



# Détectez des faux billets avec Python

Rémi Bardey - Décembre 2025



# Contexte & Mission

## Contexte de la mission

L'ONCFM veut accélérer l'identification des faux billets.

**Objectif opérationnel** : aider les équipes terrain à décider rapidement après scan des billets (mesures physiques : longueur, hauteurs, marges, diagonale, etc.).

**Contrainte** : l'ONCFM ne dispose pas des compétences internes pour construire un modèle de machine learning.

**Ressources fournies** : un cahier des charges + un jeu de données de 1 500 billets scannés (1 000 vrais, 500 faux).

**Attente méthodologique** : l'agence EMV recommande de tester 4 algorithmes (K-means, Régression logistique, KNN, Random Forest).

## Objectifs

- Réaliser le pipeline de préparation des données et l'analyse exploratoire dans un notebook.
- Tester et comparer les algorithmes recommandés (et éventuellement d'autres), puis évaluer leurs performances.
- Sélectionner un modèle final justifié (performance, robustesse, interprétabilité, usage métier).
- Livrer une application de prédiction utilisable par les équipes (script séparé du notebook).
- Produire un support de présentation synthétique : traitements amont, résultats des modèles, choix final, démonstration de l'application.

# Dataset initial

is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
True	171.81	104.86	104.95	4.52	2.89	112.83
True	171.46	103.36	103.66	3.77	2.99	113.09
True	172.69	104.48	103.50	4.40	2.94	113.16
True	171.36	103.91	103.94	3.62	3.01	113.51
True	171.73	104.28	103.46	4.04	3.48	112.54

Column	Non-Null Count	Dtype
is_genuine	1500 non-null	bool
diagonal	1500 non-null	float64
height_left	1500 non-null	float64
height_right	1500 non-null	float64
margin_low	1463 non-null	float64
margin_up	1500 non-null	float64
length	1500 non-null	float64

## Jeu de données

1 500 observations (billets scannés).

7 colonnes : 1 cible + 6 variables explicatives.

## Variables disponibles (types)

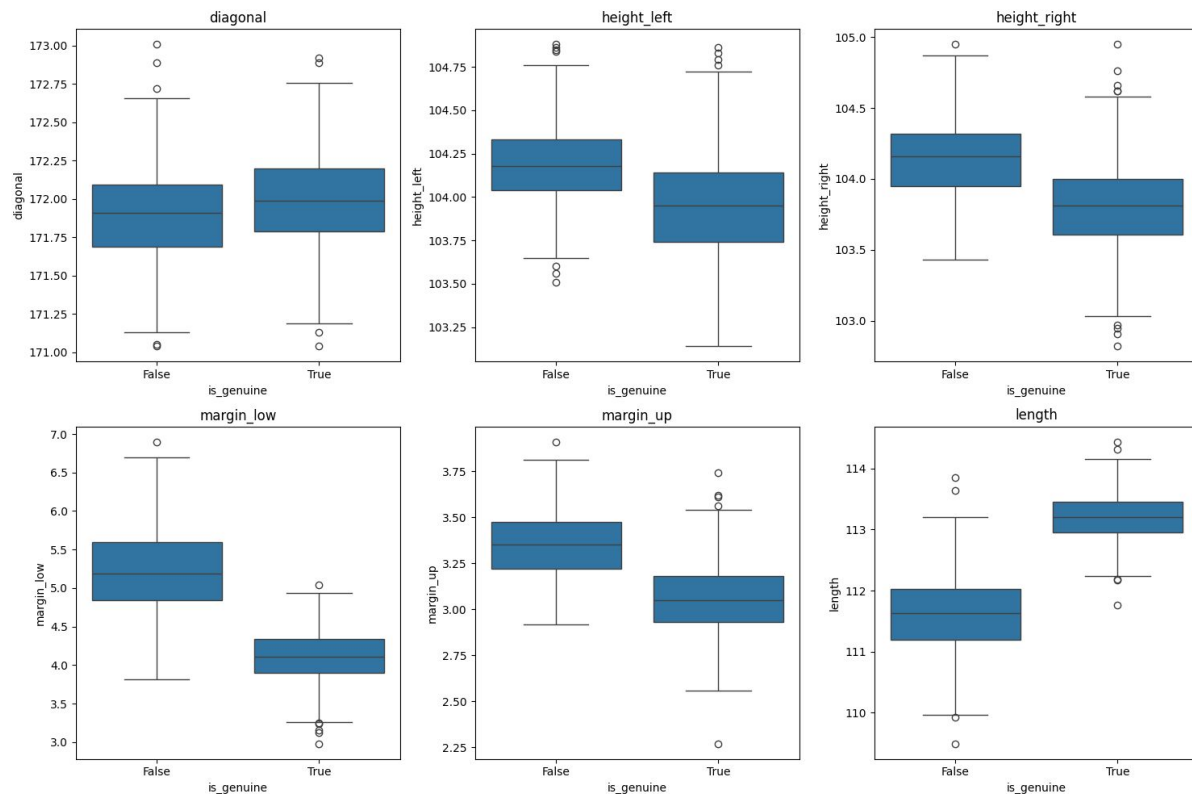
is\_genuine (bool) : indique si le billet est vrai/faux.

Variables numériques (float64) : diagonal, height\_left, height\_right, margin\_low, margin\_up, length.

## Valeurs manquantes

Toutes les colonnes sont complètes (1 500 non-null) sauf : margin\_low : 1 463 non-null soit 37 valeurs manquantes.

# EDA — variables discriminantes



- Variables les plus discriminantes : margin\_low, length (séparation nette vrai/faux).
- Séparation modérée : margin\_up, height\_left, height\_right.
- Peu discriminante : diagonal (forte superposition).
- Présence d'outliers → prise en compte au prétraitement (scaling) et choix du modèle.

# Corrélations — base de l'imputation

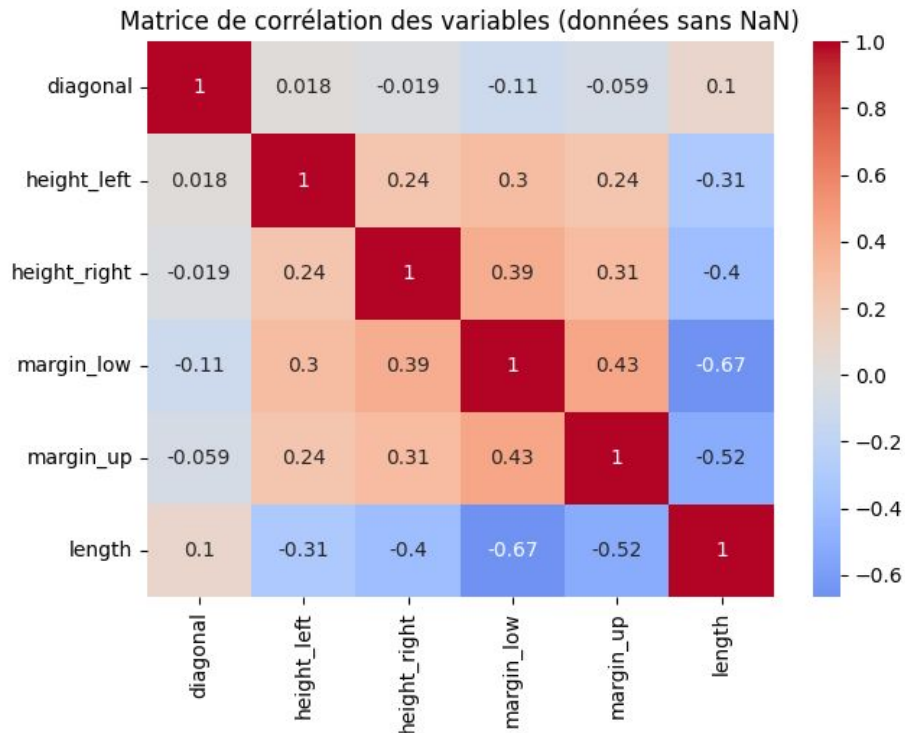
La heatmap montre les relations entre variables avant imputation/modélisation.

Signal fort autour de length : corrélations négatives avec margin\_low (-0,67) et margin\_up (-0,52) → variables liées et utiles pour prédire.

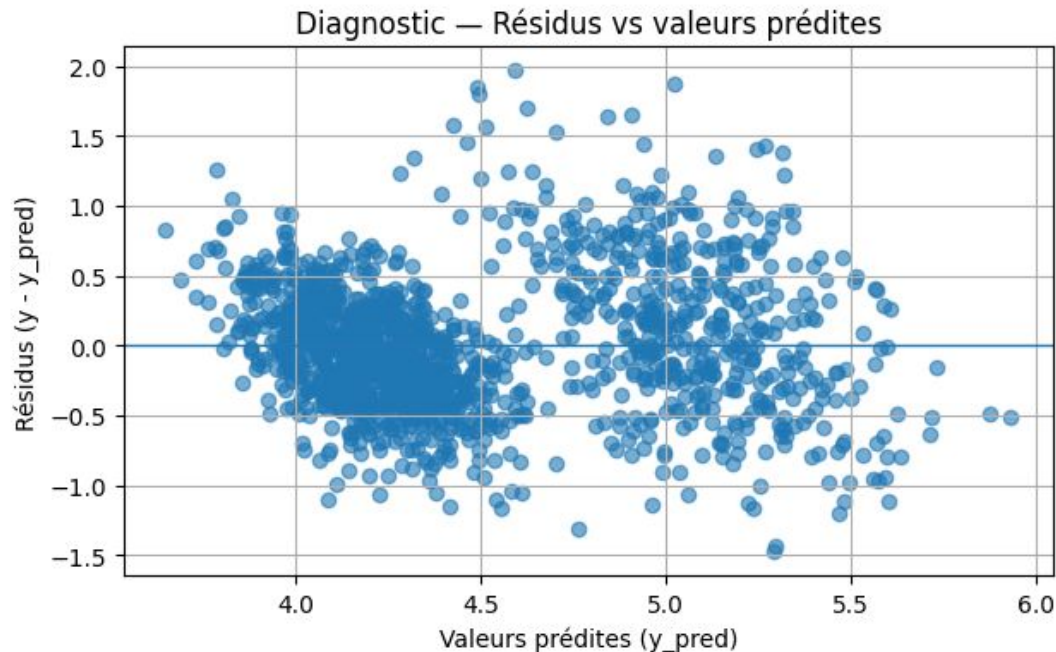
Signal modéré : height\_left et height\_right sont corrélées (~0,24), cohérent pour deux mesures proches.

diagonal reste faiblement corrélée ( $\approx 0$ ) → information plutôt indépendante, complémentaire.

Pas de corrélations extrêmes ; la multicolinéarité sera confirmée par le VIF

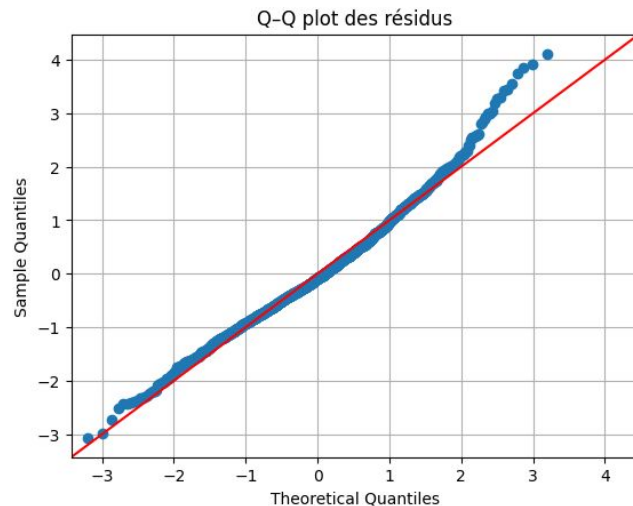
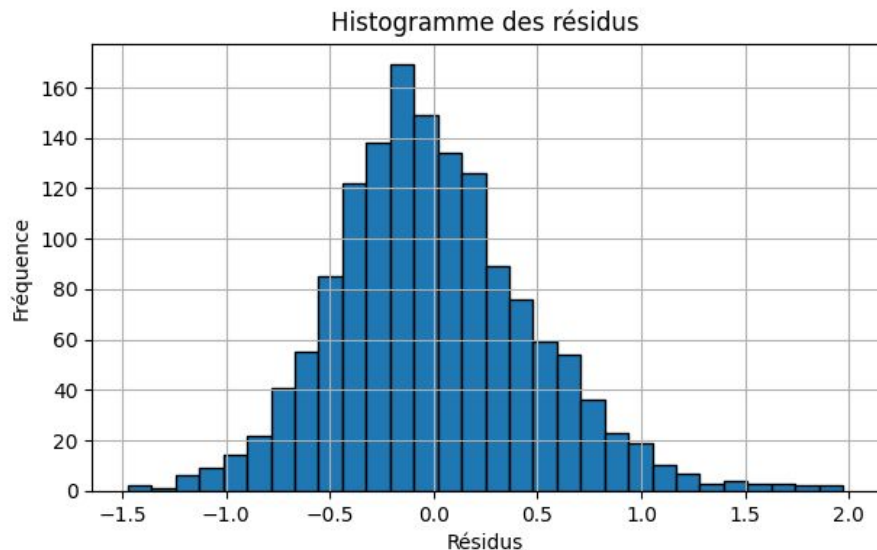


# Imputation — résidus vs valeurs ajustées



- Résidus centrés sur 0 → pas de biais global.
- Hétéroscédasticité visible (dispersion non uniforme), confirmée par Breusch–Pagan ( $p < 0,05$ ).
- Acceptable ici : l'imputation vise une valeur plausible ; la performance finale est validée par les modèles de classification.

# Imputation — normalité des résidus



- Histogramme : distribution proche d'une cloche, centrée  $\approx 0$ .
- Q-Q plot : alignement au centre, écarts en extrémités  $\rightarrow$  queues plus lourdes / outliers.
- Shapiro significatif : test sensible sur grands échantillons  $\rightarrow$  décision basée surtout sur les graphes.

# Imputation — VIF et dataset complété

Variable	VIF
length	1.576950
margin_up	1.404404
height_right	1.230115
height_left	1.138261
diagonal	1.013613



is_genuine	0
diagonal	0
height_left	0
height_right	0
margin_low	0
margin_up	0
length	0

- VIF faibles ( $\approx 1.0$  à  $1.6$ ) → pas de multicolinéarité problématique.
- Modèle linéaire stable pour l'imputation.
- On peut imputer margin\_low sans risque majeur de coefficients instables.

- Prédiction des valeurs manquantes avec `lin_reg.predict(df_missing[features])`.
- Remplacement dans le dataset : margin\_low complété.
- Données prêtes pour la suite : standardisation + entraînement des modèles.

## Préparation — définir X et y

```
# X = variables explicatives - Y = cible
X = df_billet[['diagonal',
               'height_left',
               'height_right',
               'margin_low',
               'margin_up',
               'length']]
y = df_billet["is_genuine"]
```

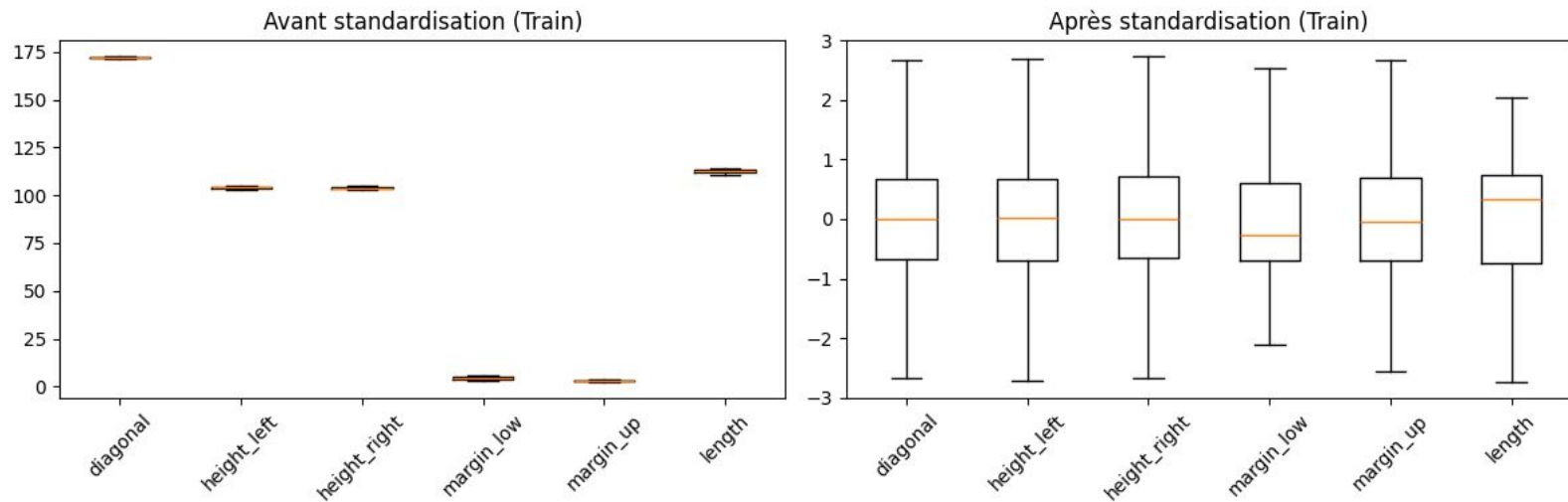
- X = mesures physiques du billet : diagonal, height\_left, height\_right, margin\_low, margin\_up, length.
- y = étiquette à prédire : is\_genuine (vrai/faux).
- Séparer X et y permet d'entraîner un modèle qui apprend la relation entre caractéristiques mesurées et authenticité.

# Préparation — split train/test

	n	n_true	n_false	% true	% false
<b>Train</b>	1200.000000	800.000000	400.000000	66.7%	33.3%
<b>Test</b>	300.000000	200.000000	100.000000	66.7%	33.3%

- Split : 1 200 billets pour l'entraînement, 300 billets pour le test (6 variables).
- La stratification conserve exactement la même proportion de classes :
- Train : 66,7% vrais / 33,3% faux
- Test : 66,7% vrais / 33,3% faux
- Intérêt : évaluation plus fiable (pas de biais dû à un déséquilibre différent entre train et test).

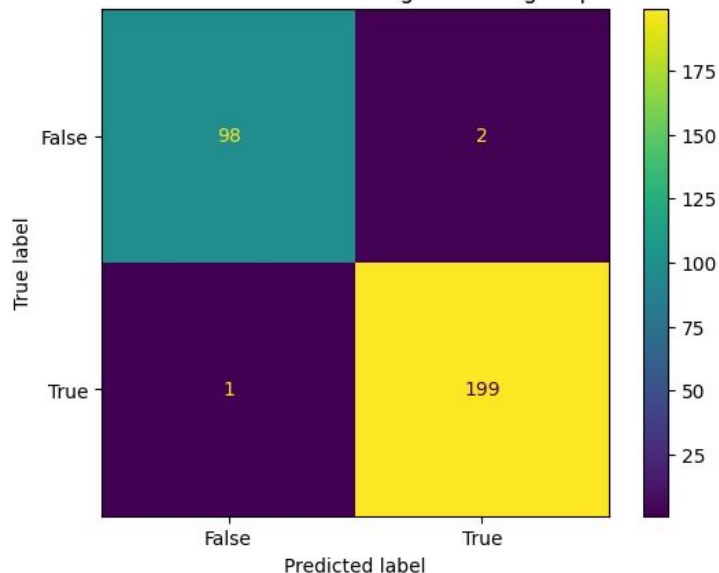
# Préparation — standardisation



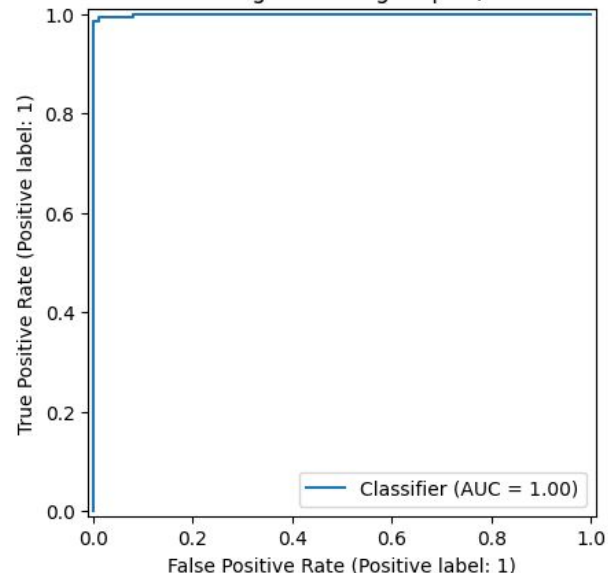
- Les variables n'ont pas la même unité / amplitude (ex. diagonal  $\sim 170$  vs marges  $\sim 3-4$ )  $\rightarrow$  risque que certaines dominent les algos basés sur distances/gradients.
- StandardScaler centre et réduit : moyenne  $\approx 0$ , écart-type  $\approx 1$  (calculé sur le train).
- Bonne pratique : fit sur train, puis transform sur test (évite la fuite de données).
- Impact attendu : amélioration de la performance/stabilité pour KNN, Régression logistique, K-means (moins critique pour Random Forest).

# Modèle 1 — Régression logistique

Matrice de confusion — Régression logistique



Courbe ROC — Régression logistique (AUC = 1.000)

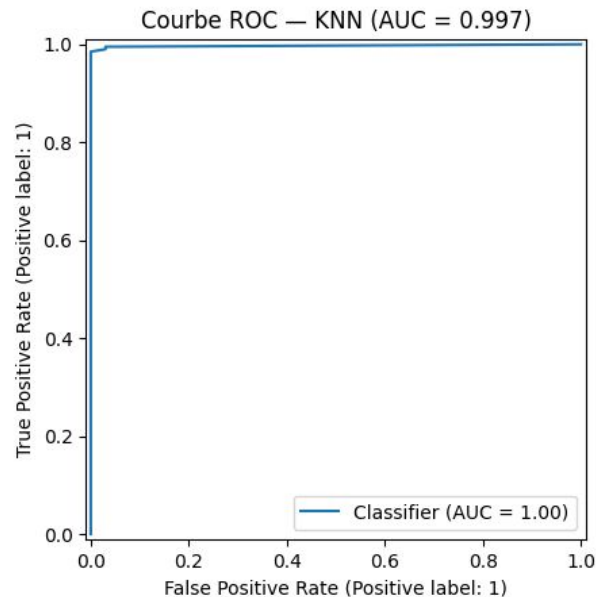
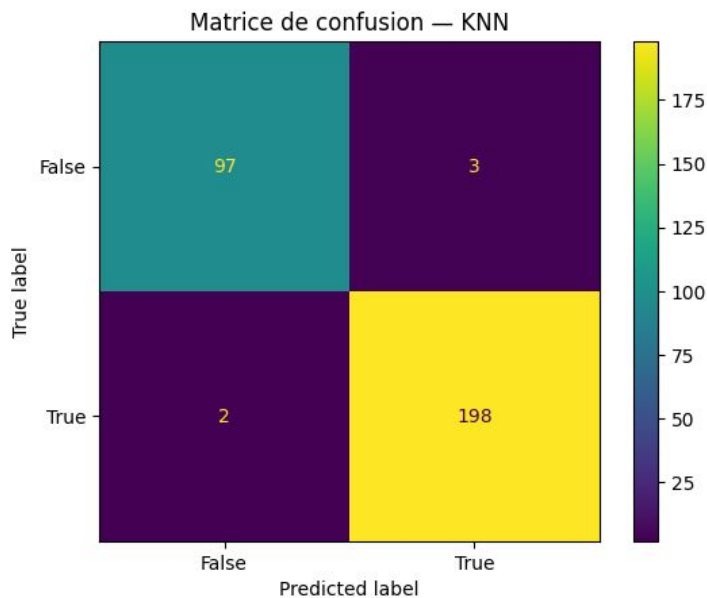


- 297 / 300 billets bien classés → Accuracy = 99,0%

Erreurs :

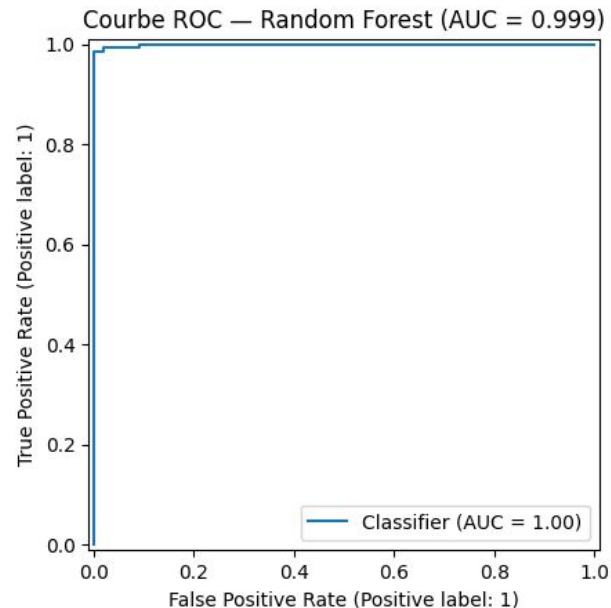
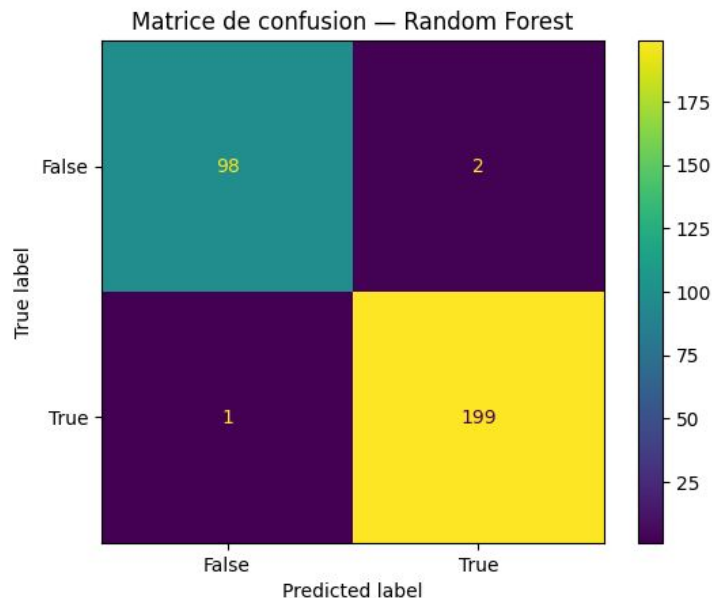
- 2 faux billets prédits vrais (risque principal métier : un faux passe)
- 1 vrai billet prédit faux (impact opérationnel : rejet d'un vrai)
- AUC = 1.000

## Modèle 2 — KNN



- 295 / 300 billets bien classés → Accuracy = 98,3%
- Erreurs :
- 3 faux billets (False) prédits vrais (True) → faux qui passent (3 / 100 = 3% des faux du test)
  - 2 vrais billets (True) prédits faux (False) → vrais rejetés (2 / 200 = 1% des vrais du test)
  - AUC = 0.997

# Modèle 3 — Random Forest

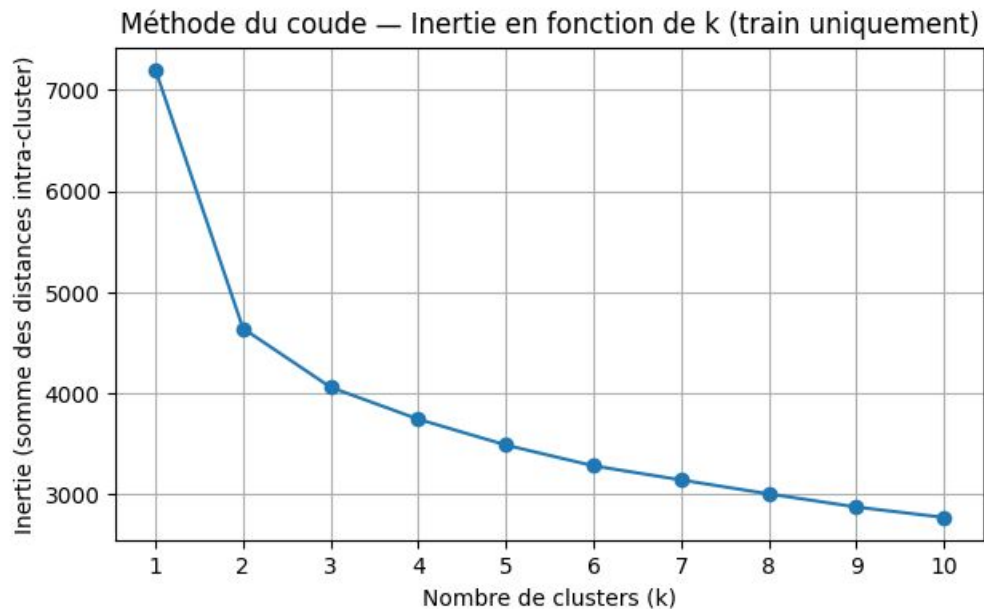


- 297 / 300 billets bien classés → Accuracy = 99,0%

Erreurs :

- 2 faux billets (False) prédits vrais (True) → faux qui passent
- 1 vrai billet (True) prédit faux (False) → vrai rejeté
- AUC = 0,999

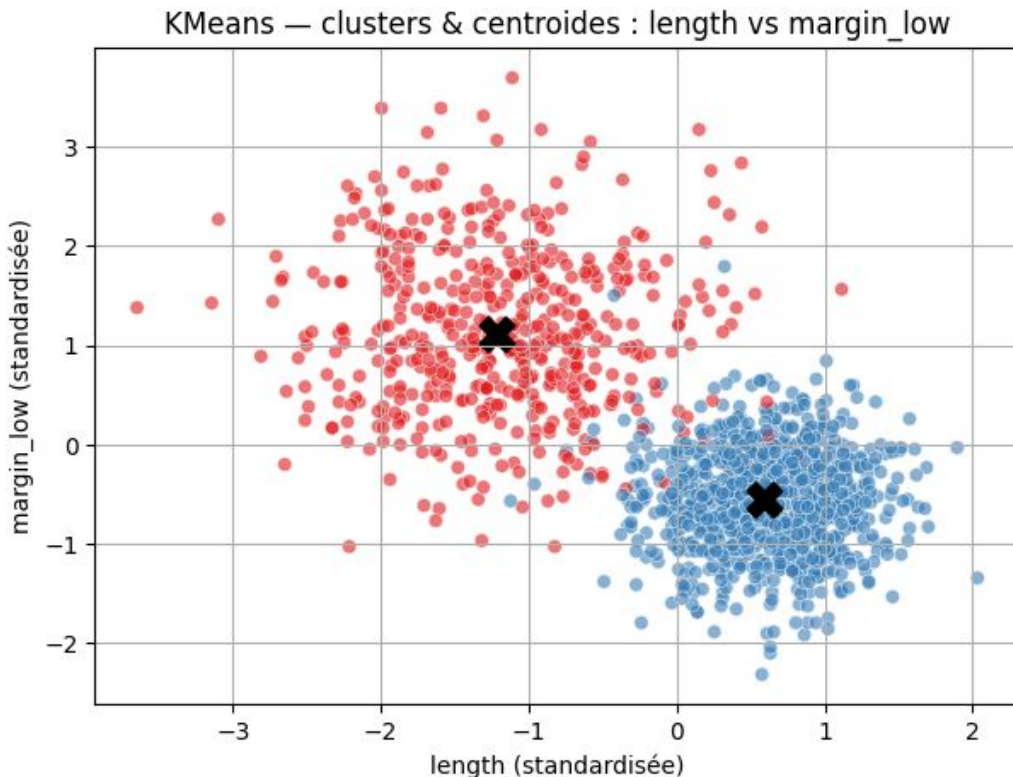
# K-means — choix de k (coude)



- K-means est non supervisé : regroupe les billets par similarité, sans utiliser `is_genuine`.
- Inertie calculée sur `X_train_scaled` uniquement (pas de fuite).
- Coude net entre `k=1` et `k=2`, puis la courbe s'aplatit → `k=2` est le compromis naturel.

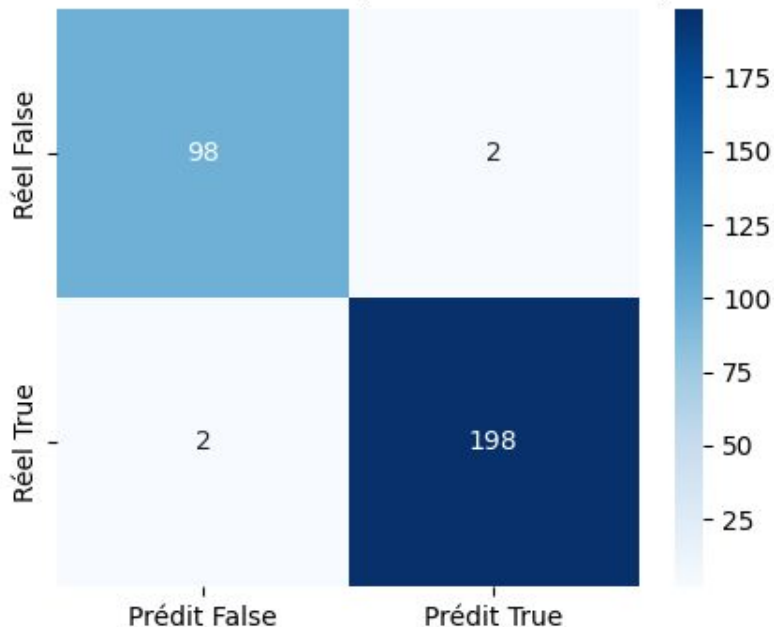
# K-means — séparation en 2 clusters

- Projection sur 2 variables standardisées (length, margin\_low).
- Deux nuages assez séparés + centroïdes distincts → données “clusterisables”, signal porté surtout par length/margin\_low.
- Limite : projection 2D ; le clustering réel se fait en 6D.



# K-means — évaluation (mapping)

Matrice de confusion — KMeans (évaluation sur test, sans fuite)



Pour comparer à des modèles supervisés, on transforme les clusters en classes :

- 296 / 300 billets bien classés → Accuracy  $\approx$  98,7%

Erreurs

- 2 faux billets prédits vrais (faux qui passent)
- 2 vrais billets prédits faux (vrais rejetés)

# Synthèse résultats des modèles

Modèle	Accuracy	Recall (False)	F1 (False)	Faux billets laissés passer (False→True)	Vrais billets rejetés (True→False)	AUC
Logistic Regression	0.990	0.980	0.985	2	1	1.000
Random Forest	0.990	0.980	0.985	2	1	0.999
KMeans (k=2)	0.987	0.980	0.980	2	2	nan
KNN (k=5)	0.983	0.970	0.975	3	2	0.997

- Les 4 modèles sont très performants (accuracy 0,983–0,990, F1(False) 0,975–0,985).
- Critère prioritaire ONCFM : minimiser les faux billets acceptés (False→True).
- Régression logistique et Random Forest : 2 faux billets laissés passer (meilleurs ex-aequo).
- KMeans : 2 aussi, mais 2 vrais billets rejetés (vs 1) et pas d'AUC (non supervisé).
- KNN : 3 faux billets laissés passer (moins bon sur le risque principal).
- Le Recall(False) est proche ( $\approx 0,97$ –0,98) : la décision se joue sur les erreurs concrètes et l'accès à une probabilité (AUC).

# Conclusion

## Décision (critère ONCFM)

- Priorité : minimiser les faux billets acceptés (False→True), puis limiter les vrais billets rejetés (True→False)

## Modèle retenu : Régression logistique

- Meilleur compromis sur le risque principal : 2 faux billets laissés passer (ex-aequo avec Random Forest)
- Erreur secondaire limitée : 1 vrai billet rejeté
- Modèle simple, interprétable, et seuil de décision ajustable selon le niveau de risque

## Suite

- Démonstration : utilisation du script pour prédire `is_genuine` à partir des mesures d'un billet scanné