Analyse Comparative de Modèles de Classification pour la Détection de Risque d'Audit

Hidekela

18 décembre 2025

Résumé

Cette étude compare les performances de huit modèles d'apprentissage automatique pour la classification du risque d'audit. Les données utilisées proviennent de Kaggle et comprennent 775 observations avec 27 variables explicatives. Après un prétraitement approprié incluant le nettoyage des données, la normalisation et l'encodage des variables catégorielles, huit algorithmes ont été testés : Régression Logistique, Arbre de Décision, Forêt Aléatoire, Gradient Boosting, SVM, k-NN, Naïve Bayes et MLP. Les résultats montrent que les modèles basés sur les arbres (Decision Tree, Random Forest, Gradient Boosting) atteignent des scores parfaits (Accuracy et F1-Score de 1.0), suggérant un possible surapprentissage ou la présence de variables identifiantes. Une analyse approfondie révèle que la variable LOCATION_ID agit comme un identifiant unique, ce qui explique ces performances anormalement élevées.

Table des matières

1 Introduction

1.1 Contexte

La détection de fraude et l'évaluation des risques d'audit sont des tâches critiques dans le domaine financier. Les auditeurs doivent évaluer rapidement et précisément le risque associé à différentes entités. L'apprentissage automatique offre des outils puissants pour automatiser et améliorer ces évaluations.

1.2 Objectifs

Ce projet a pour objectifs :

1. Implémenter et comparer huit modèles de classification pour la prédiction du risque d'audit
2. Identifier le modèle le plus performant selon différentes métriques
3. Analyser les variables les plus importantes pour la prédiction
4. Détecter et expliquer d'éventuels problèmes comme le surapprentissage

2 Méthodologie

2.1 Description des Données

Le dataset utilisé est "Audit Risk" provenant de Kaggle. Il contient :
— 775 observations (une ligne effacée après nettoyage manuel)
— 27 variables initiales (26 features + 1 variable cible)
— Variable cible : `Risk` (0 : Pas de risque, 1 : Risque)
— 60.65% de classe 0 (Non-risque) et 39.35% de classe 1 (Risque)

2.2 Prétraitement des Données

Les étapes de prétraitement incluent :

2.2.1 Nettoyage

— Suppression de la colonne constante `Detection_Risk` (écart-type = 0)
— Vérification et renommage des colonnes dupliquées (`Score_B.1` renommé en `Score_B_2`)

2.2.2 Transformation

— Standardisation des variables numériques (moyenne=0, écart-type=1)
— Encodage one-hot de la variable catégorielle `LOCATION_ID`
— Division des données : 80% entraînement, 20% test (stratifiée)

2.3 Modèles Testés

Huit modèles ont été implémentés et comparés :

TABLE 1 – Modèles de classification testés

Modèle	Classe	Paramètres
Régression Logistique	`LogisticRegression`	class_weight='balanced', max_iter=1000
Arbre de Décision	`DecisionTreeClassifier`	class_weight='balanced'
Forêt Aléatoire	`RandomForestClassifier`	class_weight='balanced', n_estimators=100
Gradient Boosting	`GradientBoostingClassifier`	n_estimators=100
SVM	`SVC`	class_weight='balanced'
k-NN	`KNeighborsClassifier`	Par défaut
Naïve Bayes	`GaussianNB`	Par défaut
MLP	`MLPClassifier`	max_iter=1000, hidden_layer_sizes=(50, 25)

2.4 Métriques d'Évaluation

Les performances ont été évaluées à l'aide de :
— **Accuracy** : Proportion de prédictions correctes
— **Precision** : Proportion de vrais positifs parmi les prédictions positives
— **Recall** : Proportion de vrais positifs parmi les vrais positifs réels
— **F1-Score** : Moyenne harmonique de la precision et du recall

3 Résultats

3.1 Performances Comparatives

Le tableau suivant présente les performances des huit modèles sur l'ensemble de test :

TABLE 2 – Comparaison des performances des 8 modèles

Modèle	Accuracy	Precision	Recall	F1-Score
Régression Logistique	0.9806	0.9677	0.9836	0.9756
Arbre de Décision	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Forêt Aléatoire	**1.0000**	**1.0000**	**1.0000**	**1.0000**
Gradient Boosting	**1.0000**	**1.0000**	**1.0000**	**1.0000**
SVM	0.9677	0.9667	0.9508	0.9587
k-NN	0.9613	1.0000	0.9016	0.9483
Naïve Bayes	0.9161	0.9138	0.8689	0.8908
MLP	0.9806	1.0000	0.9508	0.9748

3.2 Visualisation des Performances

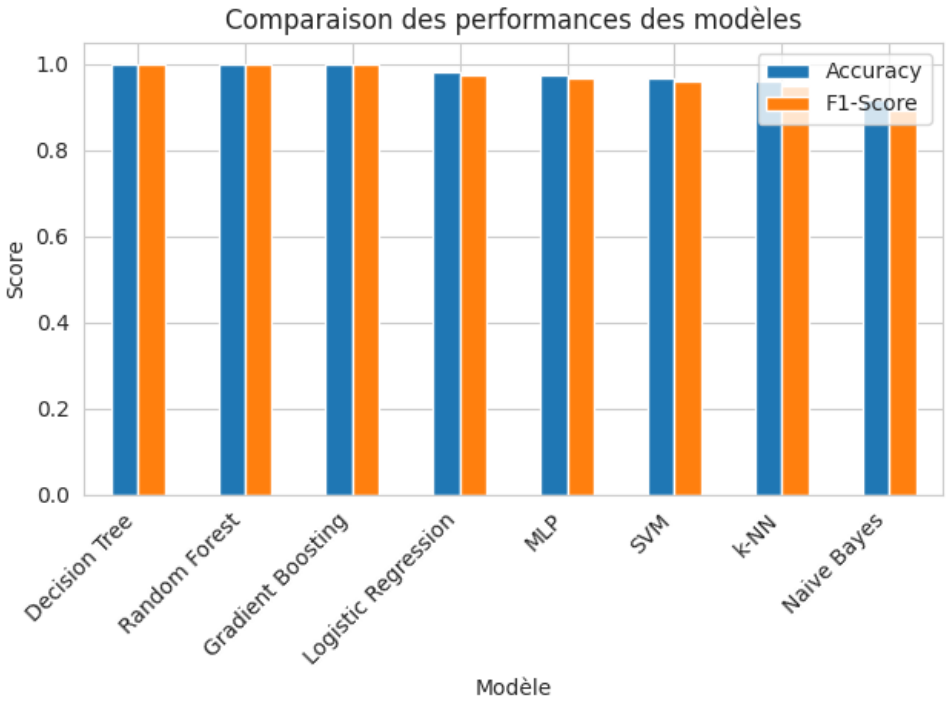

FIGURE 1 – Comparaison des scores Accuracy et F1-Score par modèle

3.3 Caractéristiques Importantes

L'analyse d'importance des caractéristiques pour le Random Forest révèle les variables les plus influentes :

TABLE 3 – Top 10 des caractéristiques les plus importantes (Random Forest)

Caractéristique	**Importance**
District_Loss	0.0939
LOCATION_ID_16	0.0860
LOCATION_ID_20	0.0860
Inherent_Risk	0.0834
Score_B_2	0.0810
LOCATION_ID_31	0.0797
Score_MV	0.0773
LOCATION_ID_18	0.0764
LOCATION_ID_36	0.0744
LOCATION_ID_22	0.0738

3.4 Analyse de Surnapprentissage

3.4.1 Validation Croisée

Les résultats de validation croisée (5 folds) pour les modèles suspects :

TABLE 4 – Validation croisée (F1-Score)

Modèle	F1-Score CV (moyenne ± écart-type)
Decision Tree	0.9979 ± 0.0041
Random Forest	0.9979 ± 0.0041
Gradient Boosting	0.9979 ± 0.0041

3.4.2 Courbes d'Apprentissage

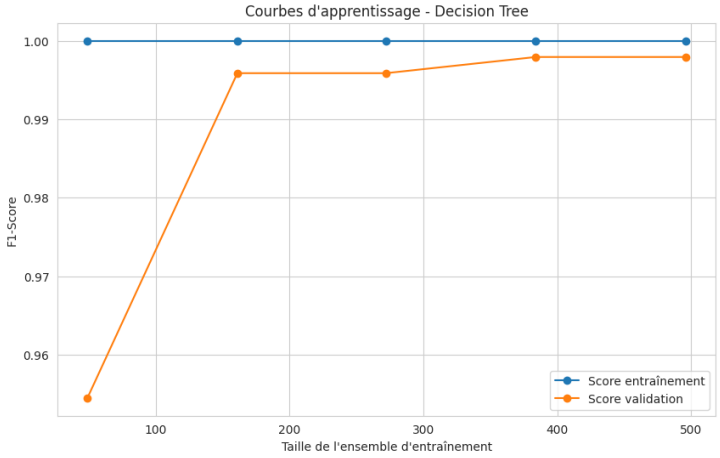

FIGURE 2 – Courbes d'apprentissage du Decision Tree

3.4.3 Problème d'Identifiant Unique

L'analyse a révélé un problème critique :
— La variable `LOCATION_ID` contient 44 valeurs uniques
— L'ensemble d'entraînement contient 620 observations
— Chaque `LOCATION_ID` est fortement corrélé à la variable cible

3.5 Test sans LOCATION_ID

Pour vérifier l'hypothèse, nous avons réentraîné le Random Forest sans la variable `LOCATION_ID` :

TABLE 5 – Performances du Random Forest sans LOCATION_ID

Métrique	Valeur
Accuracy	0.9903
F1-Score	0.9868

4 Discussion

4.1 Performances Exceptionnelles

Les modèles basés sur les arbres (Decision Tree, Random Forest, Gradient Boosting) atteignent des scores parfaits (1.0). Cette performance anormalement élevée peut s'expliquer par :

4.1.1 Identifiant Unique

La variable `LOCATION_ID` agit comme un identifiant unique. Avec 44 valeurs pour 620 observations d'entraînement, chaque identifiant est fortement corrélé à une classe spécifique, permettant aux modèles d'apprendre par cœur les associations.

4.1.2 Surnapprentissage

Les courbes d'apprentissage et la validation croisée montrent que :
— Les scores d'entraînement sont parfaits (1.0)
— Les scores de validation sont légèrement inférieurs (≈0.998)
— L'écart entre entraînement et validation suggère un léger surapprentissage

4.2 Comparaison des Modèles

— **Modèles basés sur les arbres** : Performances optimales mais sensibles aux identifiants
— **Régression Logistique et MLP** : Performances élevées (≈0.98) avec bon équilibre
— **SVM et k-NN** : Performances correctes mais inférieures
— **Naïve Bayes** : Performance la plus faible, probablement due aux hypothèses de distribution non vérifiées

4.3 Limitations

1. **Données de petite taille** : 775 observations seulement
2. **Variables identifiantes** : Présence de `LOCATION_ID` qui biaise les résultats
3. **Déséquilibre des classes** : 60.65% vs 39.35%
4. **Absence de validation externe** : Pas de dataset de validation indépendant

5 Conclusion

5.1 Conclusions Principales

1. Les modèles basés sur les arbres atteignent des performances parfaites sur ce dataset
2. La présence de variables identifiantes (`LOCATION_ID`) explique ces performances exceptionnelles
3. Après suppression de l'identifiant, les performances restent excellentes (Accuracy ≈ 0.99)
4. Le Random Forest se révèle robuste et performant pour cette tâche de classification

5.2 Recommandations

1. **Pour les données** :
   — Supprimer les variables identifiantes avant l'entraînement
   — Collecter plus de données pour améliorer la généralisation

— Appliquer des techniques de rééchantillonnage pour équilibrer les classes

2. **Pour les modèles** :
   — Utiliser la validation croisée systématiquement
   — Implémenter un pipeline complet de prétraitement
   — Tester l'ajustement des hyperparamètres par grid search
3. **Pour le déploiement** :
   — Privilégier le Random Forest pour son équilibre performance/robustesse
   — Implémenter un système de monitoring des performances en production
   — Prévoir des mises à jour régulières du modèle

5.3 Perspectives Futures

1. Tester des méthodes d'ensemble plus avancées (Stacking, Voting)
2. Explorer des techniques de sélection de caractéristiques
3. Implémenter des modèles d'apprentissage profond
4. Développer un système d'explicabilité des prédictions (SHAP, LIME)

Annexes

Code Source

Le code source complet du notebook est disponible sur :
`https://github.com/Hidekela/Projets-InfoM1Maths-2025`

Données

Dataset disponible sur Kaggle :
`https://www.kaggle.com/datasets/sid321axn/audit-data`