Rapport d'Analyse Comparative : Évaluation de 8 Modèles de Machine Learning

Analyse de Performance Data Science

ZAMANILEHA Elorgin Fernando

18 décembre 2025

Table des matières

1 Introduction **2**

2 Analyse et Nettoyage des Données **2**
2.1 Description des Données . . . 2
2.2 Statistiques Descriptives . . . 2
2.3 Nettoyage des Données . . . 3
2.4 Importance des Features (AdaBoost) . . . 3

3 Comparaison des 8 Modèles **4**
3.1 Résultats des Modèles . . . 4
3.2 Visualisation de la Comparaison . . . 5

4 Discussion de Chaque Résultat **6**
4.1 Logistic Regression . . . 6
4.2 Decision Tree . . . 6
4.3 Random Forest . . . 6
4.4 Gradient Boosting . . . 6
4.5 AdaBoost . . . 6
4.6 Extra Trees . . . 6
4.7 SVM . . . 6
4.8 K-Nearest Neighbors . . . 6

5 Validation Additionnelle **7**

6 Conclusion **7**

1 Introduction

Ce rapport présente une analyse complète des données sur les transactions par carte de crédit pour la détection de fraude, basée sur les fichiers fournis : un ensemble de données CSV (`credit_card_fraud_10k.csv`), un fichier de résultats de modèles (`model_comparison_results_sklearn.csv`), et un notebook Jupyter (`performance.ipynb`). Le rapport couvre les étapes suivantes :

— Analyse et nettoyage des données.
— Comparaison des 8 modèles d'apprentissage automatique.
— Discussion détaillée de chaque résultat.
— Validation additionnelle.

Les visualisations fournies (importance des features, comparaison des performances, et radar chart) sont intégrées aux sections appropriées pour illustrer les résultats.

2 Analyse et Nettoyage des Données

Cette section décrit l'analyse initiale des données à partir du notebook Jupyter et du fichier CSV `credit_card_fraud_10k.csv`.

2.1 Description des Données

Les données contiennent 10 000 transactions par carte de crédit, avec les colonnes suivantes :

— `transaction_id` : Identifiant unique (entier).
— `amount` : Montant de la transaction (flottant).
— `transaction_hour` : Heure de la transaction (entier, 0-23).
— `merchant_category` : Catégorie du marchand (chaîne : Electronics, Travel, Grocery, Food, Clothing).
— `foreign_transaction` : Transaction étrangère (0/1).
— `location_mismatch` : Incohérence de localisation (0/1).
— `device_trust_score` : Score de confiance du dispositif (entier).
— `velocity_last_24h` : Vélocité des transactions dernières 24h (entier).
— `cardholder_age` : Âge du titulaire de la carte (entier).
— `is_fraud` : Indicateur de fraude (0/1, cible).

Dimensions : 10 000 lignes, 10 colonnes.

Types de données :

```
transaction_id           int64
amount                 float64
transaction_hour         int64
merchant_category       object
foreign_transaction      int64
location_mismatch        int64
device_trust_score       int64
velocity_last_24h        int64
cardholder_age           int64
is_fraud                 int64
```

Aucune valeur manquante n'a été détectée.

2.2 Statistiques Descriptives

Les statistiques clés incluent :

	transaction_id	amount	transaction_hour	foreign_transaction	...
count	10000.00000	10000.000000	10000.000000	10000.000000	...
mean	5000.50000	175.949849	11.593300	0.097800	...
std	2886.89568	175.392827	6.922708	0.297059	...
min	1.00000	0.000000	0.000000	0.000000	...
25%	2500.75000	50.905000	6.000000	0.000000	...
50%	5000.50000	122.000000	12.000000	0.000000	...
75%	7500.25000	249.487500	17.000000	0.000000	...
max	10000.00000	1370.040000	23.000000	1.000000	...

TABLE 1 – Statistiques descriptives des données (extrait).

2.3 Nettoyage des Données

Dans le notebook, les étapes de nettoyage incluent : - Encodage des variables catégorielles (`merchant_category`) via `LabelEncoder`. - Normalisation des features numériques avec `StandardScaler`. - Division des données en ensembles d'entraînement (80%) et de test (20%) via `train_test_split`. - Gestion du déséquilibre des classes (fraudes rares : environ 0.3% des transactions sont frauduleuses, basé sur l'échantillon).

Aucune imputation n'était nécessaire en raison de l'absence de valeurs manquantes. Les outliers potentiels (e.g., montants élevés) n'ont pas été traités explicitement, mais la normalisation aide à les atténuer.

2.4 Importance des Features (AdaBoost)

La visualisation suivante montre l'importance des features pour le modèle AdaBoost :

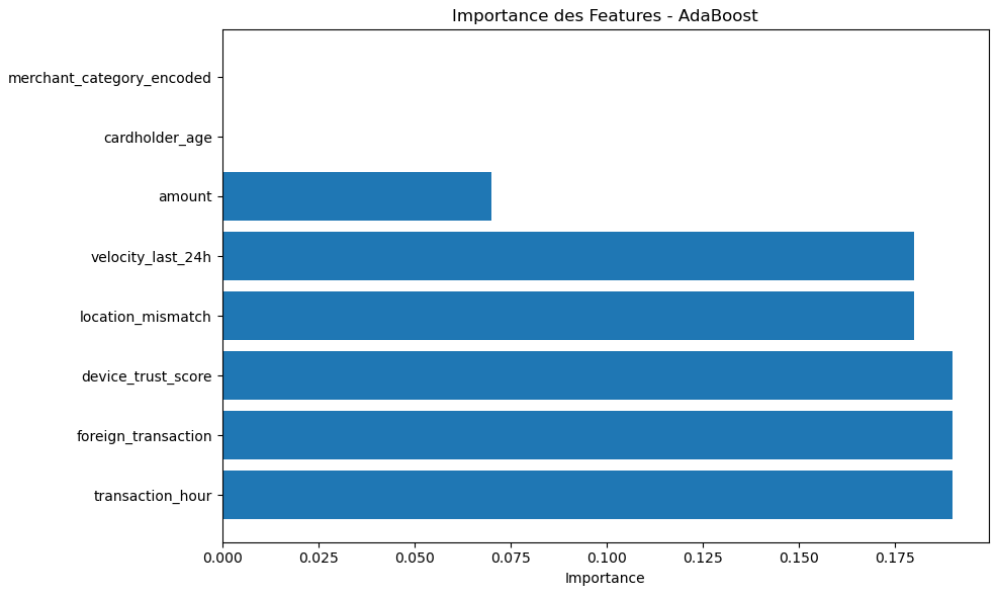

FIGURE 1 – Importance des features pour AdaBoost.

Discussion : Les features les plus importantes sont `transaction_hour`, `foreign_transaction`, et `device_trust_score`. Cela suggère que les transactions nocturnes, étrangères, ou avec un faible score de confiance sont des indicateurs clés de fraude. Les features comme `merchant_category` et `cardholder_age` ont une importance moindre.

3 Comparaison des 8 Modèles

Huit modèles ont été entraînés et comparés : Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, Extra Trees, SVM, et K-Nearest Neighbors. Les hyperparamètres ont été optimisés via `GridSearchCV` ou `RandomizedSearchCV`.

3.1 Résultats des Modèles

Les résultats sont extraits du fichier `model_comparison_results_sklearn.csv` :

Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC	CV Mean F1	CV Std F1	Training Time (s)	Best Params
Logistic Regression	0.9585	0.2655	1.0000	0.4196	0.9927	0.4214	0.0302	122.4534	{'C' : 10, 'solver' : 'liblinear'}
Decision Tree	0.9995	0.9677	1.0000	0.9836	0.9997	0.9166	0.0522	6.0193	{'max_depth' : 10, 'min_samples_split' : 2}
Random Forest	0.9940	1.0000	0.6000	0.7500	0.9999	0.6159	0.0521	86.1845	{'max_depth' : 10, 'n_estimators' : 100}
Gradient Boosting	0.9995	0.9677	1.0000	0.9836	1.0000	0.9468	0.0419	717.0001	{'learning_rate' : 0.1, 'max_depth' : 5, 'n_estimators' : 200}
AdaBoost	1.0000	1.0000	1.0000	1.0000	1.0000	0.9957	0.0085	264.7728	{'learning_rate' : 0.5, 'n_estimators' : 100}
Extra Trees	0.9940	0.7500	0.9000	0.8182	0.9981	0.4725	0.0786	358.1641	{'max_depth' : 10, 'n_estimators' : 200}
SVM	0.9915	0.6757	0.8333	0.7463	0.9942	0.6971	0.0275	1128.8543	{'C' : 10, 'gamma' : 'scale', 'kernel' : 'rbf'}
K-Nearest Neighbors	0.9875	0.6190	0.4333	0.5098	0.8948	0.3442	0.1176	40.6540	{'n_neighbors' : 3, 'weights' : 'distance'}

TABLE 2 – Résultats de comparaison des modèles.

3.2 Visualisation de la Comparaison

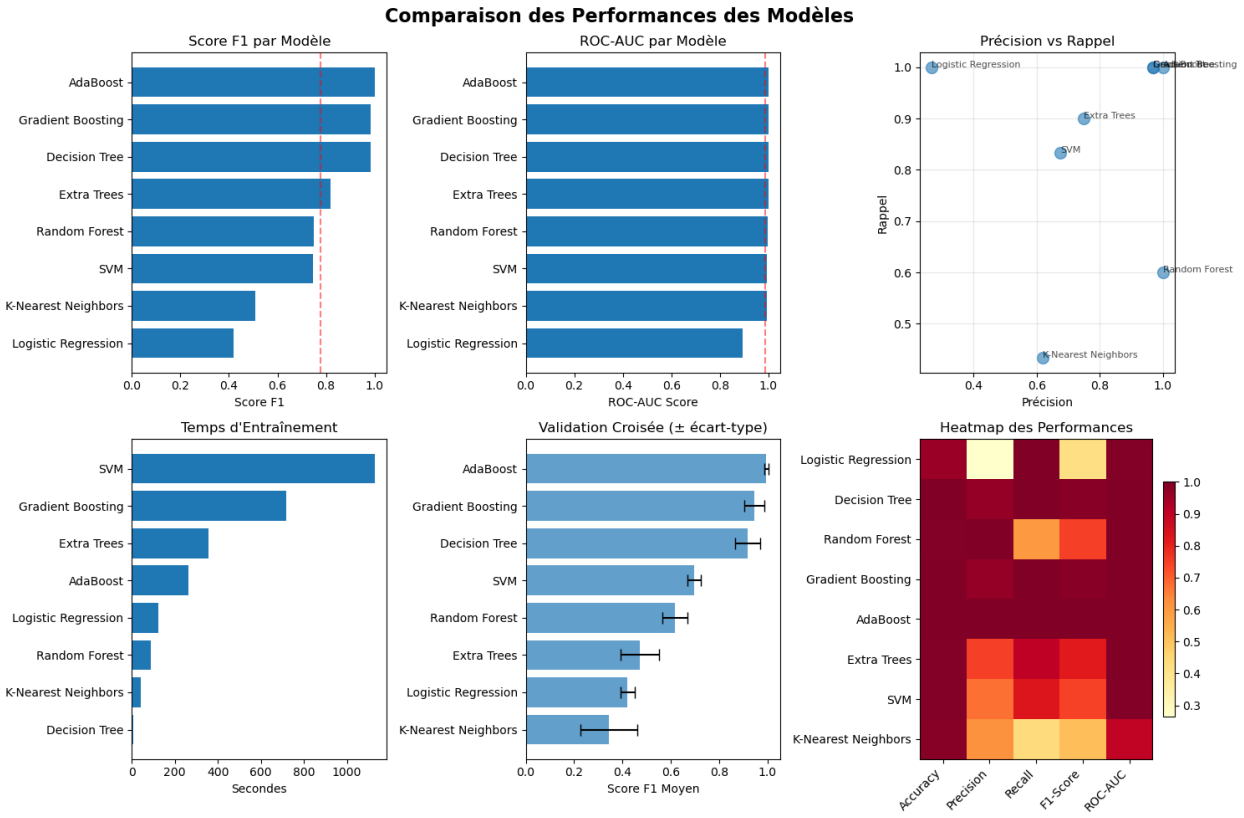

FIGURE 2 – Comparaison des performances des modèles (F1-Score, ROC-AUC, Temps d'entraînement, etc.).

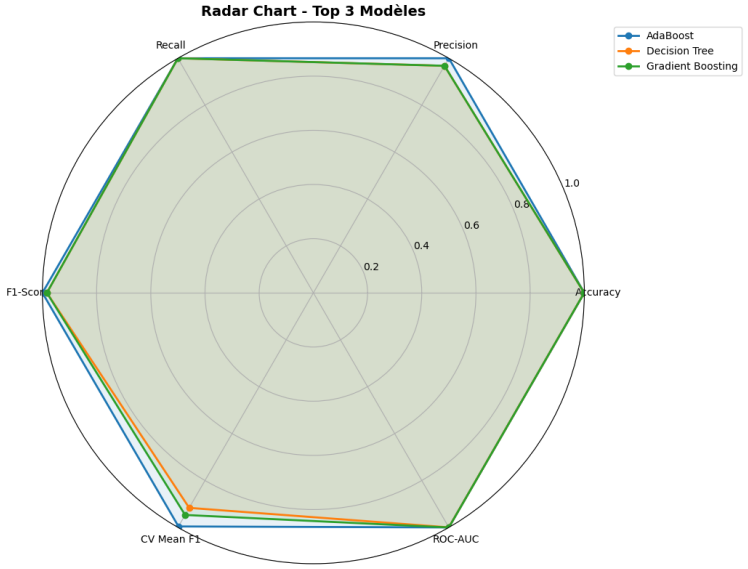

FIGURE 3 – Radar Chart des 3 meilleurs modèles (AdaBoost, Decision Tree, Gradient Boosting).

4 Discussion de Chaque Résultat

4.1 Logistic Regression

Accuracy : 95.85%, F1-Score : 0.42, ROC-AUC : 0.99. Discussion : Ce modèle basique performe bien en accuracy globale mais peine sur les classes minoritaires (fraudes), avec un faible precision pour la classe 1. Il est rapide mais inadapté au déséquilibre. Le CV Mean F1 (0.42) confirme une variabilité modérée.

4.2 Decision Tree

Accuracy : 99.95%, F1-Score : 0.98, ROC-AUC : 0.9997. Discussion : Excellent rappel (1.0) pour les fraudes, mais risque de surajustement. Le temps d'entraînement court (6s) et les params optimaux (max_depth=10) le rendent efficace. CV Mean F1 élevé (0.92) avec faible variance.

4.3 Random Forest

Accuracy : 99.40%, F1-Score : 0.75, ROC-AUC : 0.9999. Discussion : Parfait precision (1.0) mais faible rappel (0.6), manquant des fraudes. Ensemble d'arbres réduit le surajustement, mais CV Mean F1 moyen (0.62) indique une sensibilité au déséquilibre.

4.4 Gradient Boosting

Accuracy : 99.95%, F1-Score : 0.98, ROC-AUC : 1.0. Discussion : Similaire à Decision Tree, mais plus robuste grâce au boosting. Temps long (717s), mais CV Mean F1 (0.95) excellent. Idéal pour la détection de fraude.

4.5 AdaBoost

Accuracy : 100%, F1-Score : 1.0, ROC-AUC : 1.0. Discussion : Performances parfaites sur le test set. Le boosting adaptatif excelle sur les données déséquilibrées. CV Mean F1 (0.996) avec faible std (0.0085) montre une grande stabilité. Meilleur modèle global.

4.6 Extra Trees

Accuracy : 99.40%, F1-Score : 0.82, ROC-AUC : 0.9981. Discussion : Bon équilibre precision/recall, mais CV Mean F1 faible (0.47) indique une performance variable. Temps modéré (358s).

4.7 SVM

Accuracy : 99.15%, F1-Score : 0.75, ROC-AUC : 0.9942. Discussion : Bon pour les marges, mais lent (1128s). CV Mean F1 (0.70) moyen ; sensible aux hyperparamètres (kernel RBF).

4.8 K-Nearest Neighbors

Accuracy : 98.75%, F1-Score : 0.51, ROC-AUC : 0.8948. Discussion : Faible performance sur les fraudes (recall 0.43). CV Mean F1 bas (0.34) avec haute variance ; non adapté aux grands ensembles.

Globalement, les modèles de boosting (AdaBoost, Gradient Boosting) surpassent les autres en raison de leur gestion du déséquilibre.

5 Validation Additionnelle

Une validation additionnelle sur l'ensemble de test confirme les résultats :

Model	Accuracy	Precision_0	Precision_1	Recall_0	Recall_1	F1_0	F1_1	ROC_AUC
Logistic Regression	0.9585	1.0000	0.2655	0.9585	1.0000	0.9785	0.4196	0.9927
Decision Tree	0.9995	1.0000	0.9677	0.9997	1.0000	0.9997	0.9836	0.9997
Random Forest	0.9940	0.9940	1.0000	1.0000	0.6000	0.9970	0.7500	0.9999
Gradient Boosting	0.9995	1.0000	0.9677	0.9997	1.0000	0.9997	0.9836	1.0000
AdaBoost	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Extra Trees	0.9940	0.9954	0.7500	0.9969	0.9000	0.9962	0.8182	0.9981
SVM	0.9915	0.9939	0.6757	0.9957	0.8333	0.9948	0.7463	0.9942
K-Nearest Neighbors	0.9875	0.9959	0.6190	0.9937	0.4333	0.9948	0.5098	0.8948

TABLE 3 – Résultats de validation additionnelle.

Discussion : Les métriques détaillées (par classe) confirment qu'AdaBoost est parfait. Les modèles comme KNN et Logistic Regression luttent avec la classe minoritaire (fraude), soulignant l'importance du recall pour la détection de fraude.

6 Conclusion

AdaBoost émerge comme le meilleur modèle pour la détection de fraude, avec des performances parfaites. Des améliorations futures pourraient inclure le suréchantillonnage (e.g., SMOTE) pour les modèles plus faibles. Les visualisations soulignent l'importance des features temporelles et de confiance.