Développer un algorithme capable de détecter des faux billets

03/04/2025

Sommaire

ONCFM

- Contexte
- Traitement et analyse des données
- Modèles de prédiction
- Choix du modèle final
- Mise en application

Contexte

ONCFM

- L’ONCFM souhaite automatiser la détection de faux billets
- Analyser chaque billet selon ses caractéristiques géométriques
  - Développer des algorithmes des prédiction efficaces
  - Choisir le plus efficace et le livrer dans un notebook

Traitement et analyse des données

ONCFM

Importation des données

- 1500 billets scannés (1000 vrais et 500 faux)

- 7 variables

- is_genuine
- length
- height_left
- height_right
- margin_up
- margin_low
- diagonal

Traitement et analyse des données

ONCFM

Analyse exploratoire globale

	Authenticite	Diagonale	Hauteur_gauche	Hauteur_droite	Marge_basse	Marge_haute	Longueur
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

```
Nature des données dans chacune des colonnes :
is_genuine          bool
diagonal         float64
height_left      float64
height_right     float64
margin_low       float64
margin_up        float64
length           float64
```

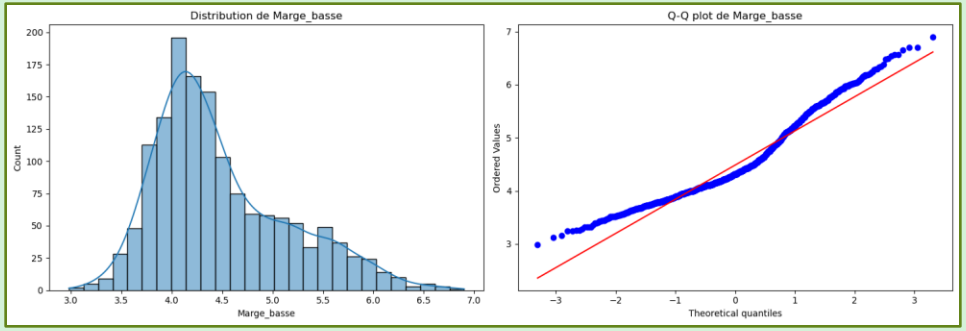
	Authenticite	Diagonale	Hauteur_gauche	Hauteur_droite	Marge_basse	Marge_haute	Longueur
count	1500.000000	1500.000000	1500.000000	1500.000000	1463.000000	1500.000000	1500.00000
mean	0.666667	171.958440	104.029533	103.920307	4.485967	3.151473	112.67850
std	0.471562	0.305195	0.299462	0.325627	0.663813	0.231813	0.87273
min	0.000000	171.040000	103.140000	102.820000	2.980000	2.270000	109.49000
25%	0.000000	171.750000	103.820000	103.710000	4.015000	2.990000	112.03000
50%	1.000000	171.960000	104.040000	103.920000	4.310000	3.140000	112.96000
75%	1.000000	172.170000	104.230000	104.150000	4.870000	3.310000	113.34000
max	1.000000	173.010000	104.880000	104.950000	6.900000	3.910000	114.44000

```
Nombre de valeurs présentes dans chacune des colonnes :
is_genuine      1500
diagonal        1500
height_left     1500
height_right    1500
margin_low      1463
margin_up       1500
length          1500
```

Traitement et analyse des données

ONCFM

Régression linéaire

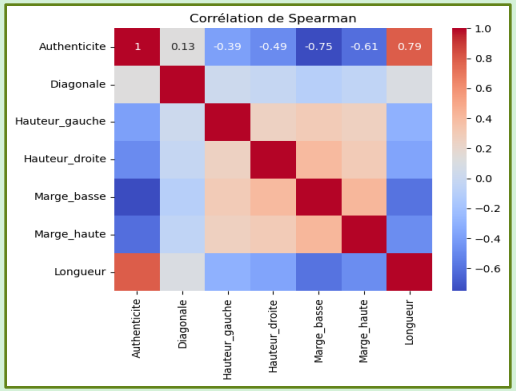


OLS Regression Results

Dep. Variable:	Marge_basse	R-squared:	0.618
Model:	OLS	Adj. R-squared:	0.617
Method:	Least Squares	F-statistic:	942.0
Date:	Tue, 01 Apr 2025	Prob (F-statistic):	2.89e-244
Time:	00:03:32	Log-Likelihood:	-606.17
No. Observations:	1170	AIC:	1218.
Df Residuals:	1167	BIC:	1234.
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	5.6622	0.220	25.769	0.000	5.231	6.093
Authenticite	-1.1373	0.032	-35.354	0.000	-1.200	-1.074
Marge_haute	-0.1337	0.065	-2.051	0.040	-0.262	-0.006

Omnibus:	21.780	Durbin-Watson:	1.948
Prob(Omnibus):	0.000	Jarque-Bera (JB):	41.200
Skew:	0.034	Prob(JB):	1.13e-09
Kurtosis:	3.917	Cond. No.	65.4



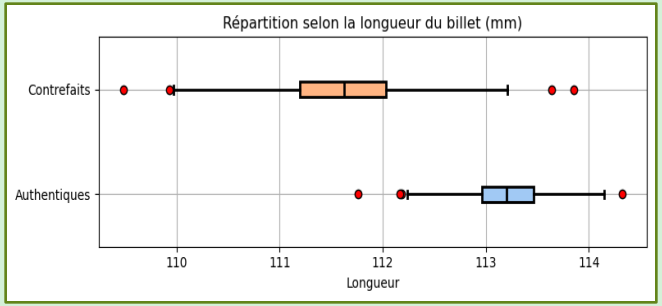
Tests de validité

- Colinéarité des variables
- Homoscédasticité des résidus
- Normalité des résidus

Traitement et analyse des données

ONCFM

Analyse de chaque variable

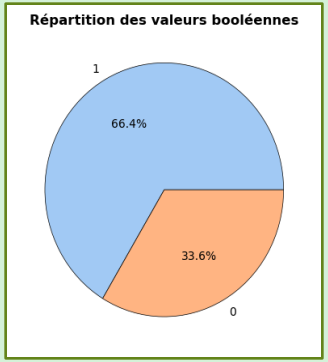


Outliers authentiques (4) :

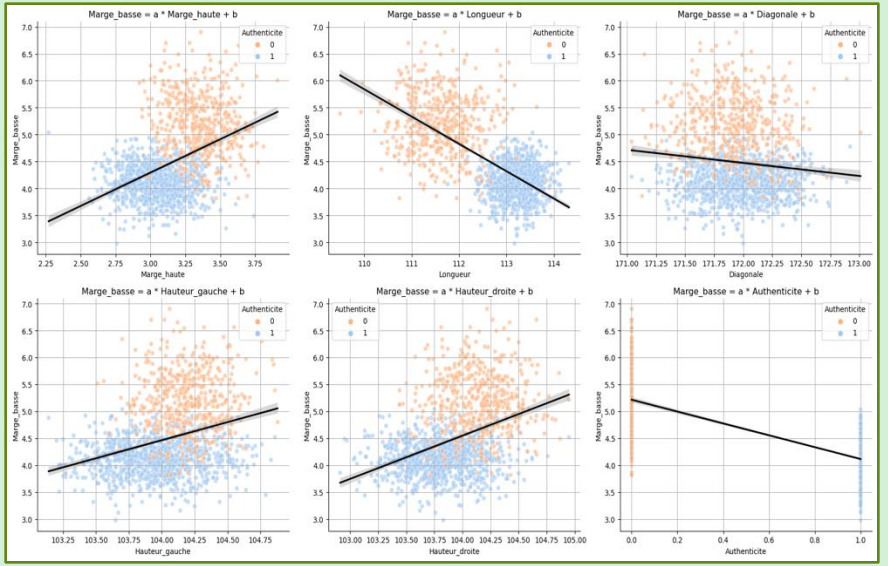
	Authenticite	Diagonale	Hauteur_gauche	Hauteur_droite	Marge_basse	Marge_haute	Longueur
578	1	171.67	103.81	103.76	4.59	3.30	112.18
711	1	171.94	104.11	104.16	4.08	3.35	111.76
923	1	171.95	103.86	103.80	4.51	2.87	112.17
946	1	172.06	104.08	103.47	4.47	2.97	114.32

Outliers contrefaits (4) :

	Authenticite	Diagonale	Hauteur_gauche	Hauteur_droite	Marge_basse	Marge_haute	Longueur
1052	0	171.75	103.96	103.83	5.39	3.54	109.49
1091	0	172.09	104.15	104.17	4.15	3.40	113.85
1291	0	171.83	104.39	104.17	5.51	3.33	113.64
1416	0	171.55	104.20	104.49	5.42	3.54	109.93




```
Authenticite
0    111.632114
1    113.203059
Name: Longueur,
```



Modèles de prédiction

ONCFM

Comment entraîner un modèle ?

```
# Affecter les bonnes données et retirer les variables non significatives
X = billets.drop(["Authenticite","Diagonale","Hauteur_gauche"], axis=1)
y = billets["Authenticite"]
```

```
# Définir le train set et le test set
X_train ,X_test, y_train, y_test = train_test_split(X,y, train_size=0.8, random_state=0)
```

```
# Définir notre modèle
modele_RL = LogisticRegression(max_iter=1000)
```

```
# Entraîner le modèle
model_marge_basse = sm.OLS(y_train, X_train).fit()
```

```
# Enregistrer les prédictions dans une variable y_pred
y_pred = modele_RL.predict(X_test)
```

Modèles de prédiction

ONCFM

Régression logistique – Définition et mise en place

➢ **Principe**

Cherche à estimer la probabilité qu'un événement se produise.

```
_____________________________
Authenticite ~ Marge_basse + Marge_haute + Longueur + Hauteur_droite + Diagonale + Hauteur_gauche + 1
Optimization terminated successfully.
         Current function value: 0.026765
         Iterations 12
remove Diagonale (p-value: 0.728 )


_____________________________
Authenticite ~ Marge_basse + Marge_haute + Longueur + Hauteur_droite + Hauteur_gauche + 1
Optimization terminated successfully.
         Current function value: 0.026807
         Iterations 13
remove Hauteur_gauche (p-value: 0.294 )


_____________________________
Authenticite ~ Marge_basse + Marge_haute + Longueur + Hauteur_droite + 1
Optimization terminated successfully.
         Current function value: 0.027189
         Iterations 13
is the final model!
```

Logit Regression Results

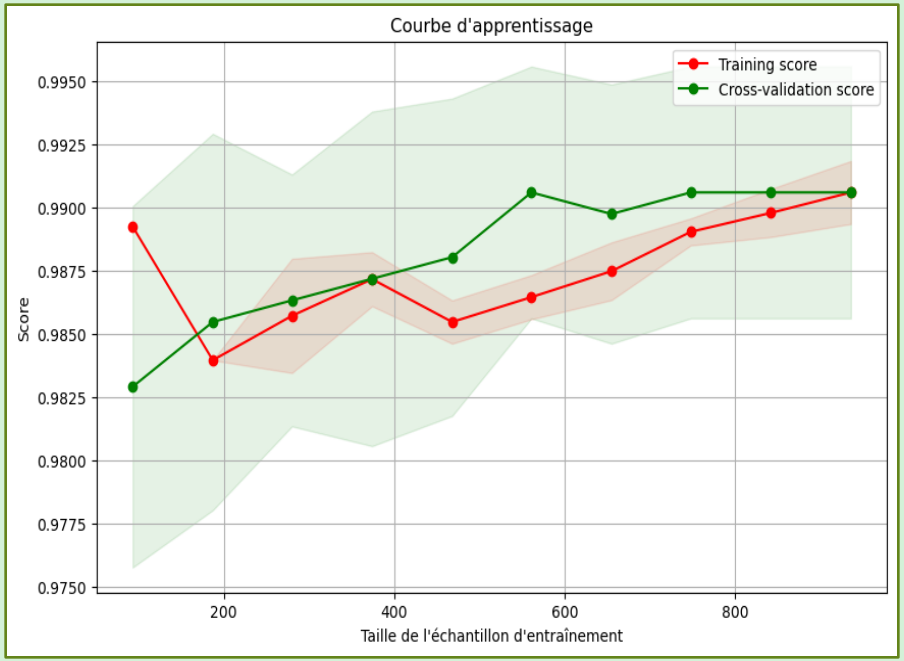
Dep. Variable:	Authenticite	No. Observations:	1463
Model:	Logit	Df Residuals:	1458
Method:	MLE	Df Model:	4
Date:	Tue, 01 Apr 2025	Pseudo R-squ.:	0.9574
Time:	15:26:23	Log-Likelihood:	-39.777
converged:	True	LL-Null:	-934.20
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-258.7502	143.318	-1.805	0.071	-539.649	22.148
Marge_basse	-6.1958	0.959	-6.461	0.000	-8.075	-4.316
Marge_haute	-10.3632	2.189	-4.735	0.000	-14.653	-6.074
Longueur	6.1300	0.886	6.917	0.000	4.393	7.867
Hauteur_droite	-3.5542	1.189	-2.990	0.003	-5.884	-1.225

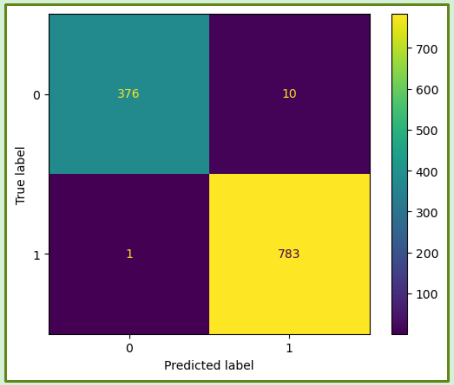
Modèles de prédiction

ONCFM

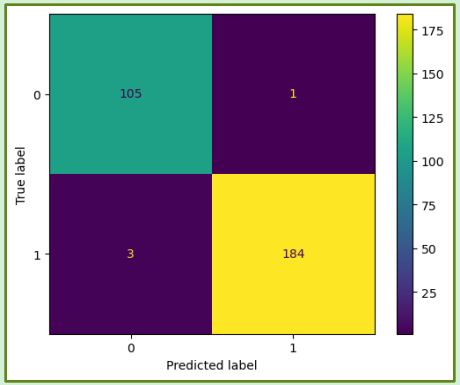
Régression logistique - Résultats



Train



Test



Performance globale

```
Précision (Accuracy): 0.99
Précision (Precision): 0.99
Rappel (Recall): 0.98
Spécificité (Specificity): 0.99
Score F1: 0.99
Le modèle a une erreur de 0.014%.
```

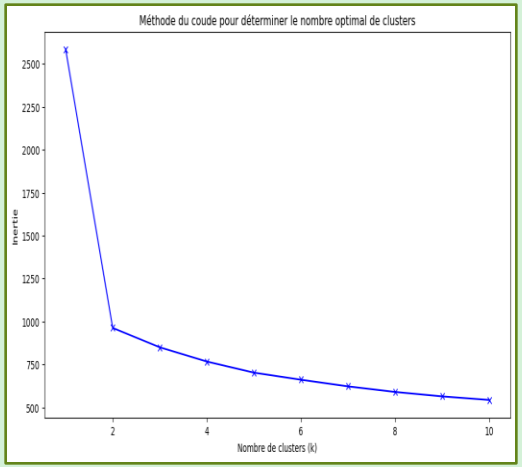
Modèles de prédiction

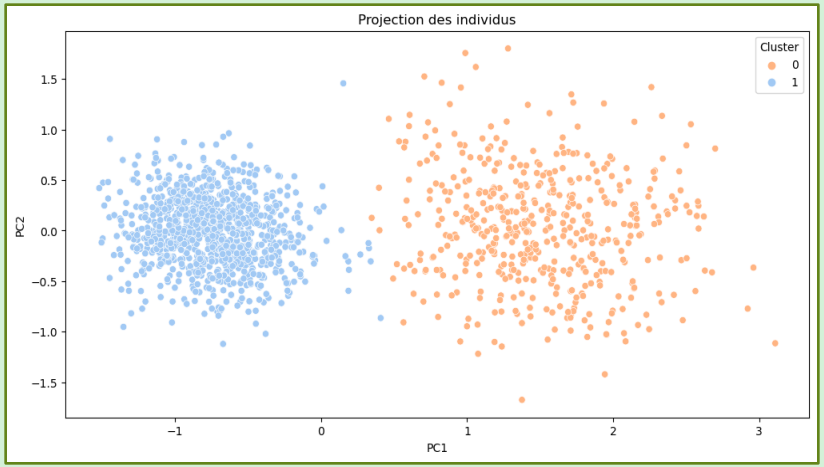
ONCFM

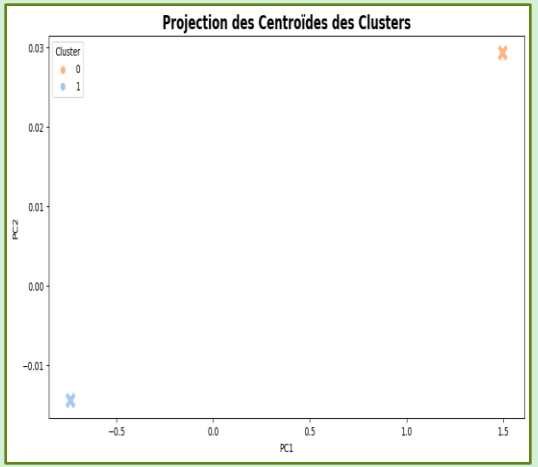
K-means - Définition et mise en place

➤ **Principe**

L'algorithme cherche à minimiser la variance intra-cluster.





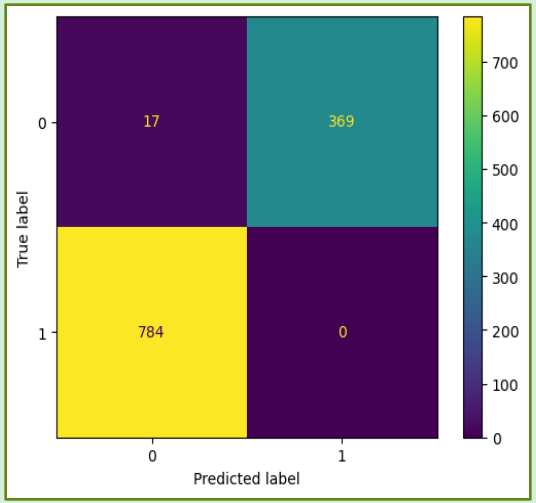


Modèles de prédiction

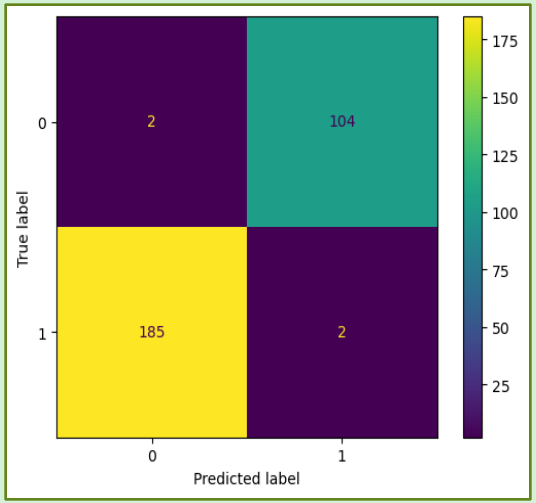
ONCFM

K-means - Résultats

Train



Test



Stabilité et cohérence des clusters

```
ARI (Adjusted Rand Index) pour train : 0.94
NMI (Normalized Mutual Information) pour train : 0.90
ARI (Adjusted Rand Index) pour test : 0.95
NMI (Normalized Mutual Information) pour test : 0.89
```

Performance globale

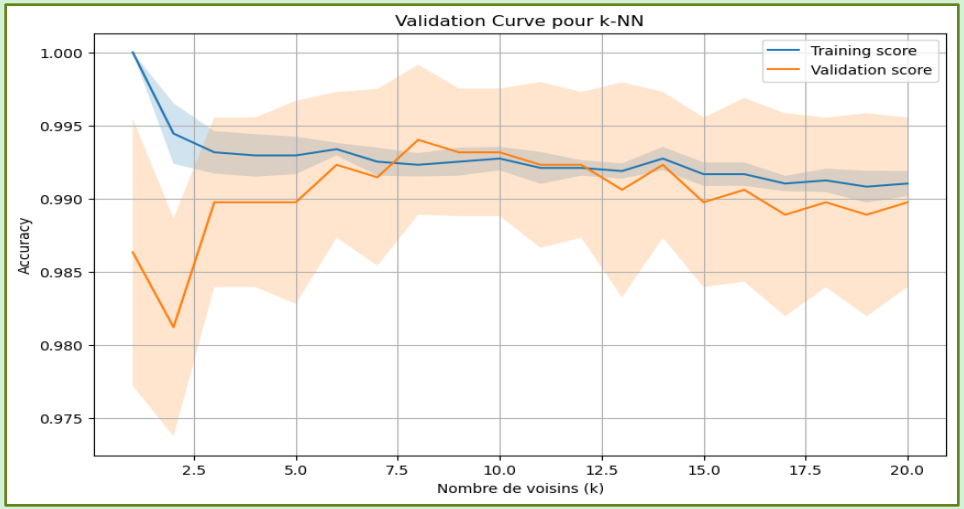
```
Précision (Accuracy): 0.99
Précision (Precision): 0.98
Rappel (Recall): 1.00
Spécificité (Specificity): 0.96
Score F1: 0.99
Le modèle a une erreur de 0.014%.
```

ONCFM

KNN - Définition et mise en place

Principe

Un point est classé selon la majorité des classes de ses k voisins les plus proches dans l'espace des données.

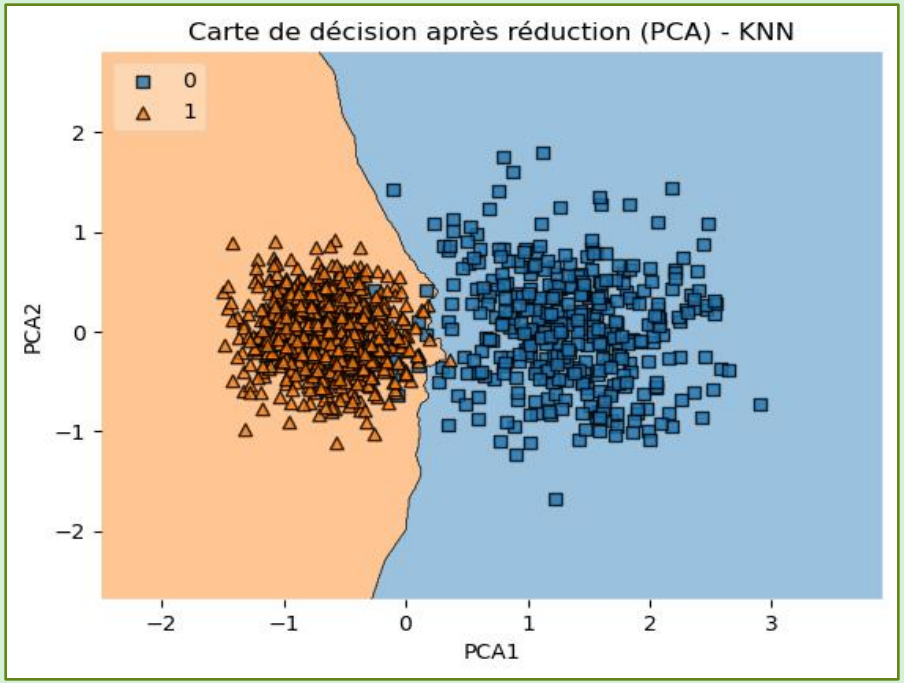



```
Meilleur k : 8
Meilleure accuracy moyenne : 0.994017094017094
```

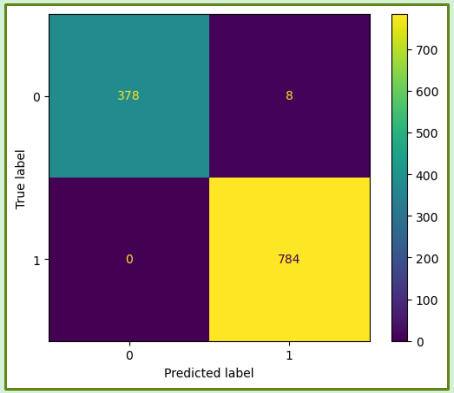
Modèles de prédiction

ONCFM

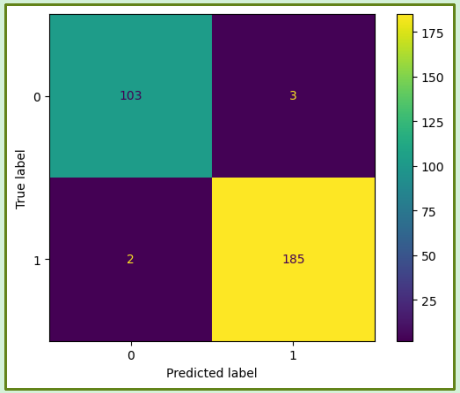
KNN - Résultats



Train



Test



Performance globale

```
Précision (Accuracy): 0.98
Précision (Precision): 0.98
Rappel (Recall): 0.99
Spécificité (Specificity): 0.97
Score F1: 0.99
Le modèle a une erreur de 0.017%.
```

ONCFM

Random Forest - Définition et mise en place

➢ Principe

Fonctionne en construisant plusieurs arbres de décision à partir de sous-échantillons aléatoires du jeu de données.

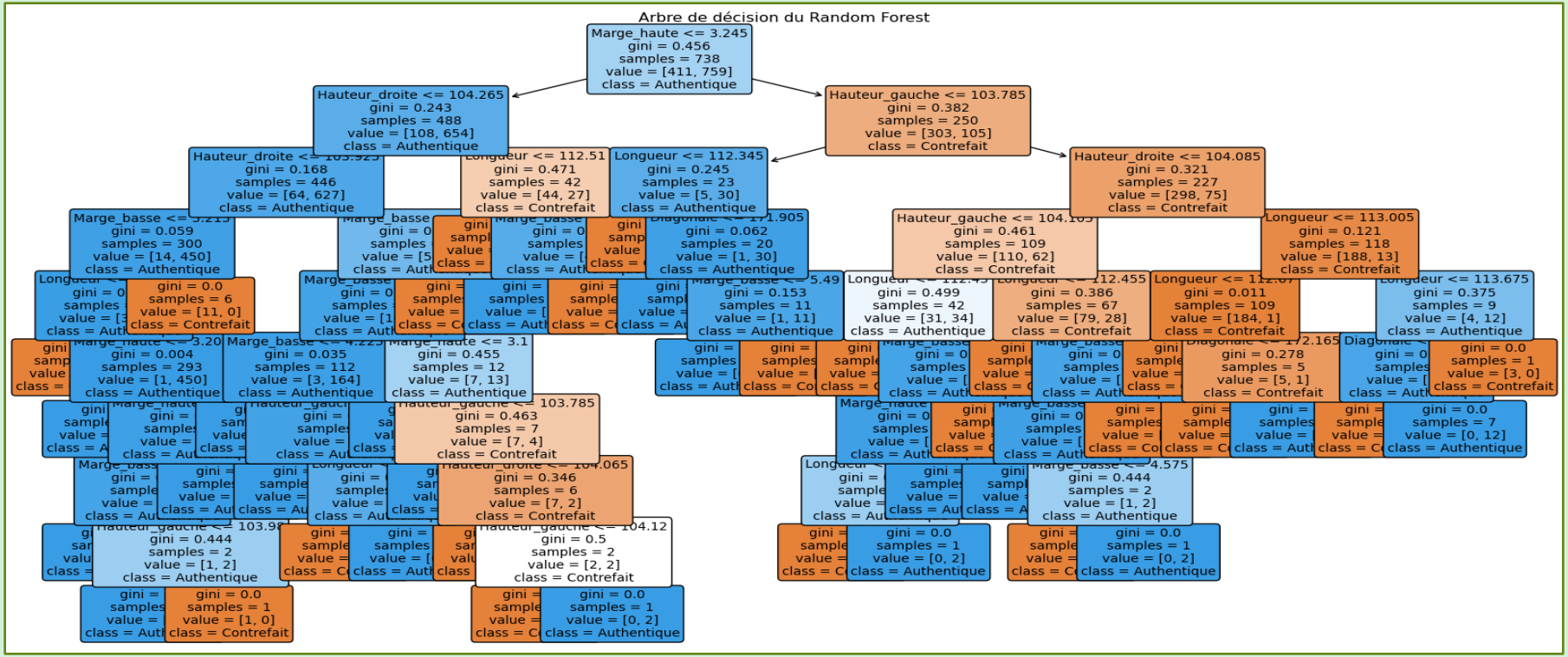
```
# Définir le modèle Random Forest
modele_RF = RandomForestClassifier(
      n_estimators=100,
      criterion='gini',
      max_depth=None,
      min_samples_split=2,
      min_samples_leaf=1,
      min_weight_fraction_leaf=0.0,
      max_features='sqrt',
      max_leaf_nodes=None,
      min_impurity_decrease=0.0,
      bootstrap=True,
      oob_score=False,
      n_jobs=None,
      random_state=None,
      verbose=0,
      warm_start=False,
      class_weight=None,
      ccp_alpha=0.0,
      max_samples=None,)
```

	importance
Longueur	0.460755
Marge_basse	0.319074
Marge_haute	0.110672
Hauteur_droite	0.061562
Hauteur_gauche	0.038114
Diagonale	0.009824

Modèles de prédiction

ONCFM

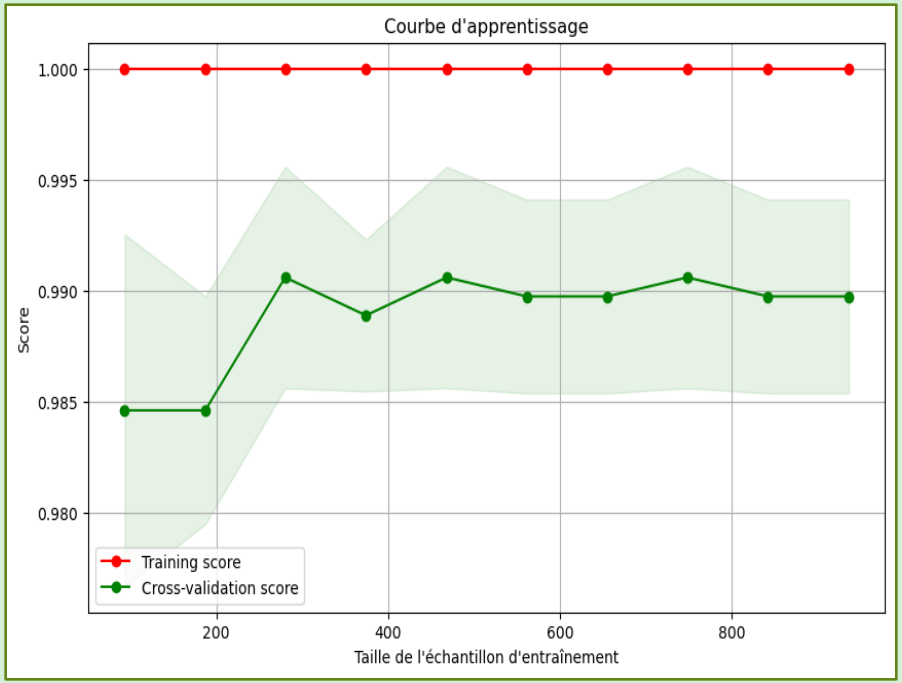
Random Forest – Arbre de décision



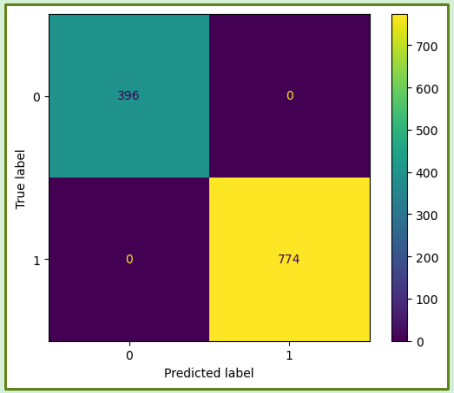
Modèles de prédiction

ONCFM

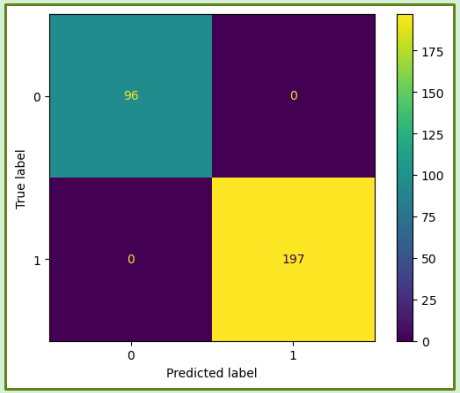
Random Forest - Résultats



Train



Test



Performance globale

```
Précision (Accuracy): 1.00
Précision (Precision): 1.00
Rappel (Recall): 0.99
Spécificité (Specificity): 1.00
Score F1: 1.00
Le modèle a une erreur de 0.003%.
```

Choix du modèle final

ONCFM

	Modèle	Modèle	f1_score
0	Régression Logistique	LogisticRegression(max_iter=1000)	0.989247
1	KMeans	KMeans(n_clusters=2, n_init=10, random_state=42)	0.989280
2	KNN	KNeighborsClassifier(n_neighbors=8)	0.986667
3	Random Forest	(DecisionTreeClassifier(max_features='sqrt', r...	0.997455

➢ **CHOIX : RANDOM FOREST**

Il présente les meilleurs résultats en termes de précision, de rappel et de score F1, tout en étant le plus stable et le plus robuste.

Mise en application

ONCFM

```
# Importation du fichier billets_production.csv
chemin_nouveau_fichier = '/Users/adrianaguilera/Desktop/P12/Inputs/billets_production.csv'
nv_billets = pd.read_csv(chemin_nouveau_fichier)
```

```
# Importation du modèle
modele = joblib.load('modele_RF.joblib')
```

```
# Appliquer la prédiction sur les données corrigées
y_pred = modele.predict(nv_billets_clean)
```

	id	Authenticite	Diagonale	Hauteur_gauche	Hauteur_droite	Marge_basse	Marge_haute	Longueur
0	A_1	0	171.76	104.01	103.54	5.21	3.30	111.42
1	A_2	0	171.87	104.17	104.13	6.00	3.31	112.09
2	A_3	0	172.00	104.58	104.29	4.99	3.39	111.57
3	A_4	1	172.49	104.55	104.34	4.44	3.03	113.20
4	A_5	1	171.65	103.63	103.56	3.77	3.16	113.33

```
Nombre total de billets : 5
Nombre de billets authentiques : 2
Nombre de billets contrefaits : 3
Pourcentage de billets authentiques : 40.00%
```

Merci pour votre attention