Application détection de faux billets

Random Forest

In [3]:
```
# Import des librairies
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import StratifiedKFold, cross_val_score, train_test
import seaborn as sns
import matplotlib.pyplot as plt

# Chargement du fichier
billets_clean = pd.read_csv('billets_clean.csv', sep=',', encoding='utf-8')
billets_clean = billets_clean.dropna()
```

In [4]:
```
#Affichage du df
billets_clean
```

Out[4]:

	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
...	...	...	...	...	...	...	...
1495	False	171.75	104.38	104.17	4.42	3.09	111.28
1496	False	172.19	104.63	104.44	5.27	3.37	110.97
1497	False	171.80	104.01	104.12	5.51	3.36	111.95
1498	False	172.06	104.28	104.06	5.17	3.46	112.25
1499	False	171.47	104.15	103.82	4.63	3.37	112.07

1500 rows × 7 columns

In [5]:
```
# Définir X et y
X = billets_clean.drop(columns=['is_genuine'])
y = billets_clean['is_genuine']
```

In [6]:
```
#Standardisation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Mise en place de la validation croisée

In [8]:
```
# Validation croisée
rf = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [9]:
```
# Définir une validation croisée stratifiée pour garder le même équilibre de cla
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

In [10]:
```
# 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 0.986
66667]
Moyenne de l'accuracy : 0.9906666666666666
Écart-type : 0.00489897948556636
```

In [11]:
```
#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,
```

In [12]:
```
#Entraînement du modèle final sur l'ensemble d'entraînement
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
```

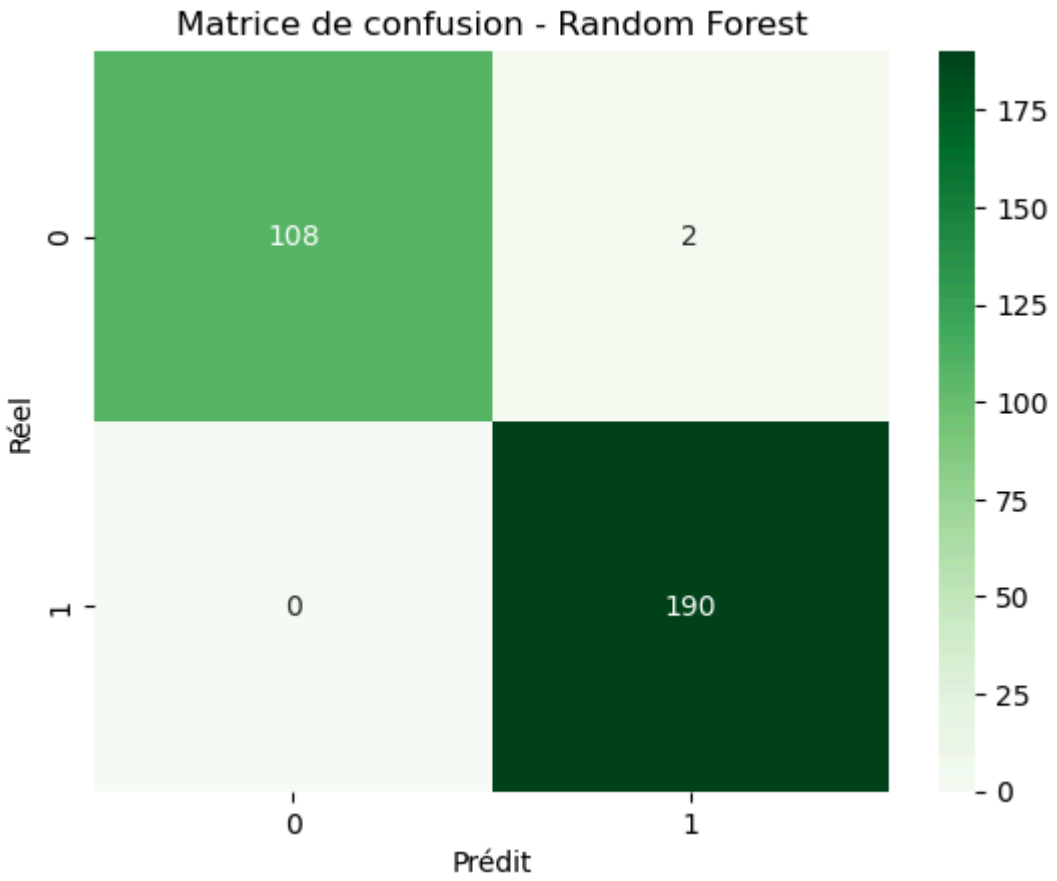
In [13]:
```
#Évaluation finale
print("\nRapport de classification :")
print(classification_report(y_test, y_pred))
```

```
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       300
   macro avg       0.99      0.99      0.99       300
weighted avg       0.99      0.99      0.99       300
```

In [14]:
```
#Affichage de la heatmap
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Greens'
plt.title("Matrice de confusion - Random Forest")
plt.xlabel("Prédit")
plt.ylabel("Réel")
plt.show()
```



Matrice de confusion du modèle Random Forest :

Cette matrice évalue la capacité du modèle Random Forest à détecter les billets vrais (0) et faux (1).

- 108 vrais billets correctement identifiés (case en haut à gauche)
- 190 faux billets correctement identifiés (case en bas à droite)
- 2 faux billets classés à tort comme vrais (case en haut à droite)
- 0 vrai billet classé à tort comme faux (case en bas à gauche)

Performance : La précision globale est de 99,3 %

Application

Chargement du fichier à tester

In [18]:
```
# Chargement du fichier à tester
nouveau_billet = pd.read_csv('billets_production.csv', sep=',', encoding='utf-8'
```

In [19]:
```
#Affichage du df à tester
nouveau_billet
```

Out[19]:

	diagonal	height_left	height_right	margin_low	margin_up	length	id
0	171.76	104.01	103.54	5.21	3.30	111.42	A_1
1	171.87	104.17	104.13	6.00	3.31	112.09	A_2
2	172.00	104.58	104.29	4.99	3.39	111.57	A_3
3	172.49	104.55	104.34	4.44	3.03	113.20	A_4
4	171.65	103.63	103.56	3.77	3.16	113.33	A_5

In [20]:
```
# Colonnes utilisées
colonnes = ['diagonal', 'height_left', 'height_right', 'margin_low', 'margin_up'

# Vérification que toutes les colonnes existent
missing_cols = [col for col in colonnes if col not in nouveau_billet.columns]
if missing_cols:
    raise ValueError(f"Colonnes manquantes dans le DataFrame : {missing_cols}")

X_new = nouveau_billet[colonnes].copy()

# Standardiser
X_scaled_new = scaler.transform(X_new)

# Prédictions
nouveau_billet['is_genuine_pred'] = rf.predict(X_scaled_new)
nouveau_billet['proba'] = rf.predict_proba(X_scaled_new)[:, 1]
```

In [21]:
```
# Colonnes utilisées
colonnes = ['diagonal', 'height_left', 'height_right', 'margin_low', 'margin_up'
X_new = nouveau_billet[colonnes]

# Standardiser
X_scaled_new = scaler.transform(X_new)

# Prédictions
nouveau_billet['is_genuine_pred'] = rf.predict(X_scaled_new)
nouveau_billet['proba'] = rf.predict_proba(X_scaled_new)[:, 1]
```

In [22]:
```
# Faux billets
faux_billets = nouveau_billet[nouveau_billet['is_genuine_pred'] == False]
print("Billets prédits comme faux :")
display(faux_billets[colonnes + ['proba']])

# Vrais billets
vrais_billets = nouveau_billet[nouveau_billet['is_genuine_pred'] == True]
print("Billets prédits comme authentiques :")
display(vrais_billets[colonnes + ['proba']])
```

```
Billets prédits 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 prédits 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