Université Moulay Ismaïl
École Normale Supérieure (ENS)
-Meknès-

Master Systèmes intelligents pour L'Éducation

Module : Intelligence artificielle et apprentissage statistique

---

Rapport de Projet : Classification de Transactions de Cartes de Crédit

Machine Learning - Projet de Classification

Réalisé par :
Ahmed BOUBA

Sous la direction de :
Dr. El Arbi ABDELLAOUI ALAOUI (ENS)

Année universitaire
2024/2025

Résumé

Dans ce rapport, nous présentons le développement d'un pipeline complet pour la détection de fraudes sur les transactions de cartes de crédit, une problématique cruciale dans le secteur financier. La démarche adoptée s'articule en plusieurs étapes clés :

1. **Exploration et Analyse des Données (EDA) :**
   - ◇ **Importation et Nettoyage Initial :** Les données brutes ont été importées depuis un fichier CSV. Une première étape de nettoyage a consisté à supprimer la colonne `id` ainsi que les enregistrements incomplets.
   - ◇ **Analyse Exploratoire :** Une analyse détaillée des distributions des variables numériques et de la variable cible `Class` a permis de vérifier l'équilibre des classes et d'identifier la présence de doublons ou d'anomalies.
2. **Prétraitement et Nettoyage des Données :**
   - ◇ **Mélange et Déduplication :** Les données ont été mélangées aléatoirement pour limiter tout biais, puis les doublons ont été éliminés afin de garantir l'indépendance des échantillons.
   - ◇ **Traitement des Outliers :** Une méthode de winsorisation a été appliquée en se basant sur les 1^{er} et $99^{ème}$ percentiles pour limiter l'impact des valeurs aberrantes.
   - ◇ **Normalisation :** Les variables numériques ont été standardisées à l'aide du `StandardScaler` pour homogénéiser les échelles et améliorer la convergence des modèles.
   - ◇ **Séparation des Données :** Le dataset a été divisé en ensembles d'entraînement (80%) et de test (20%) pour permettre une évaluation fiable des modèles.
3. **Modélisation et Évaluation :**
   - ◇ **Entraînement de Modèles :** Trois modèles de machine learning ont été développés et évalués : XGBoost, Random Forest et LightGBM.
   - ◇ **Optimisation et Validation :** Chaque modèle a été optimisé et évalué à l'aide de plusieurs métriques (précision, rappel, F1-score, ROC AUC) et validé par des techniques de cross-validation et l'analyse des courbes d'apprentissage.
   - ◇ **Interprétabilité :** L'importance des variables a été analysée pour mieux comprendre les facteurs influençant la prédiction des modèles.
4. **Déploiement avec Streamlit :**
   - ◇ Le pipeline a été intégré dans une application interactive développée avec Streamlit, permettant aux utilisateurs de visualiser en temps réel les résultats, d'explorer les données et de réaliser des prédictions sur de nouvelles transactions.

Table des matières

1- Introduction

Dans le cadre de la détection de fraudes sur les transactions par carte de crédit, ce projet ambitieux a pour objectif de développer et d'évaluer plusieurs modèles de classification afin d'identifier efficacement les comportements suspects. La méthodologie adoptée repose sur une préparation minutieuse des données, qui est cruciale pour garantir la fiabilité des résultats.

Nous commencerons par une analyse approfondie des données disponibles, suivie de l'application de techniques de prétraitement telles que la winsorisation, qui permet de limiter l'influence des valeurs extrêmes, et la standardisation, qui assure que les différentes caractéristiques des données sont comparables. Ces étapes sont essentielles pour améliorer la qualité des données et, par conséquent, la performance des modèles de machine learning.

Nous prévoyons d'utiliser trois algorithmes de machine learning distincts, sélectionnés en fonction de leur capacité à traiter des ensembles de données déséquilibrés et à s'adapter aux spécificités du problème de détection de fraude. Chaque modèle sera évalué selon des critères rigoureux, tels que la précision, le rappel et la courbe ROC, afin de déterminer celui qui offre les meilleures performances.

Enfin, pour faciliter l'accès et l'utilisation de cet outil, nous déploierons notre solution via Streamlit. Cette plateforme permettra de créer une interface interactive et conviviale, offrant aux utilisateurs la possibilité d'analyser les transactions en temps réel et d'obtenir des résultats instantanés. Grâce à cette approche, nous espérons non seulement améliorer la détection des fraudes, mais aussi fournir un outil pratique et accessible aux professionnels du secteur.

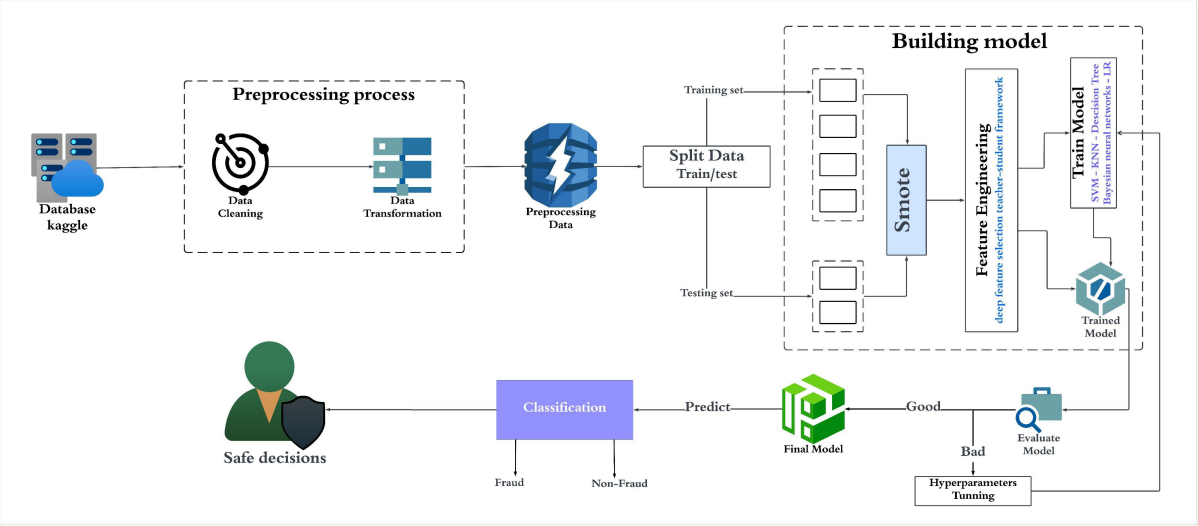


FIGURE 1.1 – Flow chart

2- Analyse et Exploration des Données (EDA)

2.1. Présentation des Données

Les données ont été importées à partir d'un fichier CSV, où nous avons soigneusement supprimé la colonne **id** ainsi que les valeurs manquantes. Pour éviter tout biais, nous avons procédé à un mélange aléatoire des enregistrements, suivi d'une déduplication. Les données utilisées proviennent du jeu de données creditcard_2023 disponible sur Kaggle, qui contient des informations sur des transactions réalisées par carte de crédit. Chaque transaction est classée comme frauduleuse (1) ou non frauduleuse (0).

- ◇ Présence d'une variable cible `Class` indiquant la validité des transactions.
- ◇ Données initialement équilibrées.

Les premières lignes :

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	1.906409	-0.778194	-0.070237	-1.304189	0.214919	0.306562	0.277640	-0.181593	0.119960	1.125153	...	0.081226	1.159525	-0.122638	0.325016	0.313886	0.146423	-0.298505	-0.264953	16237.05	0
1	-0.789265	-0.552857	-0.027642	-2.143462	0.756626	0.760031	0.179347	0.208731	-0.241051	0.751996	...	-0.039317	0.673461	-0.724156	-1.399058	1.876615	0.270871	-0.222962	-0.522283	4784.43	0
2	0.909633	-0.615014	0.321066	-0.530522	0.975265	2.486253	0.123554	0.104894	1.003178	0.529648	...	-0.136065	-0.320693	-0.114162	2.230173	0.841508	-0.970136	-0.163731	-0.000255	8299.76	0
3	1.056051	-0.380402	0.571211	-0.487959	0.252055	0.365181	0.416899	-0.155936	0.387329	0.641589	...	-0.208367	-0.716364	-0.003305	-0.843590	0.417796	0.300822	-0.274909	-0.098318	18646.13	0
4	-0.765730	0.440482	0.038668	0.485665	-0.331725	0.632844	0.135880	-0.770042	2.571246	0.984920	...	0.512723	-0.103687	0.160874	0.917847	-0.850540	-0.952398	-2.581062	-1.544571	7625.62	1

5 rows × 30 columns

2.2. Statistiques Descriptives

Une analyse descriptive des données a été réalisée à l'aide de la méthode `df.describe()`, qui fournit un résumé statistique des variables numériques. Les statistiques suivantes ont été calculées pour chaque caractéristique :

- ◇ **Count** : Nombre total d'observations non manquantes.
- ◇ **Mean** : Moyenne des valeurs.
- ◇ **Std** : Écart-type, mesurant la dispersion des données.
- ◇ **Min** : Valeur minimale observée.
- ◇ **25%** : Premier quartile ($25^{\text{ème}}$ percentile).
- ◇ **50%** : Médiane ($50^{\text{ème}}$ percentile).
- ◇ **75%** : Troisième quartile ($75^{\text{ème}}$ percentile).
- ◇ **Max** : Valeur maximale observée.

Ces statistiques ont permis d'identifier les variables présentant des distributions asymétriques ou des valeurs extrêmes, justifiant l'application de techniques de winsorisation pour limiter l'impact des outliers. Par exemple, des écarts importants entre la moyenne et la médiane, ou des valeurs extrêmes dans les colonnes `Min` et `Max`, ont été observés pour certaines variables.

2.3. Traitement des Valeurs Manquantes et des Doublons

La présence de valeurs manquantes et de doublons dans un dataset peut affecter la qualité des modèles de machine learning. Nous avons donc effectué des vérifications pour identifier ces deux problèmes.

2.3.1. Valeurs Manquantes

Pour vérifier l'intégrité des données, nous avons utilisé la méthode `df.isnull().sum()` pour compter le nombre de valeurs manquantes dans chaque colonne.

```
print(df.isnull().sum())
```

V1	V2	V3	V4	...	V26	V27	V28	Amount	Class
0	0	0	0	0	0	0	0	0	0

TABLE 2.1 – Nombre de valeurs manquantes pour chaque variable

Le résultat montre qu'aucune valeur manquante n'est présente dans le dataset. Cela confirme que les données sont complètes et prêtes pour les étapes de prétraitement et de modélisation.

2.3.2. Doublons

Nous avons également vérifié la présence de doublons dans le dataset en utilisant la méthode `df.duplicated()`.

```
# Affichage du nombre de doublons
nombre_doublons = df.duplicated().sum()
print("Nombre de doublons :", nombre_doublons)
```

Le nombre de doublons dans le dataset est de :

```
Nombre de doublons : 1
```

2.4. Détection et Traitement des Valeurs Aberrantes

Pour identifier les valeurs aberrantes, nous avons utilisé des graphiques en boîte (boxplots). Ces visualisations permettent de repérer les observations situées en dehors des limites définies par les quartiles, qui ont ensuite été traitées par winsorisation pour limiter leur impact sur les modèles.

2.5. Équilibrage des Classes

Dans un problème de classification, l'équilibre des classes est crucial pour éviter les biais dans l'entraînement des modèles. Dans ce dataset, la variable cible `Class` indique si une transaction est frauduleuse (1) ou non (0). Une analyse de la distribution des classes a été réalisée pour vérifier l'équilibre.

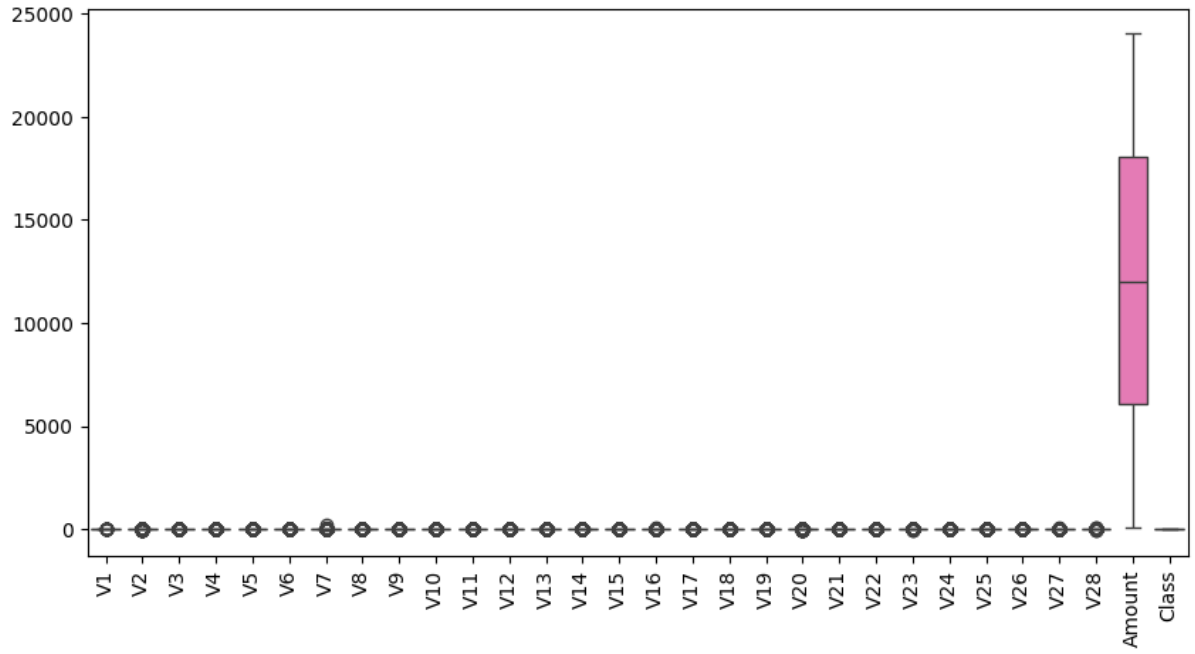


FIGURE 2.1 – Boxplot des Valeurs Aberrantes

```
print(df['Class'].value_counts())

Class
0    284315
1    284315
```

Le résultat montre que les deux classes sont parfaitement équilibrées, avec 284 315 transactions pour chaque classe (fraude et non-fraude). Cela élimine la nécessité d'appliquer des techniques de rééchantillonnage telles que le suréchantillonnage (oversampling) ou le sous-échantillonnage (undersampling).

3- Prétraitement et Nettoyage des Données

Le prétraitement des données est une étape cruciale pour garantir la qualité des données et améliorer les performances des modèles de machine learning. Voici les étapes clés suivies dans ce projet :

3.1. Importation et Nettoyage Initial :

- ◇ Les données ont été importées à partir d'un fichier CSV à l'aide de la bibliothèque `pandas`.
- ◇ La colonne `id` a été supprimée car elle ne fournit aucune information utile pour la modélisation.
- ◇ Les valeurs manquantes ont été vérifiées et supprimées pour garantir l'intégrité des données.

```
file_path = 'data/raw/creditcard_2023.csv'
df = pd.read_csv(file_path).drop(columns=['id']).dropna()
df = shuffle(df)
df = df.reset_index(drop=True)
```

3.2. Mélange et Déduplication :

- ◇ Les données ont été mélangées aléatoirement pour éviter tout biais lié à l'ordre des enregistrements.
- ◇ Les doublons ont été identifiés et supprimés pour garantir l'indépendance des échantillons. Initialement, 1 doublon a été détecté, et après nettoyage, le dataset ne contient plus de doublons.

```
df = shuffle(df)
df = df.reset_index(drop=True)
df = df.drop_duplicates()
```

3.3. Traitement des Outliers :

- ◇ Les valeurs aberrantes ont été traitées à l'aide de la winsorisation, une technique qui remplace les valeurs extrêmes par les percentiles 1^{er} et $99^{ème}$. Cette méthode permet de limiter l'impact des outliers sans les supprimer complètement.
- ◇ Les bornes de winsorisation ont été calculées pour chaque variable numérique et sauvegardées dans un fichier `winsorization_bounds.pkl` pour une utilisation future.

```
winsorization_bounds = {}
for col in num_features:
    lower = df[col].quantile(0.01)
    upper = df[col].quantile(0.99)
    winsorization_bounds[col] = (lower, upper)

joblib.dump(winsorization_bounds, "models/winsorization_bounds.pkl")

for col, (lower, upper) in winsorization_bounds.items():
    df[col] = df[col].clip(lower=lower, upper=upper)
```

3.4. Séparation des Données :

- ◇ Le dataset a été divisé en ensembles d'entraînement (80%) et de test (20%) pour permettre une évaluation fiable des modèles.
- ◇ Les ensembles d'entraînement et de test ont été sauvegardés dans des fichiers CSV distincts (`X_train.csv`, `X_test.csv`, `y_train.csv`, `y_test.csv`) pour une utilisation ultérieure.

```
X = df.drop(columns=['Class'])
y = df['Class']
X_train, X_test, y_train, y_test = train_test_split(
                                    X, y, test_size=0.2, random_state=42)
```

3.5. Normalisation :

- ◇ Les variables numériques ont été standardisées à l'aide de `StandardScaler` pour garantir que toutes les caractéristiques soient sur la même échelle. Cela améliore la convergence des modèles de machine learning.
- ◇ Le scaler a été sauvegardé dans un fichier `scaler.pkl` pour être réutilisé lors de la prédiction sur de nouvelles données.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

joblib.dump(scaler, "models/scaler.pkl")
```

Ces étapes de prétraitement ont permis de préparer un dataset propre, équilibré et prêt pour l'entraînement des modèles de machine learning.

4- Modélisation

4.1. Choix des Modèles

Pour ce projet de détection de fraudes, trois modèles de machine learning ont été sélectionnés en raison de leur efficacité avérée dans les problèmes de classification, en particulier pour les grands volumes de données. Les modèles choisis sont :

- **XGBoost (eXtreme Gradient Boosting)** : Un algorithme de boosting basé sur les arbres de décision, connu pour sa rapidité, sa précision et sa capacité à gérer des données complexes.
- **Random Forest** : Un ensemble d'arbres de décision qui combine les prédictions de plusieurs modèles pour améliorer la robustesse et réduire le surajustement (overfitting).
- **LightGBM (Light Gradient Boosting Machine)** : Une variante de boosting optimisée pour la vitesse et l'efficacité mémoire, particulièrement adaptée aux grands datasets.

Ces modèles ont été choisis pour leur capacité à traiter des données structurées, leur interprétabilité (via l'importance des variables) et leur performance en termes de précision et de rappel. De plus, comme les données sont parfaitement équilibrées (50% de fraudes et 50% de non-fraudes), il n'a pas été nécessaire d'appliquer des techniques de rééchantillonnage telles que le suréchantillonnage (oversampling) ou le sous-échantillonnage (undersampling).

4.2. Entraînement des Modèles

Les modèles ont été entraînés sur l'ensemble d'entraînement (80% des données) et évalués sur l'ensemble de test (20% des données). Les étapes suivantes ont été suivies pour chaque modèle :

Les modèles ont été entraîné avec les hyperparamètres par défaut pour établir une baseline.

4.2.1. XGBoost

```
model = XGBClassifier(random_state=42)
model.fit(X_train, y_train)
```

4.2.2. Random Forest

```
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
```

4.2.3. LightGBM

- ◇ **Hyperparamètres** : Le modèle a été configuré avec boosting_type='gbdt', num_leaves=31, learning_rate=0.05, et n_estimators=500.

```
lgbm_model = LGBMClassifier(
    boosting_type='gbdt',
    num_leaves=31,
    learning_rate=0.05,
    n_estimators=500,
    random_state=42
)
lgbm_model.fit(X_train, y_train)
```

5- Évaluation des Modèles

L'évaluation des modèles de machine learning est une étape cruciale pour s'assurer de la performance et de la robustesse des modèles sélectionnés. Nous avons utilisé plusieurs métriques pour évaluer les performances des modèles XGBoost, Random Forest et LightGBM, en plus d'analyser la matrice de confusion, les importances des caractéristiques, la validation croisée et les courbes d'apprentissage.

5.1. Métriques d'Évaluation

Les principales métriques d'évaluation que nous avons utilisées pour mesurer la performance des modèles sont l'**accuracy**, la **précision**, le **rappel**, le **score F1**, et le **ROC AUC**. Voici les résultats détaillés pour chaque modèle :

XGBoost :

Accuracy	Précision	Rappel	F1 Score	ROC AUC
0.9997	0.9994	1.0000	0.9997	1.0000

TABLE 5.1 – Résumé des performances du modèle XGBoost

Random Forest

Accuracy	Précision	Rappel	F1 Score	ROC AUC
0.9999	0.9998	1.0000	0.9999	1.0000

TABLE 5.2 – Performances du modèle Random Forest

LightGBM

Accuracy	Précision	Rappel	F1 Score	ROC AUC
0.9997	0.9995	0.9999	0.9997	0.9999

TABLE 5.3 – Performances du modèle LightGBM

Les modèles présentent des résultats exceptionnels avec des scores proches de 1 sur toutes les métriques, ce qui indique une performance de très haute qualité.

5.2. Matrice de Confusion

La **matrice de confusion** nous permet de visualiser les performances des modèles en termes de vrais positifs (TP), faux positifs (FP), vrais négatifs (TN) et faux négatifs (FN). Voici les matrices de confusion pour chaque modèle :

Matrice de Confusion XGBoost :

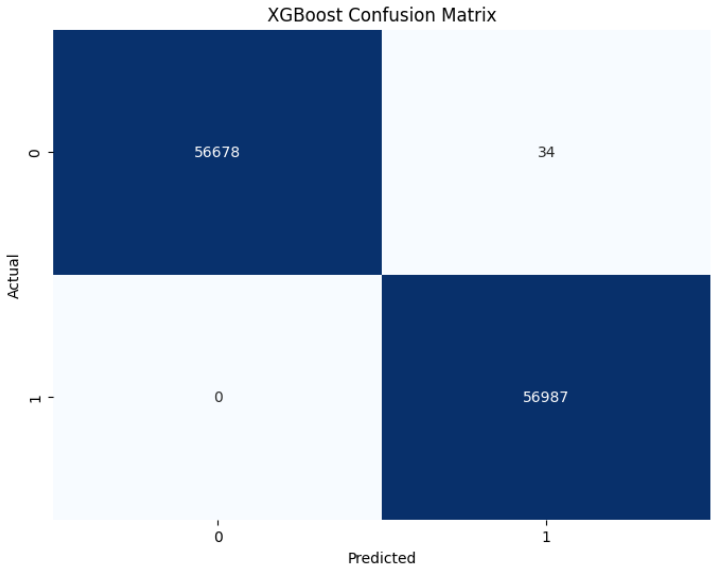


FIGURE 5.1 – Matrice de Confusion - XGBoost

Matrice de Confusion Random Forest :

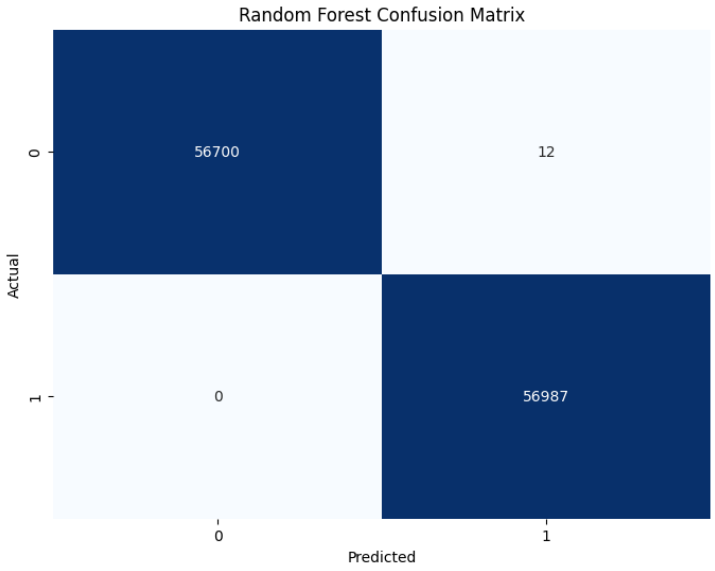


FIGURE 5.2 – Matrice de Confusion - Random Forest

Matrice de Confusion LightGBM :

Ces matrices montrent des résultats très satisfaisants, avec des faux positifs et faux négatifs quasi inexistants.

5.3. Comparaison des Modèles

Nous avons comparé les performances des trois modèles à l'aide de **l'accuracy**, de la **précision**, du **rappel**, du **score F1** et du **ROC AUC**. Les résultats indiquent que **Random Forest** obtient la meilleure **accuracy** avec 0.9999, suivi de très près par **XGBoost** et **LightGBM** (tous deux autour de 0.9997).

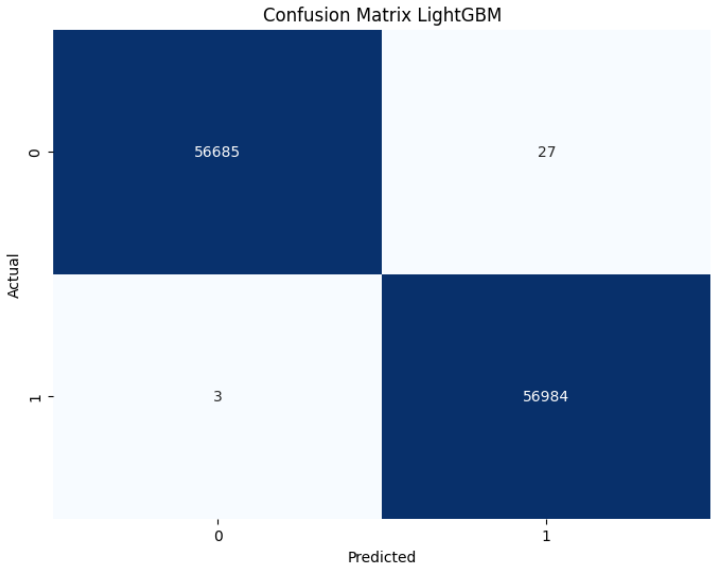


FIGURE 5.3 – Matrice de Confusion - LightGBM

Modèle	Accuracy	Précision	Rappel	F1 Score
XGBoost	0.9997	0.9994	1.0000	0.9997
Random Forest	0.9999	0.9998	1.0000	0.9999
LightGBM	0.9997	0.9995	0.9999	0.9997

TABLE 5.4 – Tableau récapitulatif des performances des modèles

Les trois modèles sont très proches en termes de performance, mais **Random Forest** semble légèrement supérieur en termes d'accuracy.

5.4. Feature Importances

L'analyse des **importances des caractéristiques** permet de déterminer quelles variables influencent le plus les prédictions des modèles. Voici un exemple d'importance des caractéristiques pour le modèle **Random Forest** :

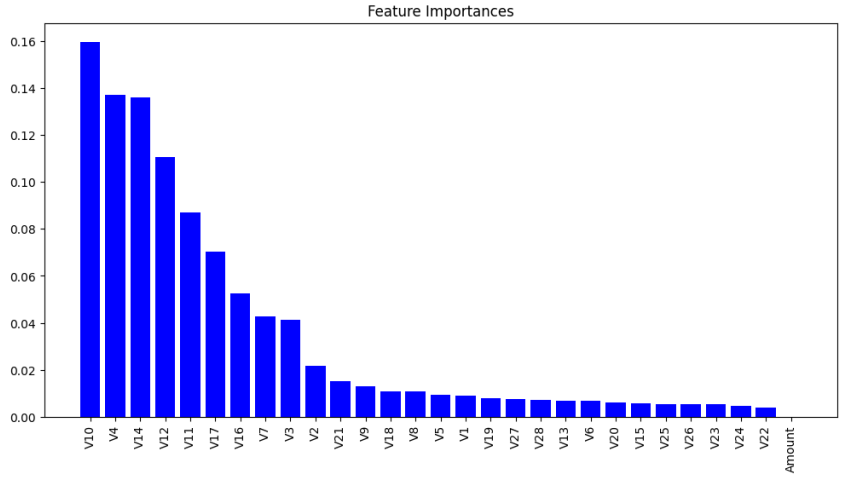


FIGURE 5.4 – Graphique des importances des caractéristiques

Ces valeurs montrent que certaines variables sont plus influentes que d'autres dans les décisions du modèle.

5.5. Cross-Validation

La **validation croisée** est utilisée pour évaluer la robustesse des modèles en les testant sur plusieurs sous-ensembles des données. Pour le modèle **Random Forest**, voici les scores de validation croisée obtenus :

```
0.99975813, 0.99992304, 0.99985708, 0.99983509, 0.99987907
```

La moyenne de la précision de la validation croisée est de **0.9999**, ce qui confirme la stabilité et la généralisation du modèle.

5.6. Courbes d'Apprentissage

Les **courbes d'apprentissage** sont utilisées pour visualiser l'évolution de la performance du modèle au fur et à mesure que le nombre d'exemples d'entraînement augmente. Elles montrent également si un modèle sur-apprend ou sous-apprend.

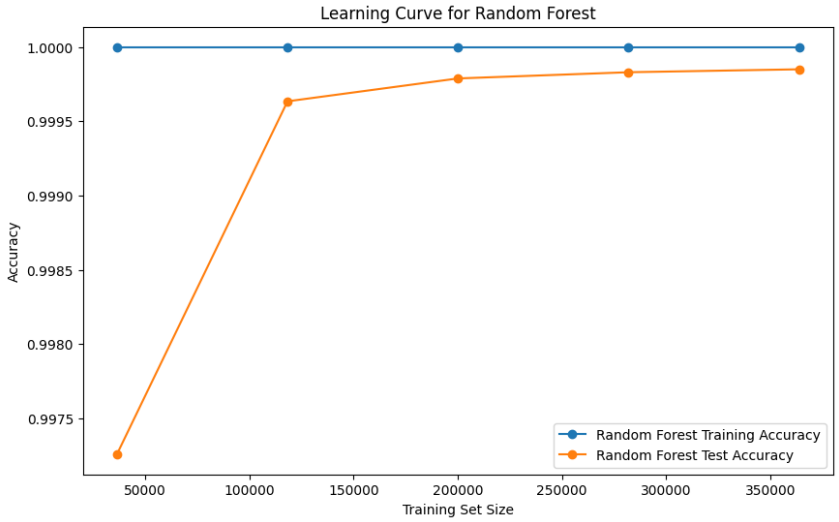


FIGURE 5.5 – Courbe d'apprentissage Random Forest

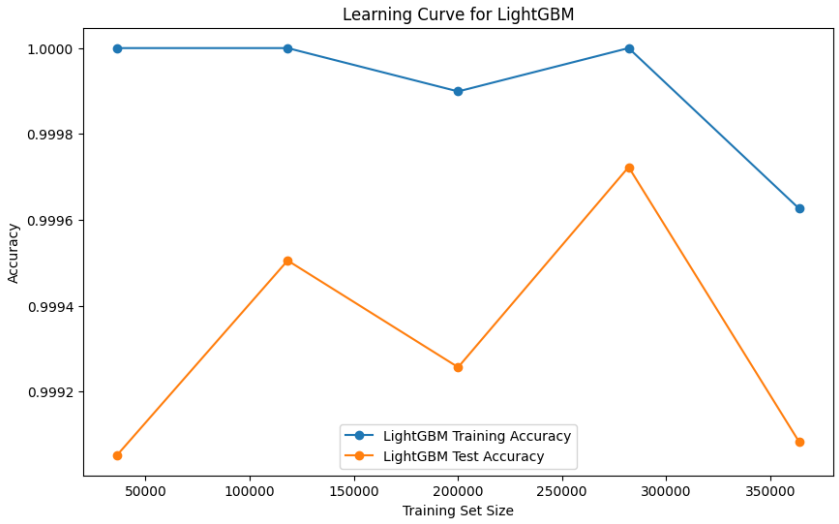


FIGURE 5.6 – Courbe d'apprentissage LightGBM

6- Déploiement avec Streamlit

Pour rendre le modèle accessible et interactif, une application web a été développée à l'aide de Streamlit, une bibliothèque Python permettant de créer des interfaces utilisateur simples et efficaces. Cette application permet aux utilisateurs de visualiser les résultats, d'explorer les données et de réaliser des prédictions sur de nouvelles transactions en temps réel.

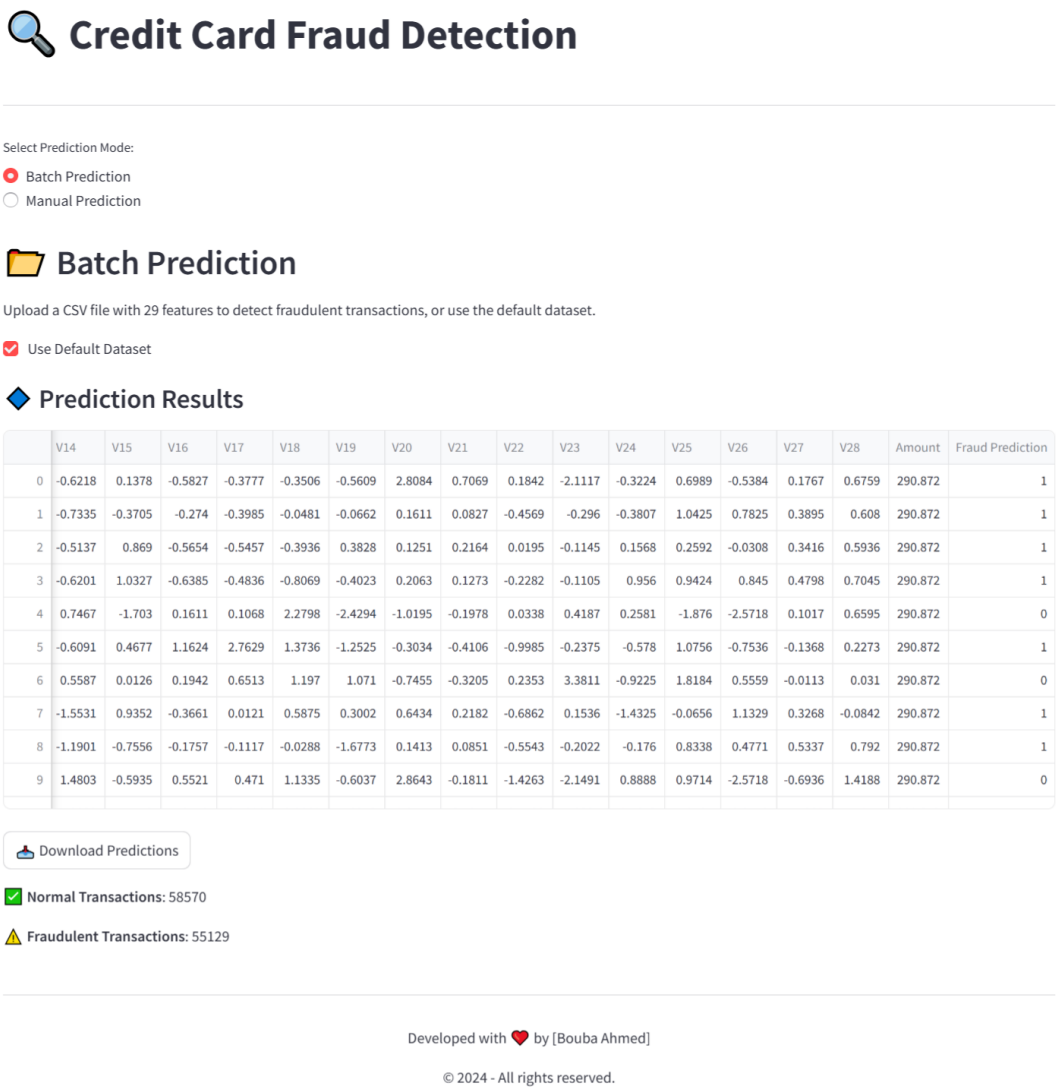
6.1. Fonctionnalités de l'Application

L'application Streamlit propose les fonctionnalités suivantes :

- ◇ **Visualisation des Données** : Les utilisateurs peuvent explorer les distributions des variables, les statistiques descriptives et les corrélations entre les caractéristiques.
- ◇ **Prédictions en Temps Réel** : Les utilisateurs peuvent saisir les détails d'une transaction (montant, caractéristiques V1 à V28, etc.) et obtenir une prédiction instantanée (fraude ou non-fraude).
- ◇ **Analyse des Résultats** : Les performances des modèles (matrices de confusion, courbes ROC, métriques de classification) sont affichées de manière interactive.
- ◇ **Importation de Données** : Les utilisateurs peuvent télécharger un fichier CSV contenant plusieurs transactions pour obtenir des prédictions en batch.

6.2. Avantages de Streamlit

- ◇ **Facilité d'Utilisation** : Streamlit permet de créer des applications web avec un minimum de code, sans nécessiter de connaissances approfondies en développement web.
- ◇ **Interactivité** : Les widgets de Streamlit (sliders, boutons, etc.) rendent l'application dynamique et conviviale.
- ◇ **Intégration Simple** : Les modèles entraînés et les artefacts de prétraitement (scaler, winsorisation) sont intégrés directement dans l'application.



	V14	V15	V16	V17	V18	V19	V20	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Fraud Prediction
0	-0.6218	0.1378	-0.5827	-0.3777	-0.3506	-0.5609	2.8084	0.7069	0.1842	-2.1117	-0.3224	0.6989	-0.5384	0.1767	0.6759	290.872	1
1	-0.7335	-0.3705	-0.274	-0.3985	-0.0481	-0.0662	0.1611	0.0827	-0.4569	-0.296	-0.3807	1.0425	0.7825	0.3895	0.608	290.872	1
2	-0.5137	0.869	-0.5654	-0.5457	-0.3936	0.3828	0.1251	0.2164	0.0195	-0.1145	0.1568	0.2592	-0.0308	0.3416	0.5936	290.872	1
3	-0.6201	1.0327	-0.6385	-0.4836	-0.8069	-0.4023	0.2063	0.1273	-0.2282	-0.1105	0.956	0.9424	0.845	0.4798	0.7045	290.872	1
4	0.7467	-1.703	0.1611	0.1068	2.2798	-2.4294	-1.0195	-0.1978	0.0338	0.4187	0.2581	-1.876	-2.5718	0.1017	0.6595	290.872	0
5	-0.6091	0.4677	1.1624	2.7629	1.3736	-1.2525	-0.3034	-0.4106	-0.9985	-0.2375	-0.578	1.0756	-0.7536	-0.1368	0.2273	290.872	1
6	0.5587	0.0126	0.1942	0.6513	1.197	1.071	-0.7455	-0.3205	0.2353	3.3811	-0.9225	1.8184	0.5559	-0.0113	0.031	290.872	0
7	-1.5531	0.9352	-0.3661	0.0121	0.5875	0.3002	0.6434	0.2182	-0.6862	0.1536	-1.4325	-0.0656	1.1329	0.3268	-0.0842	290.872	1
8	-1.1901	-0.7556	-0.1757	-0.1117	-0.0288	-1.6773	0.1413	0.0851	-0.5543	-0.2022	-0.176	0.8338	0.4771	0.5337	0.792	290.872	1
9	1.4803	-0.5935	0.5521	0.471	1.1335	-0.6037	2.8643	-0.1811	-1.4263	-2.1491	0.8888	0.9714	-2.5718	-0.6936	1.4188	290.872	0

FIGURE 6.1 – Application Streamlit(Interface de prediction)

7- Résumé des Résultats

Ce projet de détection de fraudes sur les transactions de cartes de crédit a permis de développer et d'évaluer trois modèles de machine learning : XGBoost, Random Forest et LightGBM. Les résultats obtenus sont exceptionnels, avec des performances proches de la perfection sur l'ensemble de test. Voici un résumé des principaux résultats :

Modèle	Accuracy	Précision	Rappel	F1 Score
XGBoost	0.9997	0.9994	1.0000	0.9997
Random Forest	0.9999	0.9998	1.0000	0.9999
LightGBM	0.9997	0.9995	0.9999	0.9997

TABLE 7.1 – Tableau récapitulatif des performances des modèles

Les trois modèles se sont révélés particulièrement efficaces pour différencier les transactions frauduleuses des transactions légitimes, comme l'indiquent les matrices de confusion qui montrent un faible nombre de faux positifs et de faux négatifs. L'équilibre des classes dans le jeu de données a joué un rôle crucial dans ces performances, rendant superflues les techniques de rééchantillonnage. L'évaluation des modèles a mis en évidence que XGBoost, Random Forest et LightGBM offrent des résultats exceptionnels, avec des scores proches de 1 pour toutes les métriques essentielles. Bien que le modèle Random Forest ait légèrement devancé les autres en termes de précision, les différences sont marginales. Les courbes d'apprentissage révèlent que tous les modèles s'adaptent efficacement aux données sans risque de surapprentissage, et la validation croisée a attesté de la stabilité et de la robustesse de ces modèles.

7.1. Perspectives d'Amélioration

Bien que les résultats soient déjà excellents, plusieurs pistes d'amélioration peuvent être explorées pour renforcer encore la robustesse et l'efficacité du système de détection de fraudes :

- **Optimisation des Hyperparamètres** : Une recherche systématique des hyperparamètres (via GridSearchCV ou RandomizedSearchCV) pourrait améliorer les performances des modèles, en particulier pour XGBoost et LightGBM.
- **Ensemble Learning** : Combiner les prédictions des trois modèles (par exemple, en utilisant un vote majoritaire ou un stacking) pourrait augmenter la robustesse et la précision globale.
- **Amélioration de l'Interface Utilisateur** : Enrichir l'application Streamlit avec des visualisations interactives et des explications détaillées des prédictions pour une meilleure expérience utilisateur.

En conclusion, ce projet a démontré l'efficacité des modèles de machine learning pour la détection de fraudes, tout en ouvrant la voie à des améliorations futures pour renforcer encore le système.

Références

Les références suivantes ont été utilisées pour la réalisation de ce projet :

- **Bibliothèques Python :**
  - `pandas` : Documentation officielle pour la manipulation des données. `https://pandas.pydata.org/docs/`
  - `numpy` : Documentation officielle pour les calculs numériques. `https://numpy.org/doc/`
  - `scikit-learn` : Documentation officielle pour les modèles de machine learning et les outils de prétraitement. `https://scikit-learn.org/stable/`
  - `xgboost` : Documentation officielle pour l'implémentation de XGBoost. `https://xgboost.readthedocs.io/`
  - `lightgbm` : Documentation officielle pour l'implémentation de LightGBM. `https://lightgbm.readthedocs.io/`
  - `matplotlib` et `seaborn` : Documentation officielle pour la visualisation des données. `https://matplotlib.org/stable/contents.html` et `https://seaborn.pydata.org/`
  - `streamlit` : Documentation officielle pour la création d'applications web interactives. `https://docs.streamlit.io/`
- **Ressources :**
  - Kaggle : Dataset "Credit Card Fraud Detection" utilisé pour ce projet. creditcard_2023