ces variables, issues d'une transformation PCA, sont difficiles à interpréter directement. Cependant, une analyse statistique révèle que les moyennes et médianes des variables sont proches de zéro, ce qui est typique après une transformation PCA.

### Observations Clés:

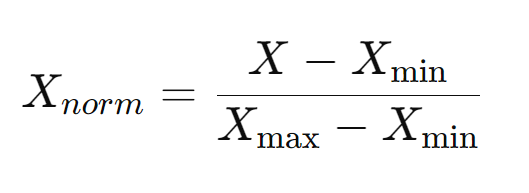
* Le déséquilibre des classes nécessite une stratégie spécifique pour éviter que le modèle privilégie la classe majoritaire.
* Les montants des transactions et la distribution temporelle pourraient être des indicateurs indirects de fraude.
* Les variables anonymisées contiennent probablement les informations les plus significatives, bien que leur interprétation soit limitée.

### Normalisation et Standardisation :

Le prétraitement des données est une étape cruciale dans le machine learning, car il permet de rendre les caractéristiques des données compatibles avec les algorithmes choisis. La normalisation et la standardisation des variables permettent d'améliorer la convergence des modèles et de garantir que les différentes caractéristiques sont sur des échelles comparables.

**Normalisation de la variable Time avec MinMaxScaler**

La variable Time, représentant le temps écoulé depuis la première transaction, a été normalisée à l'aide de la méthode **MinMaxScaler**. Cette technique permet de mettre les valeurs de Time sur une échelle allant de 0 à 1, ce qui est particulièrement utile pour les algorithmes sensibles à l'échelle des variables, tels que les réseaux neuronaux ou les SVM (Support Vector Machines).

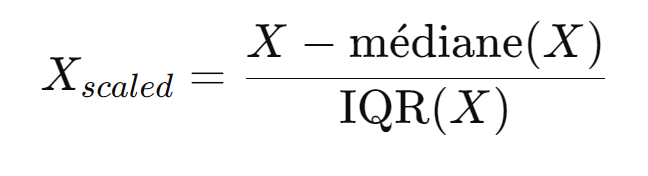
L'application du **MinMaxScaler** a été réalisée selon la formule suivante :

Où :

* **X** est la valeur originale de la variable Time.
* **Xmin** et **Xmax**​ sont respectivement les valeurs minimales et maximales de la variable Time.

**Standardisation de la variable Amount avec RobustScaler**

La variable **Amount**, représentant le montant des transactions, présente une distribution biaisée avec des valeurs aberrantes (**outliers**) qui pourraient affecter la performance des algorithmes de machine learning. En raison de cette caractéristique, nous avons choisi d'appliquer le **RobustScaler**, une méthode de standardisation qui est robuste aux **outliers**. Contrairement au **StandardScaler**, qui centre les données sur la moyenne et les réduit selon l'écart-type, le **RobustScaler** utilise la médiane et l'écart interquartile (IQR) pour la mise à l'échelle, ce qui permet de réduire l'impact des valeurs extrêmes.

La formule du **RobustScaler** est la suivante :

Où :

* **X** est la valeur originale de la variable Amount.
* **mediane(X)** est la médiane des valeurs de Amount.
* **IQR(X)** est l'écart interquartile (c'est-à-dire la différence entre le 75%-centile et le 25%-centile).

### Division des Données:

La division du dataset en ensembles d'entraînement, de validation et de test est une étape clé dans le processus de machine learning. Cette séparation permet de garantir une évaluation fiable des modèles et d'éviter le sur-apprentissage (**overfitting**). Pour ce projet de détection de fraude, nous avons choisi une répartition de **60% pour l'entraînement**, **20% pour la validation** et **20% pour les tests**.

**6.1 Objectifs de la Division des Données**

La division des données a pour but de créer trois ensembles distincts, chacun ayant un objectif spécifique :

* **Ensemble d'entraînement (60%)** : Utilisé pour entraîner le modèle de machine learning. C'est à partir de cet ensemble que le modèle apprend les relations entre les variables d'entrée (features) et la variable cible (fraude ou non fraude).
* **Ensemble de validation (20%)** : Utilisé pour ajuster les hyperparamètres du modèle et sélectionner le meilleur modèle parmi ceux entraînés. Cet ensemble permet de tester les performances du modèle pendant le processus d'entraînement et d'éviter le sur-apprentissage.
* **Ensemble de test (20%)** : Utilisé pour évaluer la performance finale du modèle une fois l'entraînement terminé et les hyperparamètres optimisés. Il est crucial que cet ensemble n'ait pas été utilisé pendant l'entraînement ou la validation, afin de garantir une évaluation impartiale du modèle.

### Gestion du Déséquilibre des Classes :

Le déséquilibre des classes est un problème fréquent dans les problèmes de classification, où l'une des classes (dans notre cas, les transactions frauduleuses) est beaucoup moins représentée que l'autre (transactions légitimes). Ce déséquilibre peut entraîner un modèle qui prédit principalement la classe majoritaire, négligeant ainsi la classe minoritaire, ce qui est particulièrement problématique dans le contexte de la détection de fraude. Pour résoudre ce problème, nous avons choisi d'appliquer la technique de **suréchantillonnage** de la classe minoritaire à l'aide de **SMOTE** (Synthetic Minority Over-sampling Technique).

Dans notre projet, l'application de SMOTE a été réalisée après la séparation du dataset en ensembles d'entraînement et de test, afin de garantir que les données de test restent représentatives de la distribution originale des classes. Les étapes suivantes ont été suivies :

1. **Séparation des données** : Le dataset a été divisé en deux ensembles, un pour l'entraînement (80%) et un pour les tests (20%).
2. **Application de SMOTE** : SMOTE a été appliqué uniquement sur l'ensemble d'entraînement pour créer des échantillons synthétiques de la classe minoritaire (frauduleuse).
3. **Équilibrage des classes** : Après l'application de SMOTE, la distribution des classes dans l'ensemble d'entraînement était plus équilibrée, avec un nombre beaucoup plus élevé de transactions frauduleuses par rapport à la distribution initiale.

# Modélisation du Data Set :

La modélisation est une étape cruciale dans un projet de machine learning. L'objectif est de développer des modèles capables de prédire efficacement les transactions frauduleuses en utilisant les données prétraitées. Dans ce projet, nous avons expérimenté avec plusieurs algorithmes de machine learning afin d'identifier celui qui offre les meilleures performances. Chaque algorithme a été évalué en utilisant des métriques appropriées, comme la **précision**, le **rappel** et le **F1-score.**

## Algorithmes Testés :

### Arbre de décision ( DecisionTreeClassifier ) :

Un arbre de décision est un algorithme de machine learning utilisé pour les tâches de classification et de régression. Il s’agit d’une structure arborescente dans laquelle chaque nœud interne représente un test sur une caractéristique (par exemple, « le montant est-il supérieur à 1000 ? »), chaque branche correspond à un résultat possible du test, et chaque feuille représente une prédiction (par exemple, « transaction frauduleuse » ou « transaction légitime »).

decision\_tree\_model = DecisionTreeClassifier( max\_depth=10,

                                              min\_samples\_split=20,

                                              min\_samples\_leaf=10,

                                              class\_weight="balanced",

                                              random\_state=42,

                                              max\_leaf\_nodes=50,)

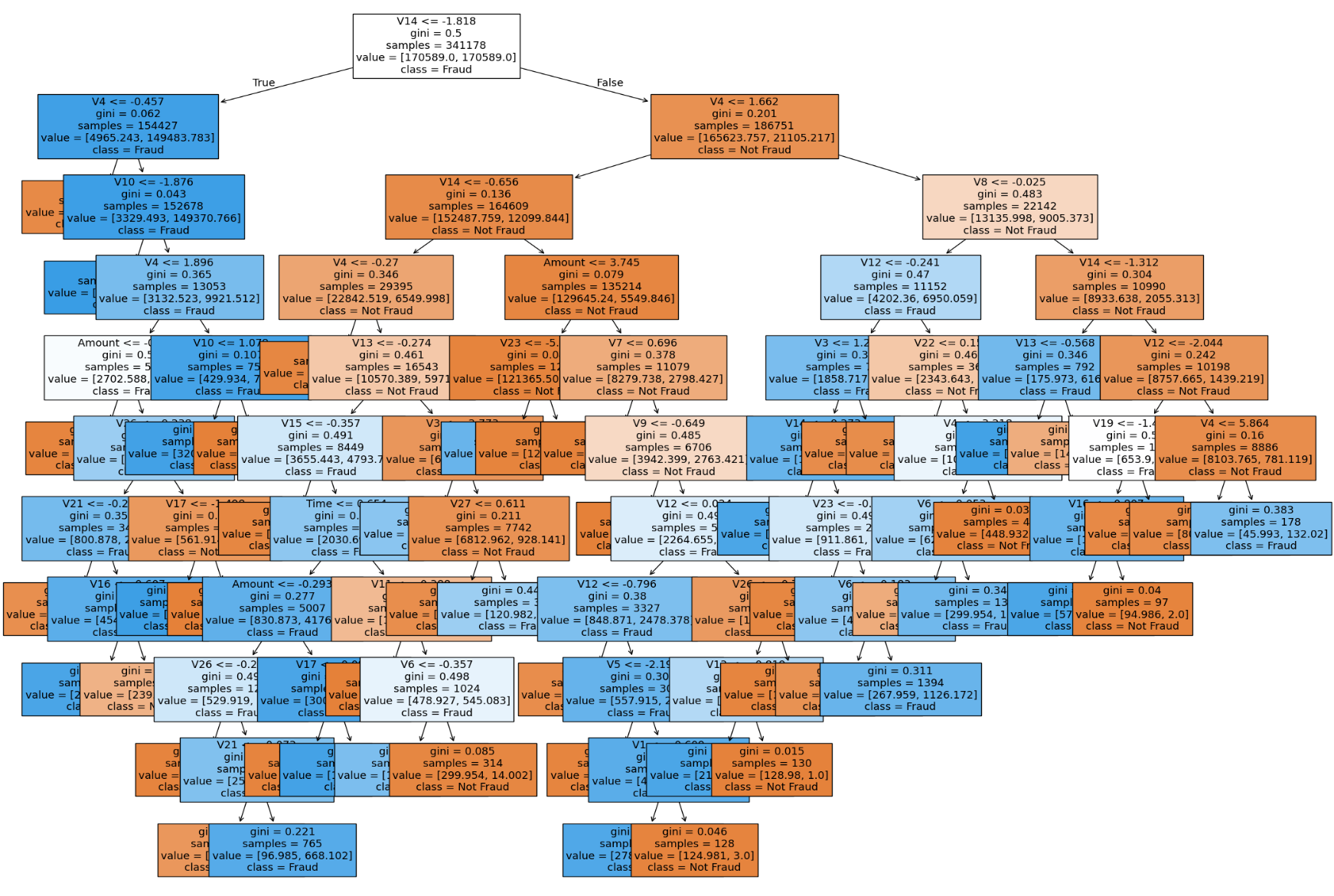
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe** | **Précision** | **Rappel** | **F1-score** | **Support** |
| **Not Fraud** | 0.97 | 0.98 | 0.98 | 57005 |
| **Fraud** | 0.98 | 0.97 | 0.97 | 56721 |

* **Résultat d’algorithme :**

**Les métriques de test du qualité d’’algorithme :**

**La matrice de confusion :**

|  |  |  |
| --- | --- | --- |
| **Classe Prédite / Réelle** | **Précision** | **Rappel** |
| **Not Fraud** | 55,993 | 1,012 |
| **Fraud** | 1,819 | 54,902 |

**L’arbre générée par l’algorithme :**

### Forêt d'arbre de décision (Random Forest) :

La forêt d'arbres de décision, ou **Random Forest**, est un algorithme de machine learning basé sur un ensemble d'arbres de décision. Il combine les prédictions de plusieurs arbres indépendants pour produire une prédiction globale plus robuste et précise.

random\_forest\_model = RandomForestClassifier( max\_depth=10,

                                              min\_samples\_split=20,

                                              min\_samples\_leaf=10,

                                              random\_state=42,

                                              max\_leaf\_nodes=50,)

* **Résultat d’algorithme :**

**Les métriques de test du qualité d’’algorithme :**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe** | **Précision** | **Rappel** | **F1-score** | **Support** |
| **Not Fraud** | 0.96 | 1.00 | 0.98 | 57005 |
| **Fraud** | 0.99 | 0.96 | 0.98 | 56721 |

**La matrice de confusion :**

|  |  |  |
| --- | --- | --- |
| **Classe Prédite / Réelle** | **Précision** | **Rappel** |
| **Not Fraud** | 56731 | 274 |
| **Fraud** | 2490 | 54231 |