

ONCFM

Organisation nationale
de lutte contre le
faux-monnayage

**CRÉATION D'UNE APPLICATION
DE DÉTECTION DE FAUX BILLETS**

SOMMAIRE

1. *Introduction*
2. *Les données initiales*
3. *Analyse des données*
4. *Régression linéaire multiple*
5. *Test des algorithmes*
 - 5.1 **Kmean**
 - 5.2 **Régression logistique**
 - 5.3 **KNN**
 - 5.4 **Random Forest**
- 6. *Application fonctionnelle*
- 7. *Conclusion*





1. INTRODUCTION

- **Contexte** : L'ONCFM met en place des méthodes d'identification des faux billets en euros pour lutter contre la contrefaçon.
- **Objectif de la mission** : Mettre à disposition des équipes une **application de machine learning** qui permettra, après avoir scanné des billets, de déterminer la nature des billets (**vrai billet ou faux billet**)



2. LES DONNÉES INITIALES

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

3. ANALYSE DES DONNÉES

	diagonal	height_left	height_right	margin_low	margin_up	length
count	1500.000000	1500.000000	1500.000000	1463.000000	1500.000000	1500.000000
mean	171.958440	104.029533	103.920307	4.485967	3.151473	112.67850
std	0.305195	0.299462	0.325627	0.663813	0.231813	0.87273
min	171.040000	103.140000	102.820000	2.980000	2.270000	109.49000
25%	171.750000	103.820000	103.710000	4.015000	2.990000	112.03000
50%	171.960000	104.040000	103.920000	4.310000	3.140000	112.96000
75%	172.170000	104.230000	104.150000	4.870000	3.310000	113.34000
max	173.010000	104.880000	104.950000	6.900000	3.910000	114.44000



- **Valeur manquantes:**

Margin_low = 37 valeur manquantes

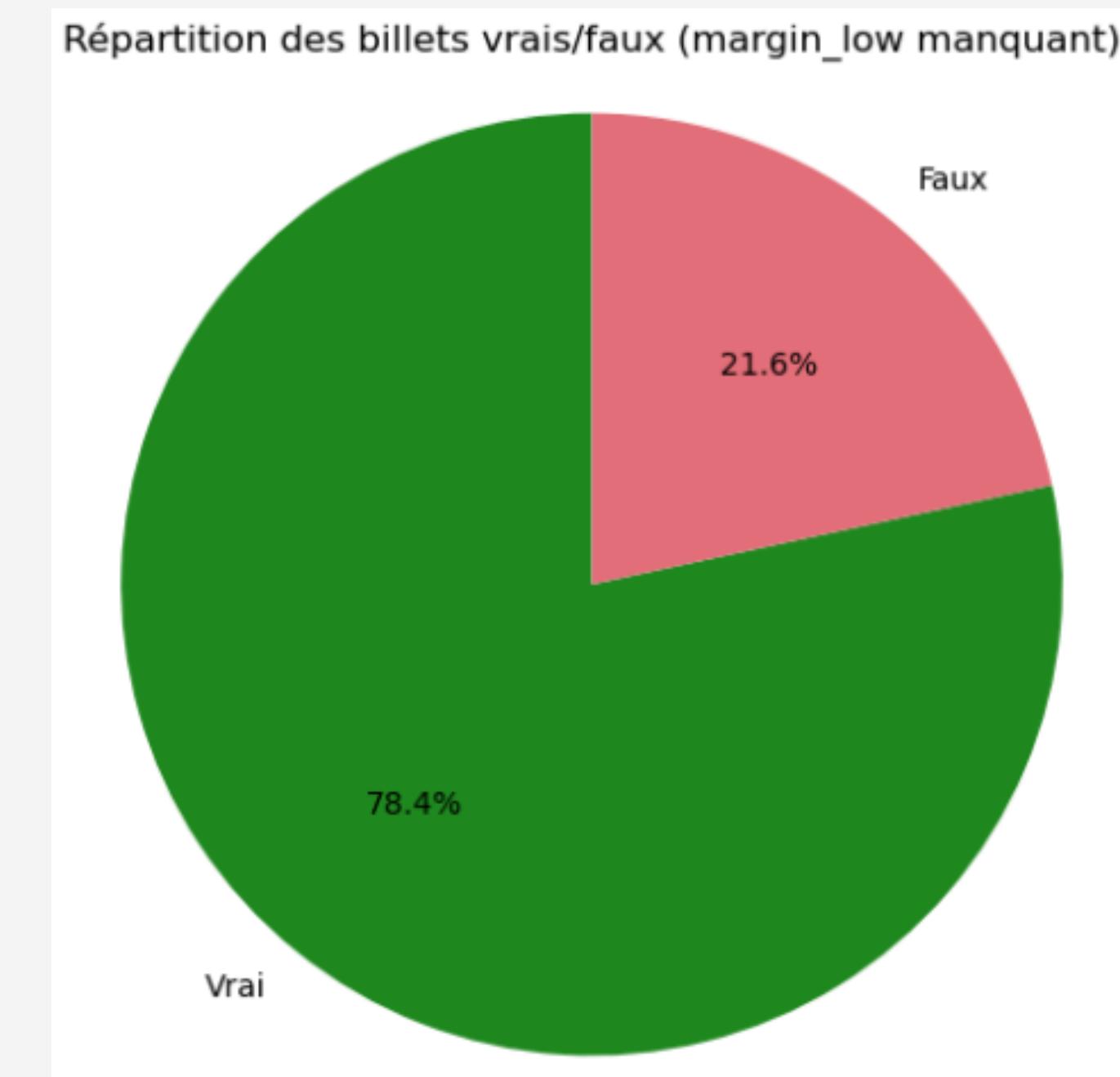
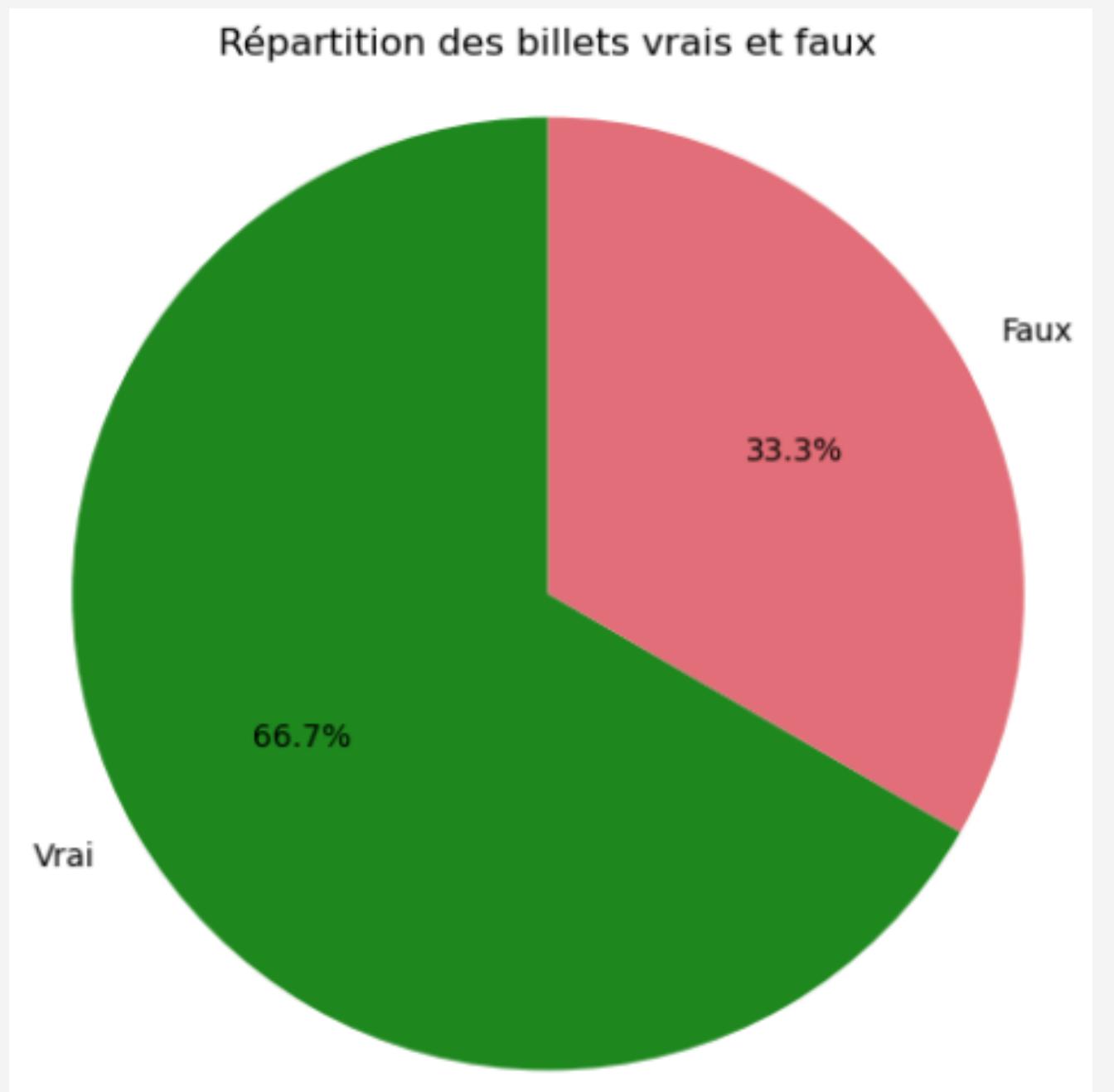


- **Solution:**

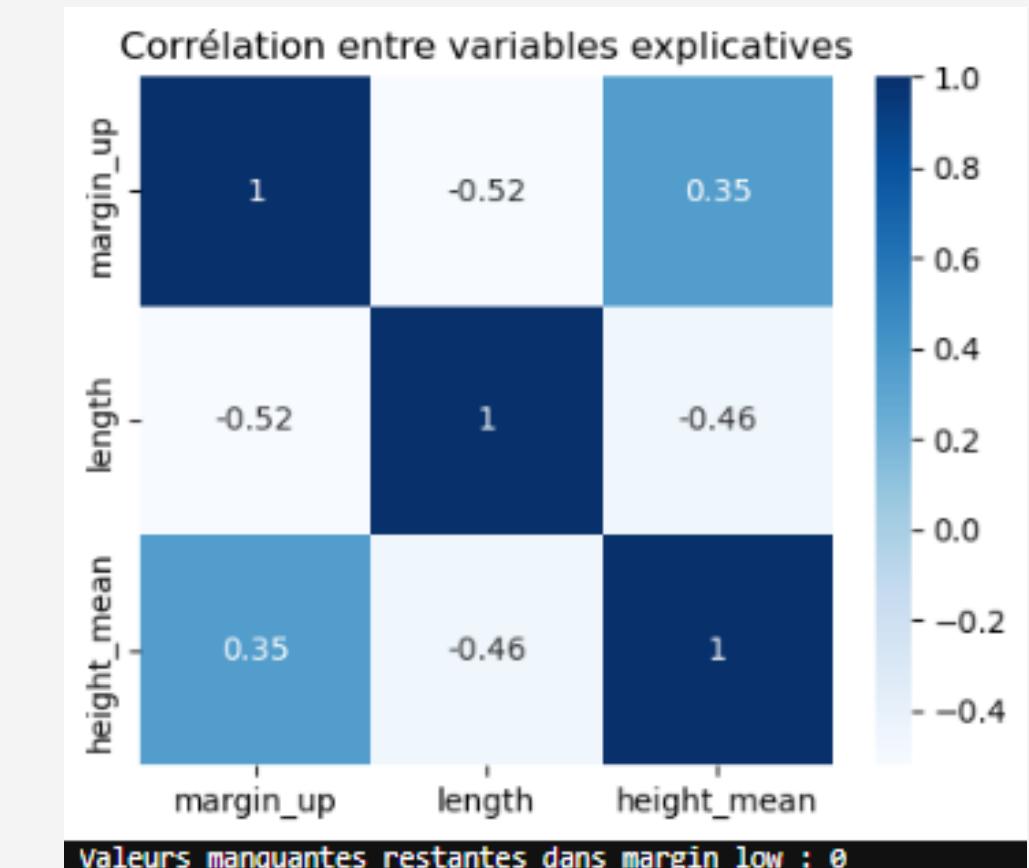
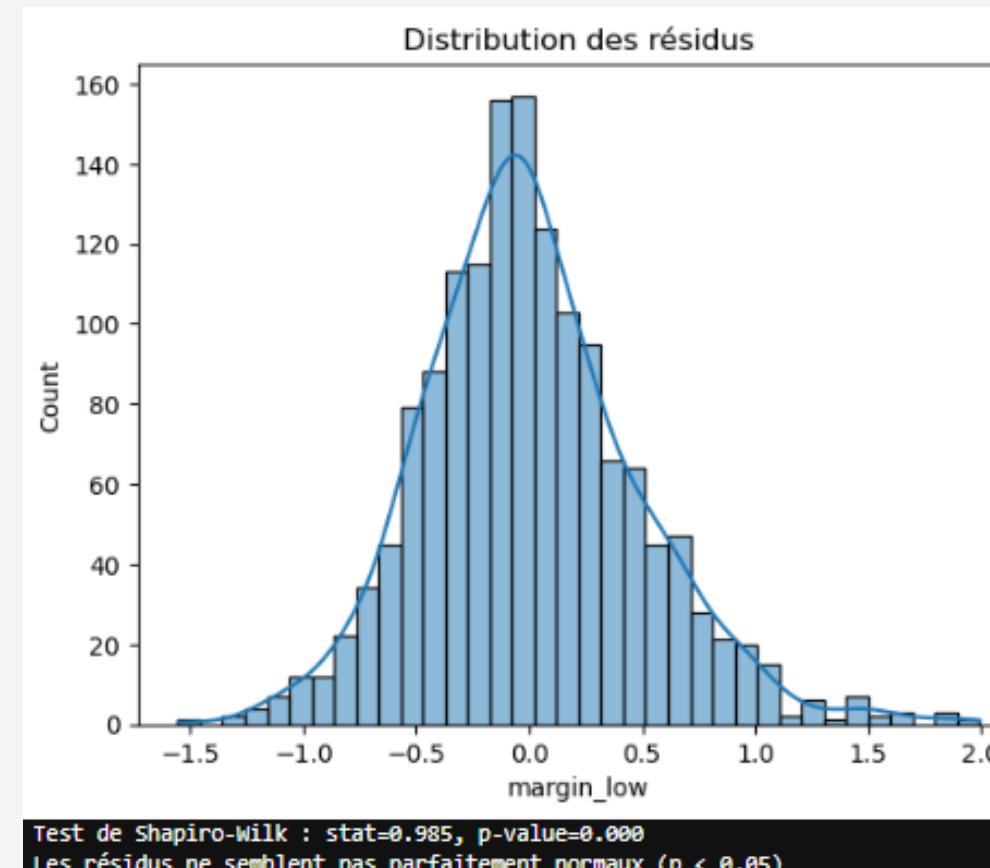
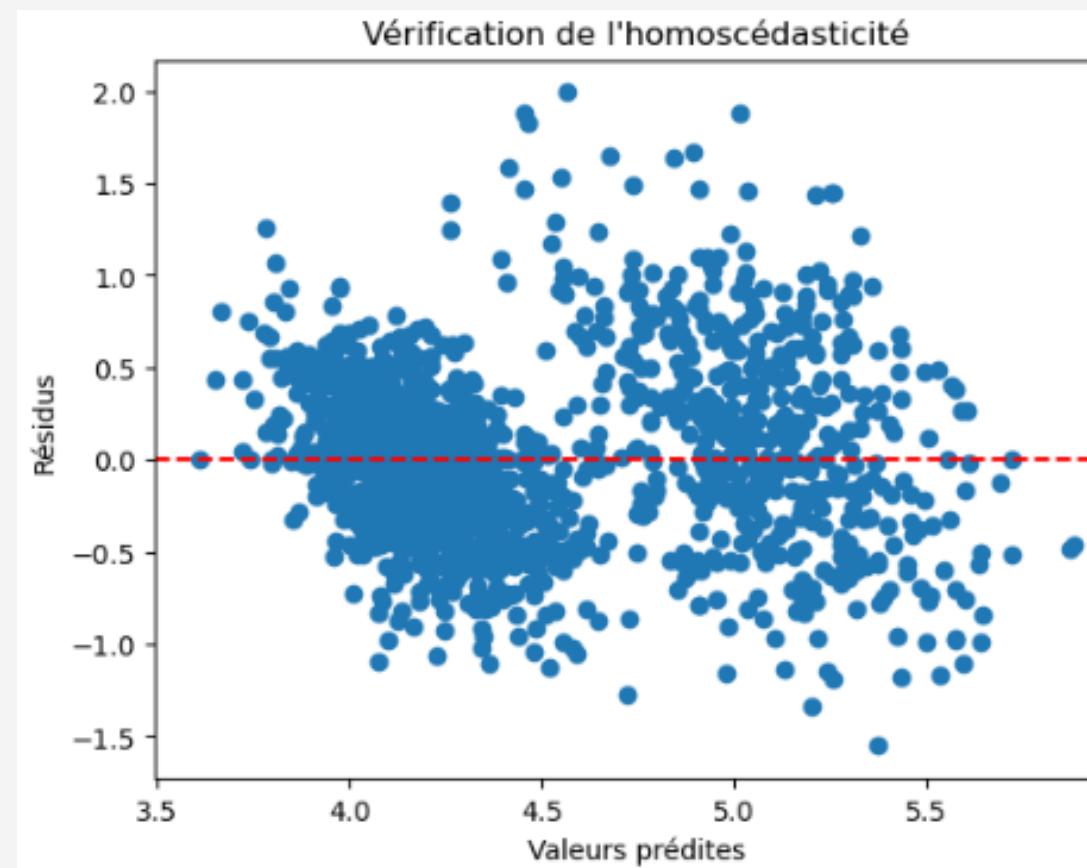
Compléter les valeurs manquantes

grâce à la Régression linéaire multiple

3. ANALYSE DES DONNÉES



4. RÉGRESSION LINÉAIRE MULTIPLE



4. RÉGRESSION LINÉAIRE MULTIPLE

```
VIF des variables explicatives :  
    Variable          VIF  
0   margin_up      260.694257  
1   length        16635.328184  
2   height_mean   19056.357739  
  
Coefficient de détermination R² : 0.481  
  
Coefficients de la régression :  
    Variable  Coefficient  
2   height_mean   0.436950  
0   margin_up      0.259612  
1   length        -0.414433
```



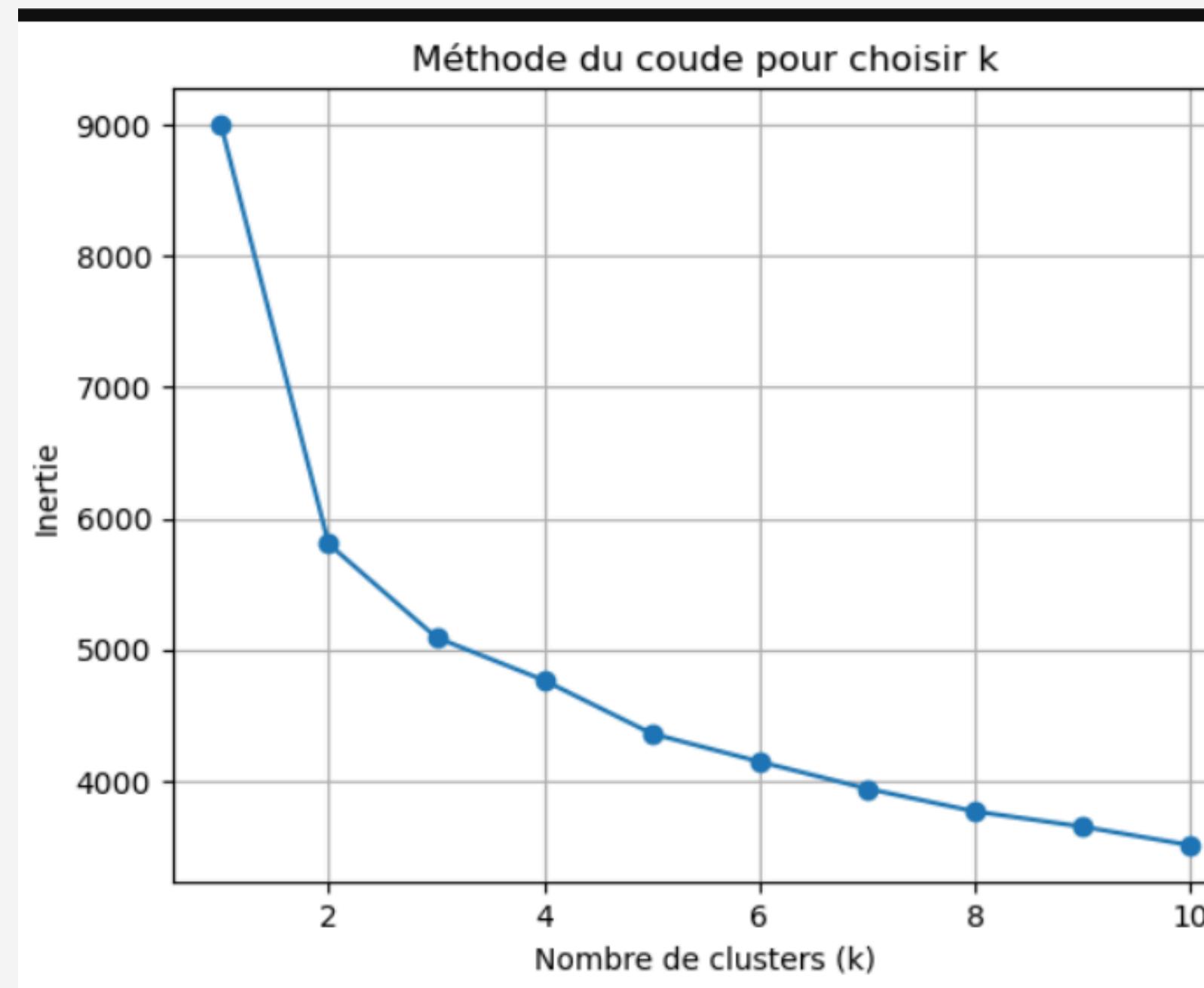
Le modèle de régression linéaire obtient un R^2 de 0.481, indiquant une précision moyenne.

Mais, les valeurs prédites restent cohérentes avec la tendance générale des données.

- **Seulement 37 valeurs sur 1500 sont manquantes**, l'impact de cette approximation est **négligeable** pour la suite de notre analyse.
- **Nous conservons donc les résultats de la régression linéaire** pour compléter les valeurs manquantes de margin_low

5.TEST DES ALGORITHMES

5.1 KMEAN



5.TEST DES ALGORITHMES

5.1 KMEAN

cluster	0	1
is_genuine		
False	14	486
True	990	10



```
#Evoluer la qualité du clustering
from sklearn.metrics import silhouette_score

score = silhouette_score(X_scaled, kmeans.labels_)
print("Score de silhouette : ", score)
```

Score de silhouette : 0.3427474069132479

5.TEST DES ALGORITHMES

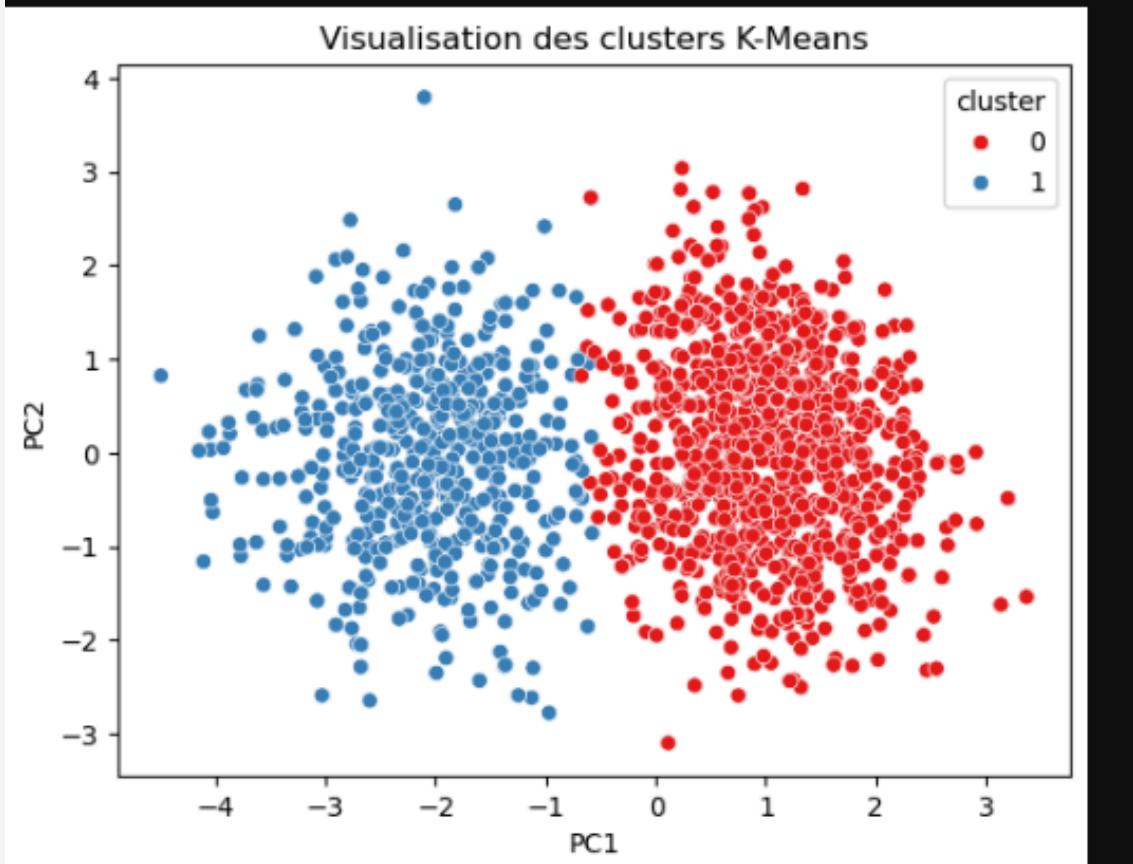
ACP

```
# Visualisation de l'ACP
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Ajoute les composantes au DataFrame
billets_clean['PC1'] = X_pca[:, 0]
billets_clean['PC2'] = X_pca[:, 1]

# Affiche les clusters
sns.scatterplot(data=billets_clean, x='PC1', y='PC2', hue='cluster', palette='Set1')
plt.title("Visualisation des clusters K-Means")
plt.show()
```

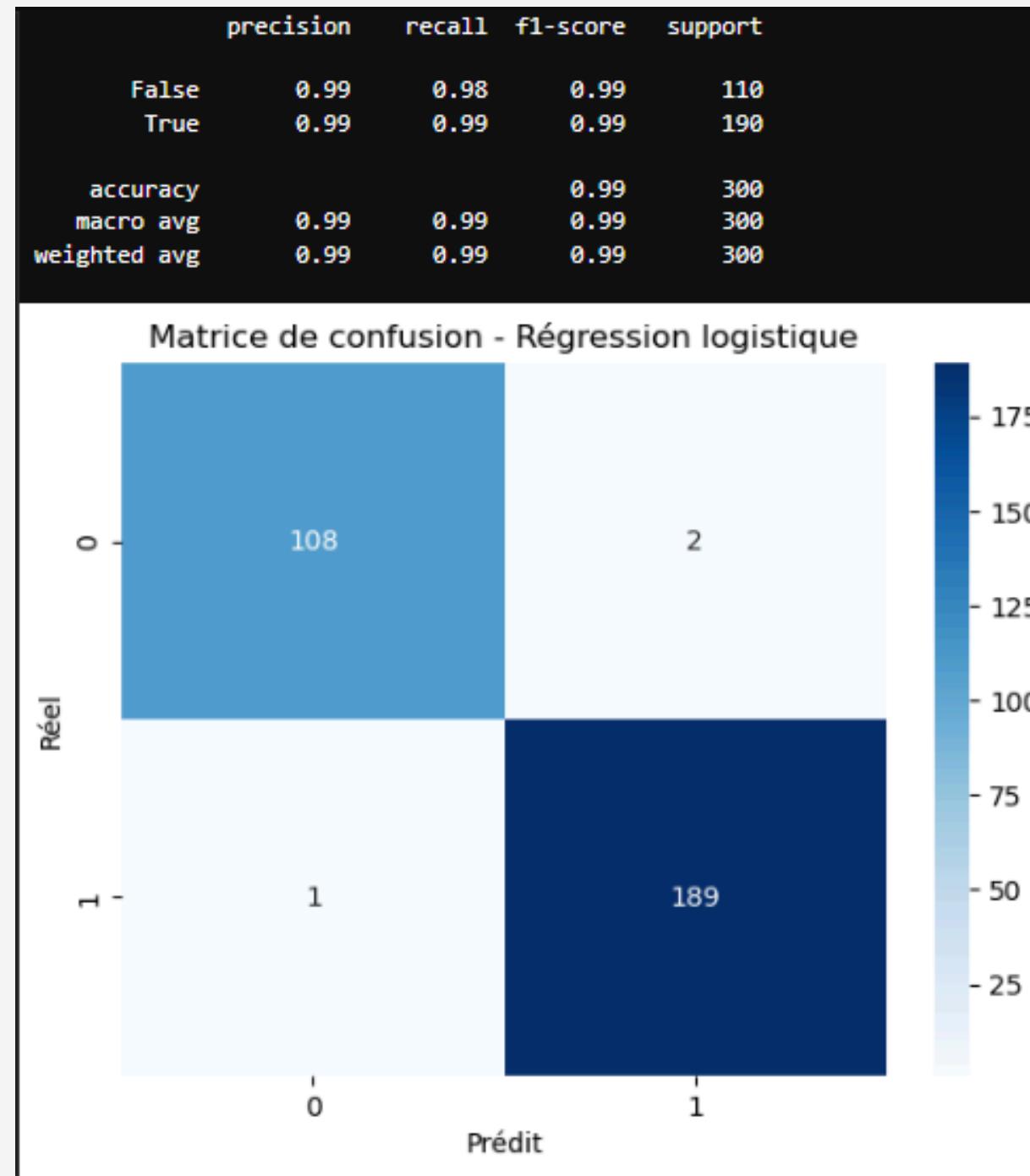


- K-Means ne connaît pas les vraies classes (*is_genuine*)
- Il regroupe les données par similarité, sans savoir ce qu'est un vrai ou un faux billet. Contrairement aux modèles supervisés, K-Means ne peut pas prédire si un nouveau billet est vrai ou faux.

Quand K-Means est utile:

- Pour explorer les données sans étiquettes
- Pour regrouper les observations selon leurs caractéristiques

5.2 RÉGRESSION LOGISTIQUE



Billets prédis comme faux :

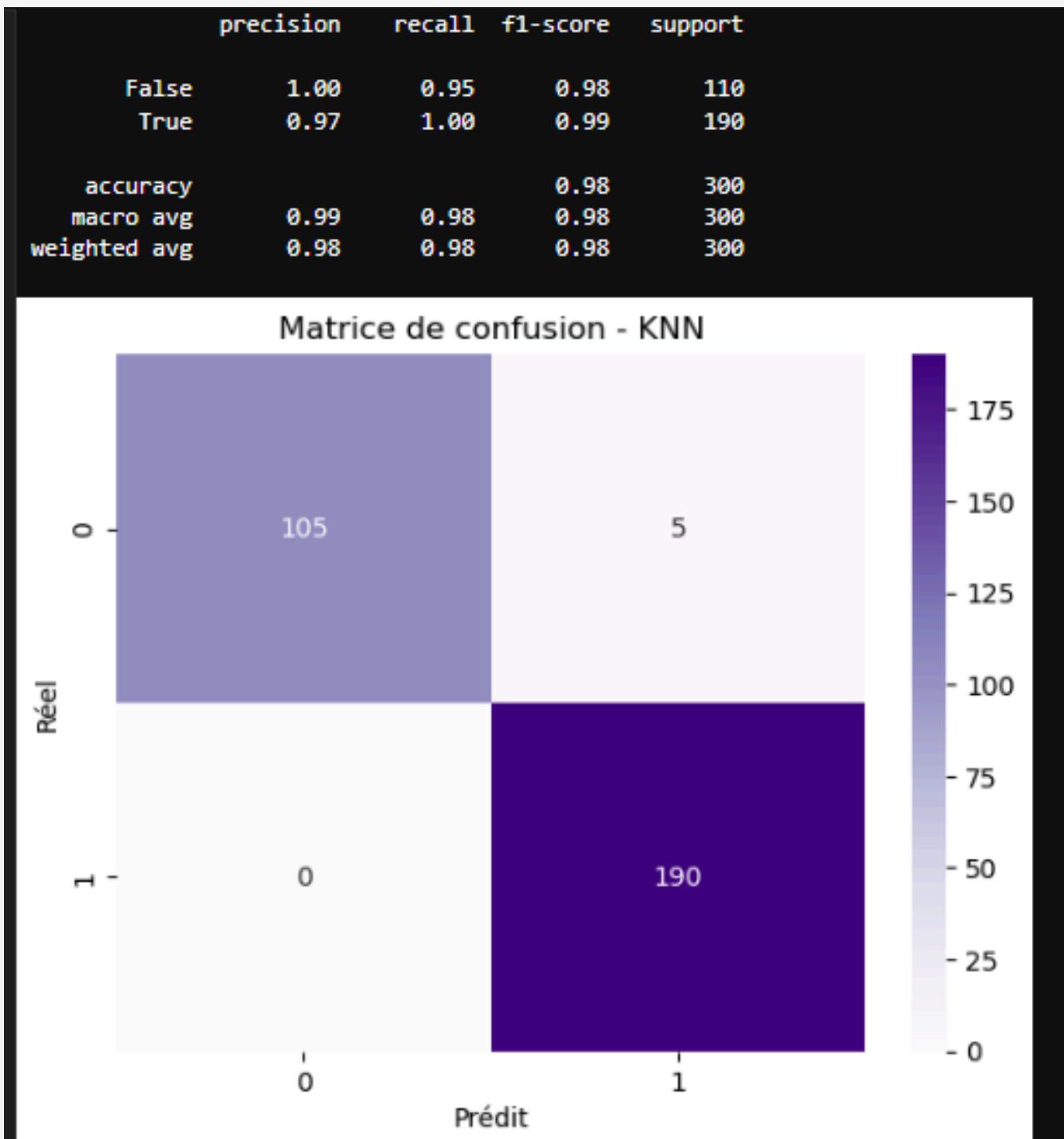
	diagonal	height_left	height_right	margin_low	margin_up	length	proba
0	171.76	104.01	103.54	5.21	3.30	111.42	0.001752
1	171.87	104.17	104.13	6.00	3.31	112.09	0.000183
2	172.00	104.58	104.29	4.99	3.39	111.57	0.000349

Résultats complets :

	diagonal	height_left	height_right	margin_low	margin_up	length	is_genuine_pred	proba
0	171.76	104.01	103.54	5.21	3.30	111.42	False	0.001752
1	171.87	104.17	104.13	6.00	3.31	112.09	False	0.000183
2	172.00	104.58	104.29	4.99	3.39	111.57	False	0.000349
3	172.49	104.55	104.34	4.44	3.03	113.20	True	0.970106
4	171.65	103.63	103.56	3.77	3.16	113.33	True	0.999768

5.TEST DES ALGORITHMES

5.3 KNN



5.4 RANDOM FOREST

Mise en place de la validation croisée

```
# Validation croisée
rf = RandomForestClassifier(n_estimators=100, random_state=42)

# Définir une validation croisée stratifiée pour garder le même équilibre de classes
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

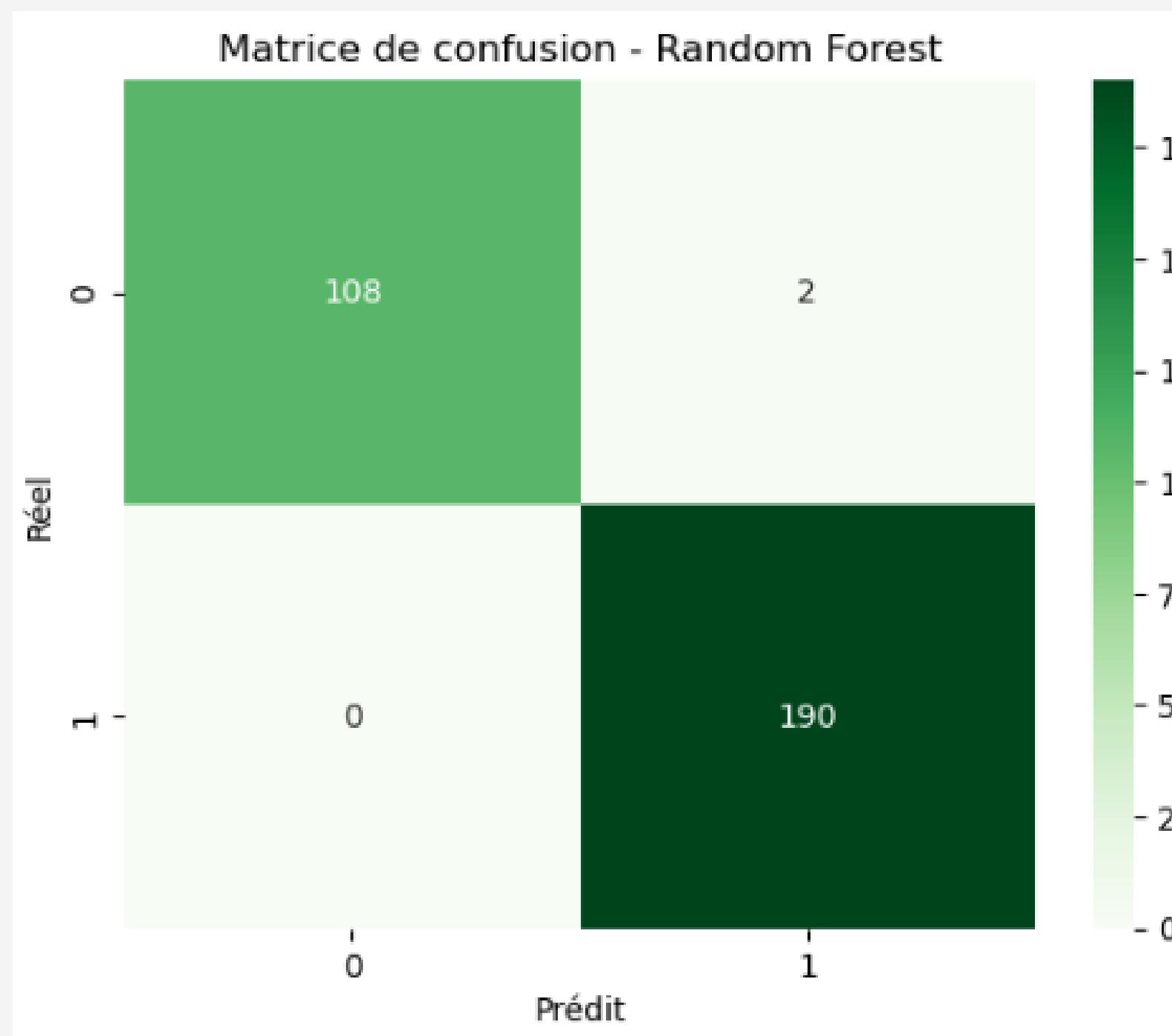
# Calculer les scores d'accuracy sur les 5 folds
scores = cross_val_score(rf, X_scaled, y, cv=cv, scoring='accuracy')
print("Scores de validation croisée : ", scores)
print("Moyenne de l'accuracy : ", scores.mean())
print("Écart-type : ", scores.std())

Scores de validation croisée : [0.99666667 0.98666667 0.98666667 0.99666667 0.98666667]
Moyenne de l'accuracy : 0.9906666666666666
Écart-type : 0.00489897948556636

#Split final train/test pour évaluation finale
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

#Entrainement du modèle final sur l'ensemble d'entraînement
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

5.4 RANDOM FOREST



Rapport de classification :

	precision	recall	f1-score	support
False	1.00	0.98	0.99	110
True	0.99	1.00	0.99	190
accuracy				0.99
macro avg	0.99	0.99	0.99	300
weighted avg	0.99	0.99	0.99	300

Billets prédis comme faux :

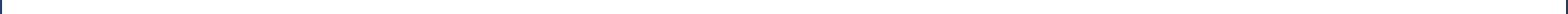
	diagonal	height_left	height_right	margin_low	margin_up	length	proba
0	171.76	104.01	103.54	5.21	3.30	111.42	0.01
1	171.87	104.17	104.13	6.00	3.31	112.09	0.00
2	172.00	104.58	104.29	4.99	3.39	111.57	0.00

Billets prédis comme authentiques :

	diagonal	height_left	height_right	margin_low	margin_up	length	proba
3	172.49	104.55	104.34	4.44	3.03	113.20	0.99
4	171.65	103.63	103.56	3.77	3.16	113.33	1.00

Random Forest a été retenu pour l'application finale

- **Précision élevée** : 99,3 % de réussite pour détecter les faux billets
- **Robuste au surapprentissage** : stable sur de nouvelles données
- **Gère les données complexes** : traite facilement plusieurs caractéristiques des billets
- **Interprétable** : identifie les caractéristiques les plus importantes
- **Résistant au bruit** : fiable malgré les variations ou anomalies



**MERCI POUR VOTRE
ATTENTION**

