Détection de Fraude Bancaire

Guide Complet : de l'Exploration des Données à la Modélisation

Fiche Projet – Data Science

2026

1. Introduction et Contexte

◎ Objectif du Projet

Ce projet vise à développer un système de détection automatique de fraudes bancaires en utilisant des techniques de Machine Learning. L'objectif est de :

— Analyser les caractéristiques des transactions frauduleuses,

— Construire un modèle prédictif performant,

— Évaluer rigoureusement sa capacité de détection.

La fraude bancaire représente un enjeu financier majeur pour les institutions financières. Détecter ces transactions frauduleuses de manière automatisée permet de :

— Réduire les pertes financières,

— Protéger les clients,

— Améliorer la confiance dans le système bancaire.

2. Phase 1 : Exploration des Données (EDA)

L'analyse exploratoire des données est une étape fondamentale qui précède toute modélisation.

2.1 Chargement et Inspection Initiale

```python
import pandas as pd
import numpy as np

# Chargement du dataset
data_credit = pd.read_csv('creditcard_2023.csv')

# Apercu des donnees
print(data_credit.head())
print(data_credit.info())
print(data_credit.describe())
```

Points Clés

Cette première inspection permet d'identifier :

— **La variable cible** : `Class` (0 = Normal, 1 = Fraude)

— **Les features** : Variables anonymisées $V_1, V_2, \ldots, V_{28}$ et `Amount`

— **Les types de données** et la présence éventuelle de valeurs manquantes

2.2 Analyse de la Distribution des Classes

```python
# Verification de l'equilibre des classes
counts = data_credit['Class'].value_counts()
print(f"Transactions normales (0) : {counts[0]}")
print(f"Transactions frauduleuses (1) : {counts[1]}")

# Visualisation
import matplotlib.pyplot as plt
data_credit['Class'].value_counts().plot(kind='bar')
plt.title('Distribution des Classes')
plt.xlabel('Class')
plt.ylabel('Nombre de transactions')
plt.show()
```

Attention - Contexte Pédagogique

Particularité de ce dataset pédagogique :
Les classes sont équilibrées (50% normales / 50% frauduleuses). Cette configuration simplifie l'apprentissage mais ne reflète **pas la réalité**.

2.3 Analyse des Corrélations

La corrélation de Pearson mesure la relation linéaire entre chaque variable et la fraude :

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{1}$$

```python
# Calcul des correlations avec la cible
correlation = data_credit.corr()['Class'].sort_values(ascending
    =False)
print(correlation)

# Visualisation heatmap (optionnel)
import seaborn as sb
plt.figure(figsize=(12, 8))
sb.heatmap(data_credit.corr(), cmap='coolwarm', center=0)
plt.title('Matrice de Correlation')
plt.show()
```

Points Clés

Les variables avec $|\rho|$ élevé (ex : V14, V4, V10) sont des candidats prioritaires pour la modélisation car elles contiennent une forte information prédictive.

2.4 Visualisation des Variables Clés

```python
# Boxplots pour comparer les distributions
import seaborn as sb

fig, axes = plt.subplots(2, 2, figsize=(14, 10))

sb.boxplot(x='Class', y='V14', data=data_credit, ax=axes[0,0])
sb.boxplot(x='Class', y='V4', data=data_credit, ax=axes[0,1])
sb.boxplot(x='Class', y='V10', data=data_credit, ax=axes[1,0])
sb.boxplot(x='Class', y='Amount', data=data_credit, ax=axes
    [1,1])

plt.tight_layout()
plt.show()
```

À Retenir

Une bonne séparation visuelle entre les deux classes indique un fort pouvoir discriminant de la variable.

3. Phase 2 : Prétraitement des Données

Le prétraitement transforme les données brutes en format exploitable par les algorithmes de ML.

3.1 Les Quatre Piliers du Preprocessing

1. **Sélection des Features (Feature Selection)**
   Conservation uniquement des variables à fort pouvoir prédictif pour réduire le bruit.
2. **Mise à l'Échelle (Scaling)**
   Standardisation des variables pour éviter qu'une variable à grande échelle domine le modèle.
3. **Nettoyage**
   Vérification et traitement des valeurs manquantes et des doublons.
4. **Séparation Train/Test**
   Division du dataset pour entraîner et évaluer objectivement le modèle.

3.2 Standardisation (Z-score)

La standardisation ramène chaque variable à une distribution centrée réduite :

$$\boxed{z = \frac{x - \mu}{\sigma}} \tag{2}$$

où x = valeur d'origine, μ = moyenne, σ = écart-type.

3.3 Implémentation Complète

```python
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# 1. Selection des features importantes
important_features = ['V14', 'V4', 'V10', 'V12', 'V17',
                      'V2', 'V3', 'V9', 'V7', 'Amount']
X = data_credit[important_features].copy()
y = data_credit['Class']

# 2. Verification des donnees manquantes
print(f"Valeurs manquantes : {X.isnull().sum().sum()}")

# 3. Standardisation de la variable Amount
scaler = StandardScaler()
X['Amount'] = scaler.fit_transform(X[['Amount']])

# 4. Separation Train/Test (80/20)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

print(f"Taille train : {X_train.shape}")
print(f"Taille test : {X_test.shape}")
```

À Retenir

Pourquoi `stratify=y` ?
Cette option garantit que la proportion de fraudes est identique dans les ensembles d'entraînement et de test (ici 50/50), assurant une évaluation fiable.

4. Phase 3 : Modélisation par Régression Logistique

4.1 Principe de la Régression Logistique

La régression logistique modélise la probabilité qu'une transaction soit frauduleuse via la fonction sigmoïde :

$$P(Y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}} \tag{3}$$

Le modèle estime les coefficients β qui maximisent la vraisemblance des données observées.

4.2 Pourquoi la Régression Logistique ?

— **Interprétabilité** : Les coefficients indiquent l'influence de chaque variable

— **Probabilités** : Permet d'ajuster le seuil de décision selon les coûts métier

— **Efficacité** : Rapide à entraîner, même sur de gros volumes

— **Baseline** : Modèle de référence en finance

4.3 Entraînement du Modèle

```python
from sklearn.linear_model import LogisticRegression

# Initialisation et entrainement
model = LogisticRegression(
    solver='lbfgs',
    max_iter=1000,
    random_state=42
)

model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]
```

5. Phase 4 : Évaluation des Performances

5.1 Matrice de Confusion

	Prédit Normal	**Prédit Fraude**
Réel Normal	TN (Vrai Négatif)	FP (Faux Positif)
Réel Fraude	FN (Faux Négatif)	TP (Vrai Positif)

```
from sklearn.metrics import confusion_matrix,
    classification_report
import seaborn as sb

# Matrice de confusion
cm = confusion_matrix(y_test, y_pred)
sb.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Matrice de Confusion')
plt.ylabel('Classe Reelle')
plt.xlabel('Classe Predite')
plt.show()
```

5.2 Métriques de Performance

Points Clés

Les 4 Métriques Essentielles :

— **Accuracy** : $\frac{TP+TN}{TP+TN+FP+FN}$ – Taux de bonnes prédictions global

— **Precision** : $\frac{TP}{TP+FP}$ – Fiabilité des alertes fraude

— **Recall (Sensibilité)** : $\frac{TP}{TP+FN}$ – Capacité à détecter les fraudes réelles

— **F1-Score** : $2 \times \frac{Precision \times Recall}{Precision+Recall}$ – Compromis optimal

```
from sklearn.metrics import accuracy_score, precision_score,
    recall_score, f1_score

print(f"Accuracy  : {accuracy_score(y_test, y_pred):.4f}")
print(f"Precision : {precision_score(y_test, y_pred):.4f}")
print(f"Recall    : {recall_score(y_test, y_pred):.4f}")
print(f"F1-Score  : {f1_score(y_test, y_pred):.4f}")

# Rapport detaille
print("\n", classification_report(y_test, y_pred))
```

5.3 Courbe ROC et AUC

```python
from sklearn.metrics import roc_curve, roc_auc_score

# Calcul de l'AUC
auc = roc_auc_score(y_test, y_proba)
print(f"AUC-ROC : {auc:.4f}")

# Trace de la courbe ROC
fpr, tpr, thresholds = roc_curve(y_test, y_proba)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc:.4f})')
plt.plot([0, 1], [0, 1], 'k--', label='Random Classifier')
plt.xlabel('Taux de Faux Positifs')
plt.ylabel('Taux de Vrais Positifs')
plt.title('Courbe ROC')
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

$$0.5 \leq AUC \leq 1 \tag{4}$$

Plus l'AUC est proche de 1, meilleure est la capacité discriminante du modèle.

6. Résultats du Projet

6.1 Performances Obtenues

Métrique	Valeur
Accuracy	0.9580
Precision	0.9791
Recall	0.9360
F1-score	0.9571
AUC-ROC	0.9907

6.2 Validation Croisée

```python
from sklearn.model_selection import cross_val_score

# Validation croisee 5-fold
cv_scores = cross_val_score(model, X, y, cv=5, scoring='f1')
print(f"F1-Score moyen : {cv_scores.mean():.4f} (+/- {cv_scores
    .std():.4f})")
```

Métrique	Moyenne $\pm$ Écart-type
F1-score (CV)	0.9573 ± 0.0010

Points Clés

Interprétation des Résultats :

— Recall de 93,6% : le modèle détecte correctement plus de 9 fraudes sur 10

— Precision de 97,9% : très peu de fausses alertes

— AUC de 0.99 : excellente capacité de discrimination

— Faible variance : modèle stable et robuste

7. Limites et Perspectives

7.1 Contexte Pédagogique vs Réalité

Attention - Contexte Pédagogique

Différences Majeures avec la Réalité Bancaire :

1. **Déséquilibre extrême** : En pratique, les fraudes représentent moins de 0,1% à 1% des transactions (et non 50%). Cela nécessite :
   — Des techniques de rééquilibrage (SMOTE, sous-échantillonnage)
   — Des métriques adaptées (Precision-Recall plutôt qu'Accuracy)
   — Un seuil de décision ajusté selon les coûts métier
2. **Coûts asymétriques** : Une fraude non détectée (FN) coûte bien plus cher qu'une fausse alerte (FP). Le modèle doit privilégier le Recall.
3. **Évolution temporelle** : Les fraudeurs adaptent leurs techniques. Le modèle nécessite un réentraînement régulier.

7.2 Méthodes Plus Avancées

En pratique, les institutions financières utilisent des approches plus sophistiquées :

— **Random Forest** : Capture les interactions non-linéaires entre variables

— **XGBoost / LightGBM** : Performance supérieure grâce au boosting

— **Réseaux de Neurones** : Deep Learning pour patterns complexes

— **Isolation Forest** : Détection d'anomalies non supervisée

— **Approches hybrides** : Combinaison de plusieurs modèles (ensemble)

7.3 Améliorations Possibles

1. **Feature Engineering** : Créer de nouvelles variables (fréquence, montants moyens, géolocalisation)
2. **Ajustement du seuil** : Optimiser le seuil de décision selon les coûts métier

3. **Analyse temporelle** : Intégrer l'historique des transactions
4. **Explainability** : Utiliser SHAP ou LIME pour interpréter les prédictions

8. Conclusion

Synthèse du Projet

Ce projet a permis de :

— Maîtriser le pipeline complet d'un projet ML : EDA → Preprocessing → Modélisation → Évaluation

— Construire un modèle de régression logistique performant (AUC = 0.99)

— Comprendre les métriques adaptées à la détection de fraude

— Identifier les différences entre contexte pédagogique et réalité opérationnelle

À Retenir

Points à retenir :
La régression logistique constitue une excellente **baseline** pour la détection de fraude grâce à son interprétabilité et sa rapidité. Toutefois, dans un contexte réel avec déséquilibre extrême (99% vs 1%), des modèles plus sophistiqués (XGBoost, Neural Networks) et des techniques de rééquilibrage sont nécessaires pour maximiser le Recall tout en contrôlant les faux positifs.