ONCFM

Présentée par Bouzouita Hayette - Data Analyst en mission pour *L'Organisation Nationale de lutte Contre le Faux-Monnayage (ONCFM)*

Détection de faux billets avec Python

Objectif : Mettre en place une modélisation qui serait capable d'identifier automatiquement les faux billets. *Et ce à partir simplement des caractéristiques géométriques d'un billet.*

6 caractéristiques géométriques d'un billet :

- diagonal : la diagonale du billet (en mm)
- height_left : la hauteur du billet (mesurée sur le côté gauche, en mm)
- height_right : la hauteur du billet (mesurée sur le côté droit, en mm)
- margin_low: la marge entre le bord inférieur du billet et l'image de celui-ci (en mm)
- margin_up: la marge entre le bord supérieur du billet et l'image de celui-ci (en mm)
- length : la longueur du billet (en mm)

```
billets = pd.read_csv('billets.csv', sep=";")
```

	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

	count	mean	std	min	25%	50%	75%	max
diagonal	1500.0	171.958440	0.305195	171.04	171.750	171.96	172.17	173.01
height_left	1500.0	104.029533	0.299462	103.14	103.820	104.04	104.23	104.88
height_right	1500.0	103.920307	0.325627	102.82	103.710	103.92	104.15	104.95
margin_low	1463.0	4.485967	0.663813	2.98	4.015	4.31	4.87	6.90
margin_up	1500.0	3.151473	0.231813	2.27	2.990	3.14	3.31	3.91
length	1500.0	112.678500	0.872730	109.49	112.030	112.96	113.34	114.44

0 doublons

```
billets.isna().sum()
```

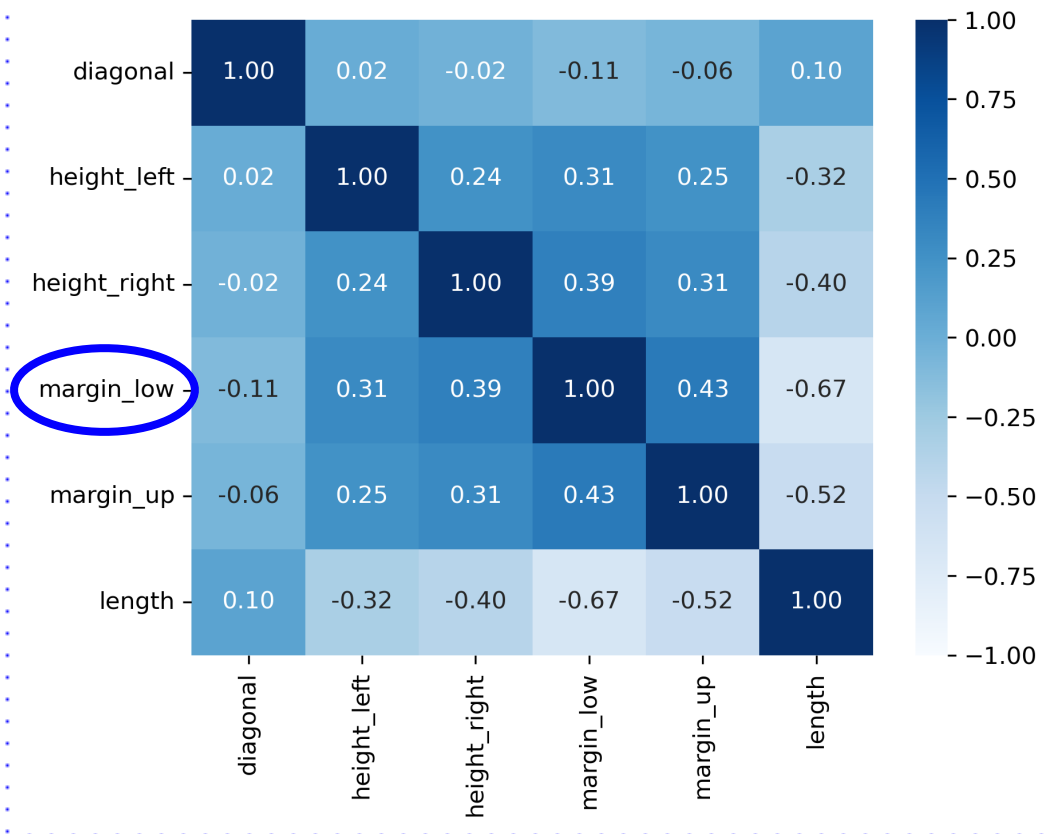
```
is_genuine      0
diagonal        0
height_left     0
height_right    0
margin_low     37
margin_up       0
length          0
dtype: int64
```

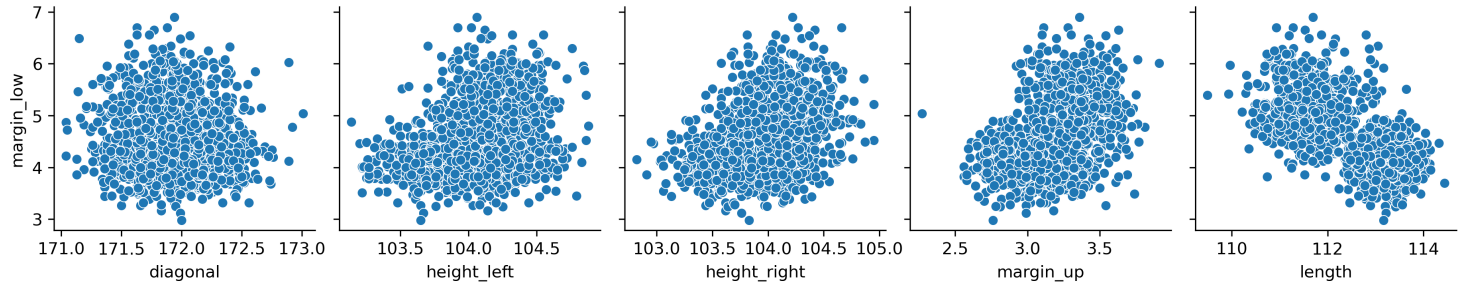
```
(billets["margin_low"].isna().sum() / len(billets))*100
```

```
2.466666666666667
```

Régression Linéaire

Méthode qui consiste à estimer la valeur manquante d'une variable continue à partir des relations observées avec d'autres variables.





	Coefficient
length	-0.461099
margin_up	0.331252

Intercept: 55.39568940242489

Score R^2 : 0.454277286393189

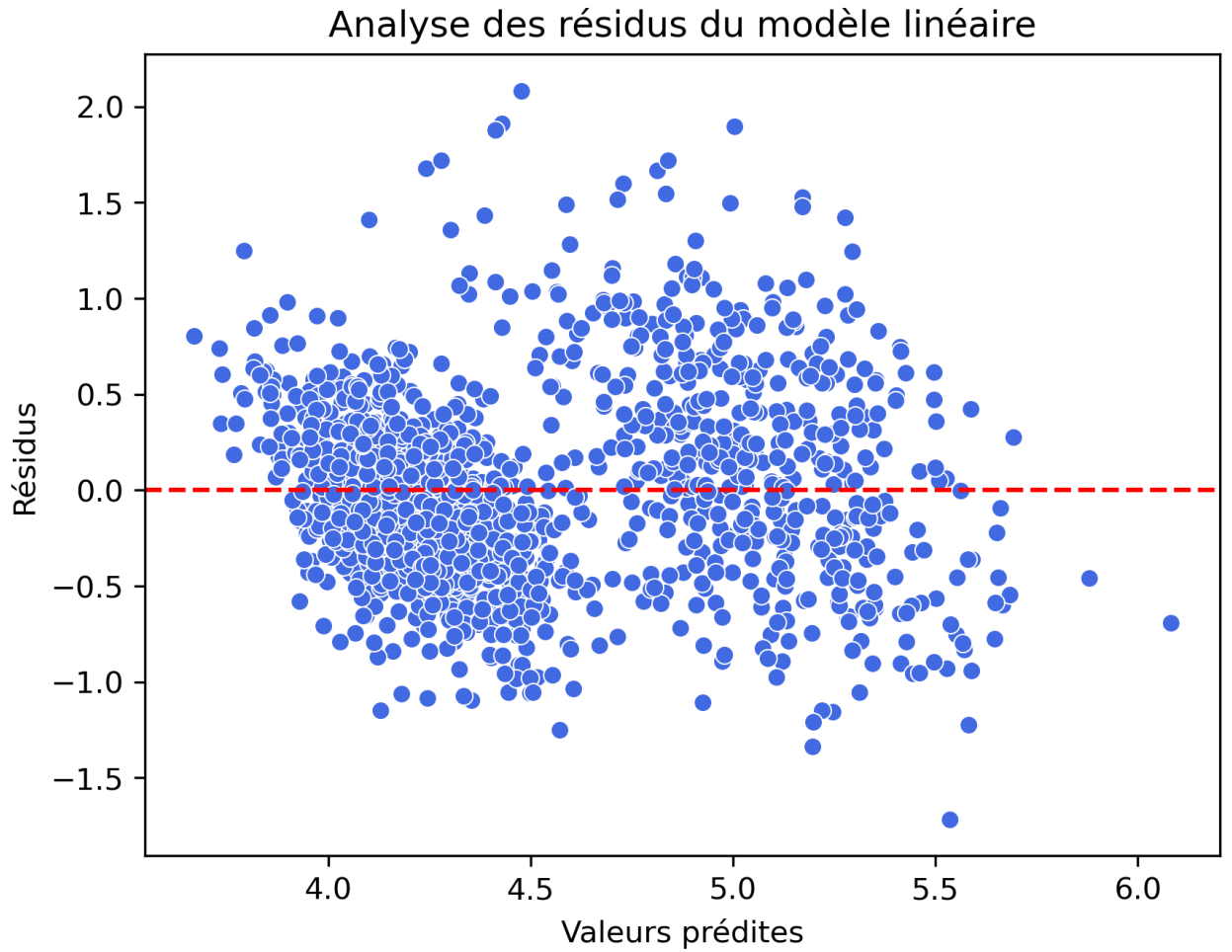
margin_low = 55.39 - 0.46*length + 0.33*margin_up

Hypothèse 1 de la Régression Linéaire

Linéarité

Méthode 1 :
Scatterplot entre la variable cible y et chaque variable explicative X

Méthode 2 :
Scatterplot des résidus vs valeurs prédites



Résidus i (erreur) = yi - $\hat{yi}$

Hypothèse 2

Indépendance des résidus

Méthode : Test de Durbin–Watson

```
Durbin-Watson : 1.8747486051663327
```

Interprétation :
Le test donne une valeur entre 0 et 4 :
DW ≈ 2 -> Aucune autocorrélation (erreurs indépendantes)
DW < 1.5 -> Autocorrélation positive (résidus liés entre eux)
DW > 2.5 -> Autocorrélation négative (alternance de signes dans les résidus)

Hypothèse 3

Homoscédasticité

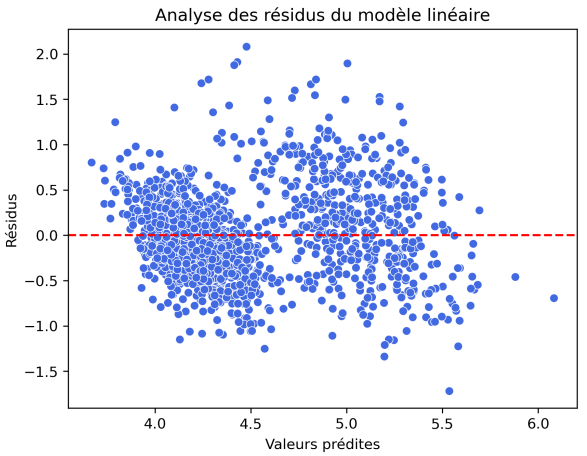
Méthode 1:
Scatterplot résidus vs valeurs prédites

Méthode 2: Test de Breusch–Pagan

```
{'Statistique LM': 69.18185711374537, 'p-value LM': 9.49184426089401e-16,

'Statistique F': 36.23338952128672, 'p-value F': 4.3863636385048543e-16}
```

Interprétation :
H0 : La variance des résidus est constante -> Homoscédasticité
H1 : La variance des résidus n'est pas constante -> Hétéroscédasticité
si p_value < 0.5 on rejette H0



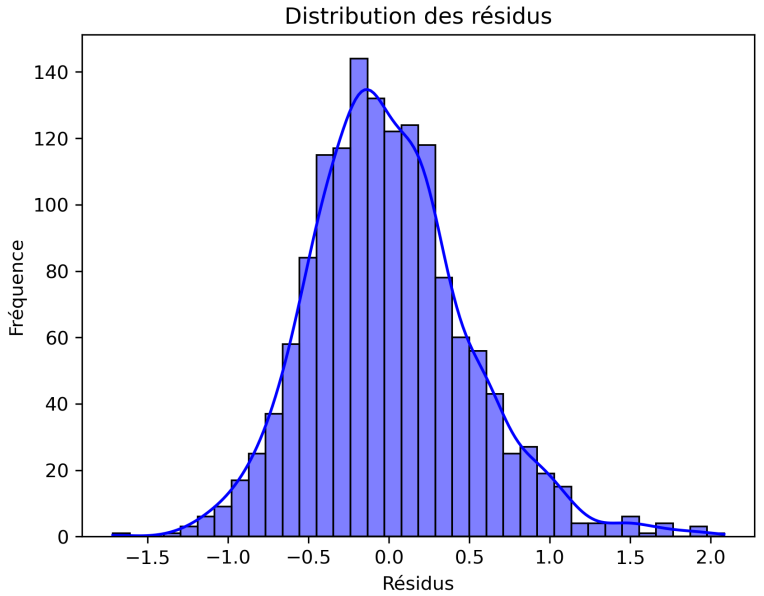
Solution : Transformation logarithmique de la variable cible afin de stabiliser la variance des résidus.

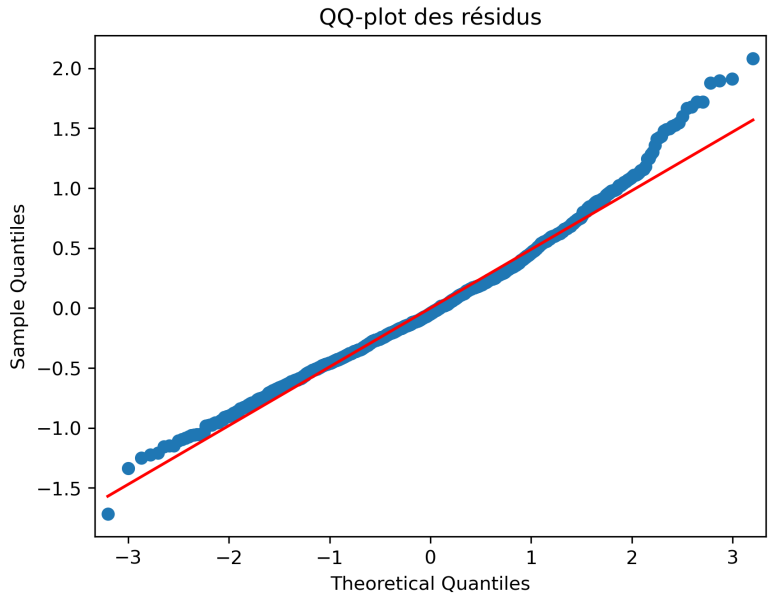
Hypothèse 4

Normalité
des résidus

Méthode 1:
Histogramme, QQ-plot

Méthode 2:
Test de Shapiro–Wilk

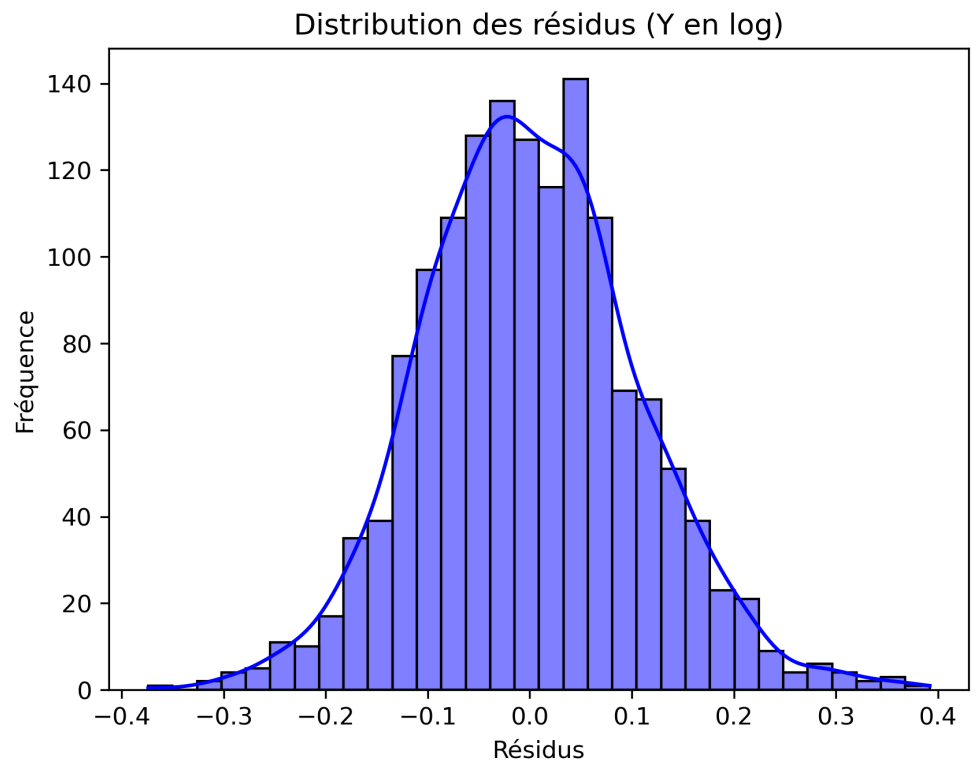


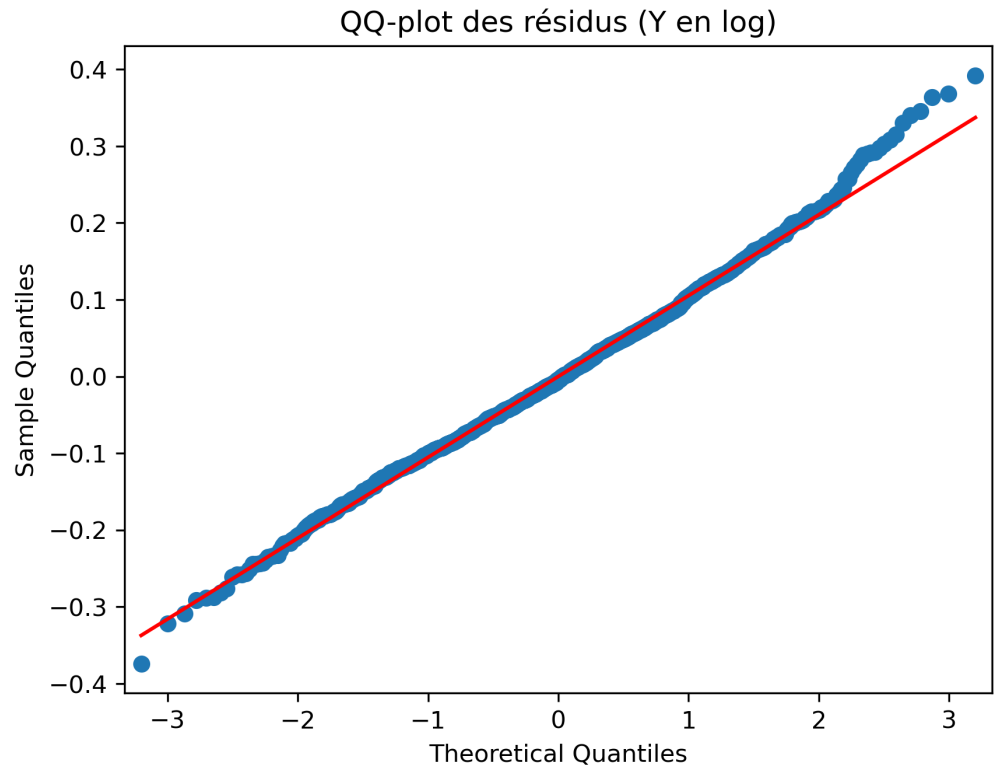



```
Statistique de Shapiro-Wilk : 0.9835276366015211
p-value : 7.061289084592733e-12
```

Solution : Transformation logarithmique de la variable cible

Interprétation :
H0 : Normalité des résidus
H1 : Non - normalité des résidus
si p_value < 0.5 on rejette H0






```
Statistique de Shapiro-Wilk : 0.9964320268816002
p-value : 0.0018019276108824324
```

```
{'Statistique LM': 28.423456686187922, 'p-value LM': 6.728601410461491e-07, 'Statistique F': 14.463587514812993, 'p-value F': 6.02506162387871e-07}
```

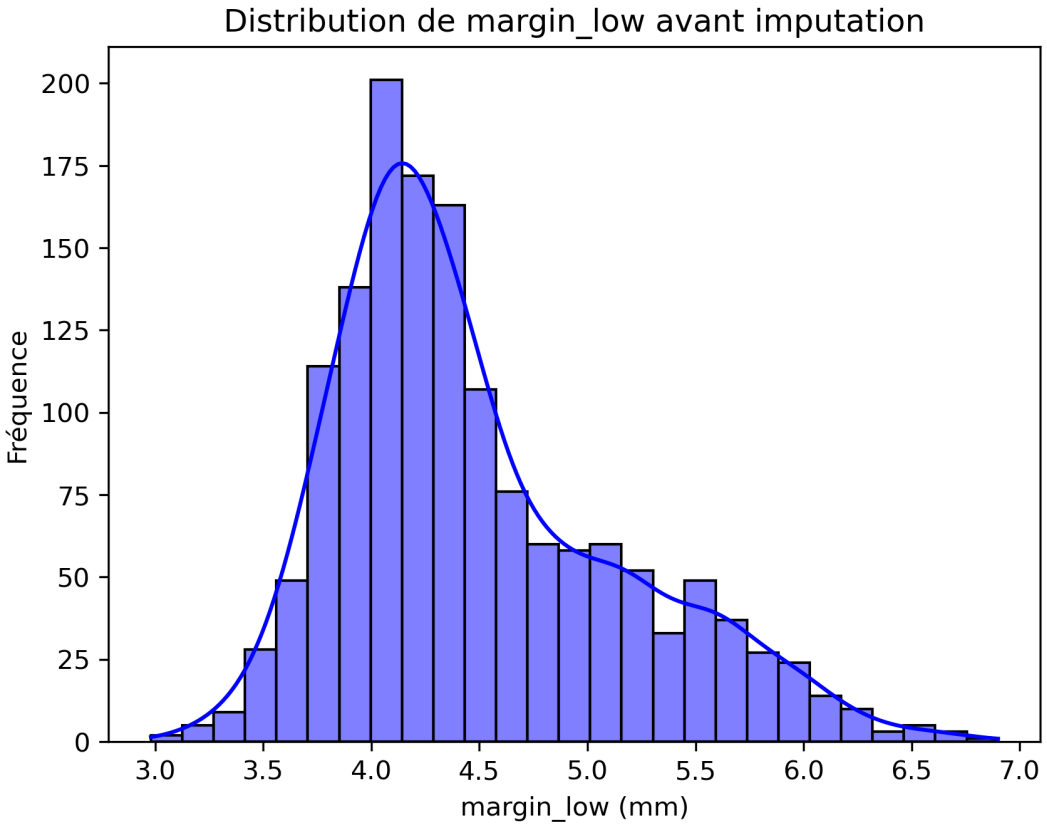
Hypothèse 5

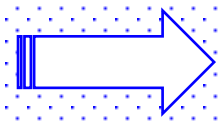
Absence de multi colinéarité

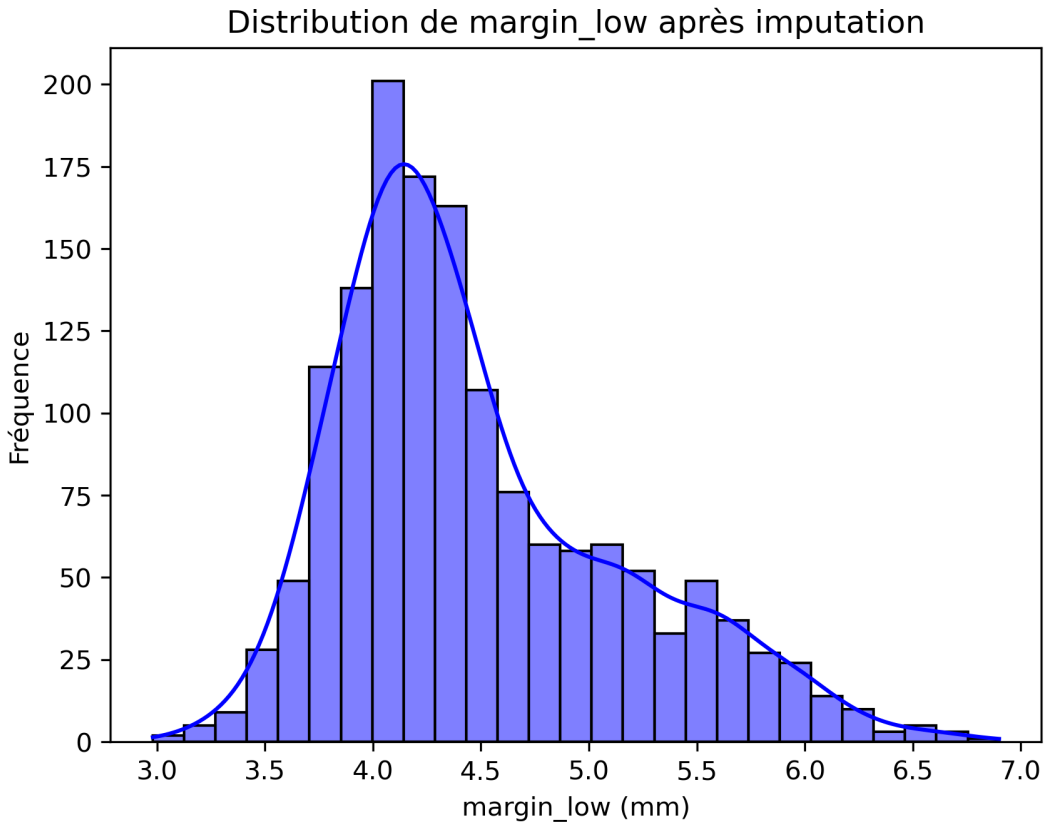
Méthode : Variance Inflation Factor (VIF)

1.3728452237329334,

Interprétation :
VIF < 5 → pas de souci
VIF entre 5 et 10 → multicolinéarité modérée
VIF > 10 → problème sérieux

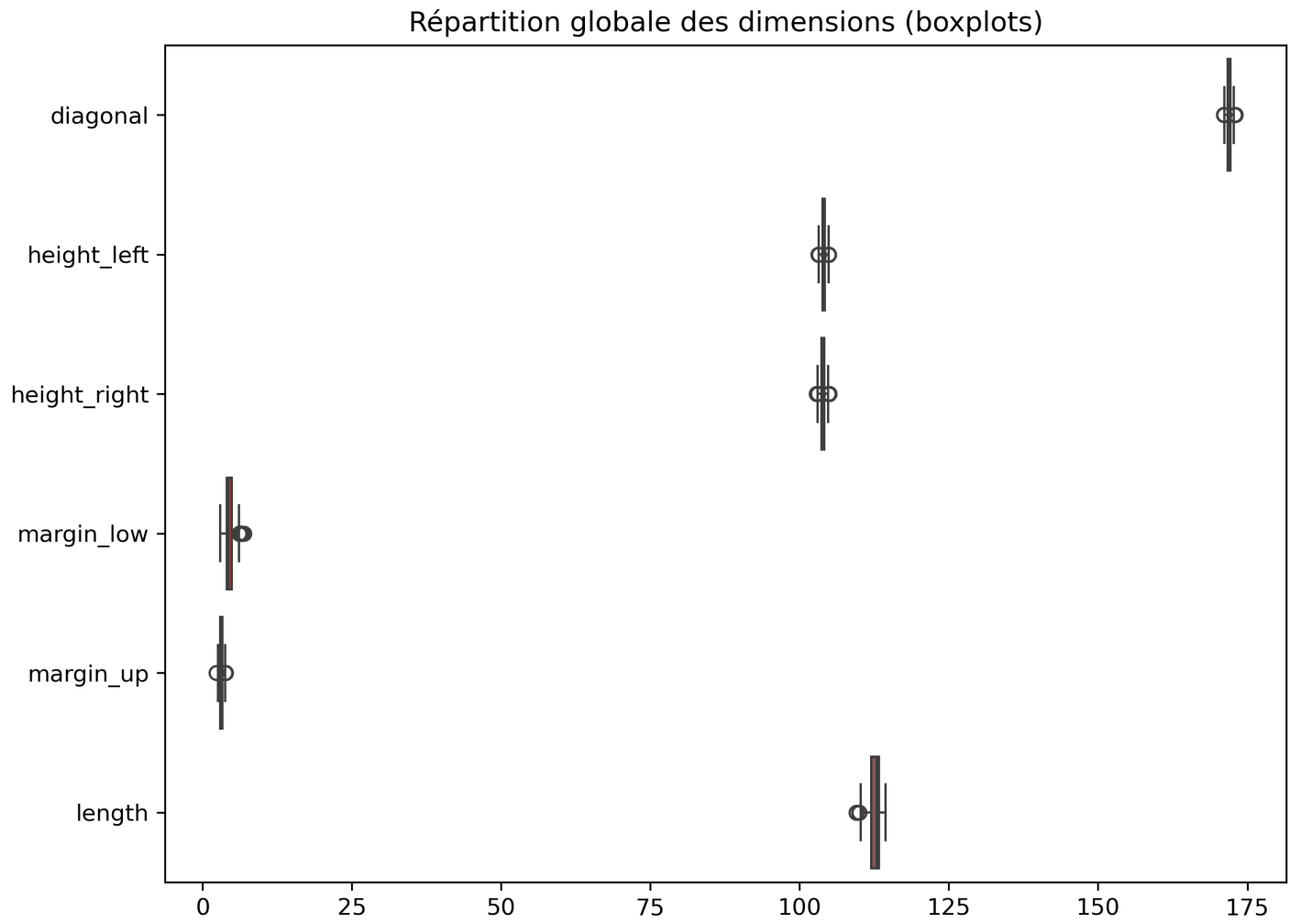


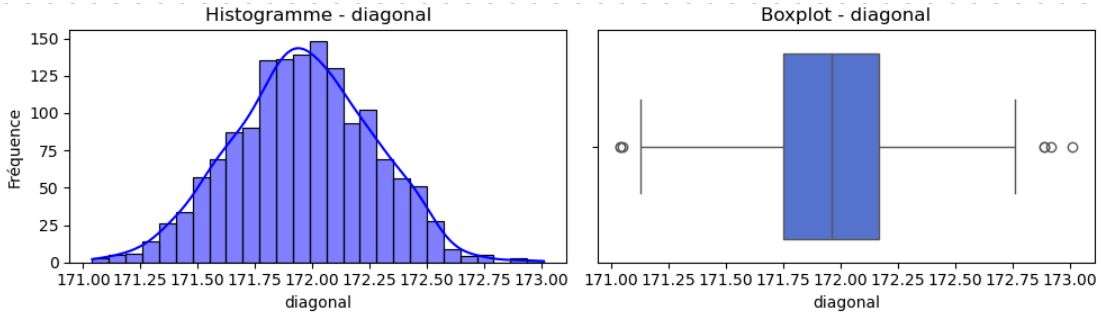




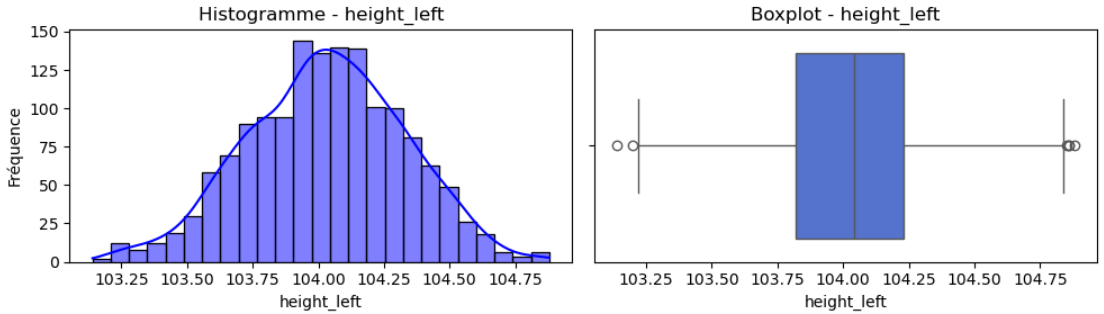

```
Avant  | mean, median, std : 4.485967190704033 4.31 0.6638126241773387
Après  | mean, median, std : 4.483173433424014 4.31 0.6590884493182334
```

Analyse Descriptive et Exploratoire

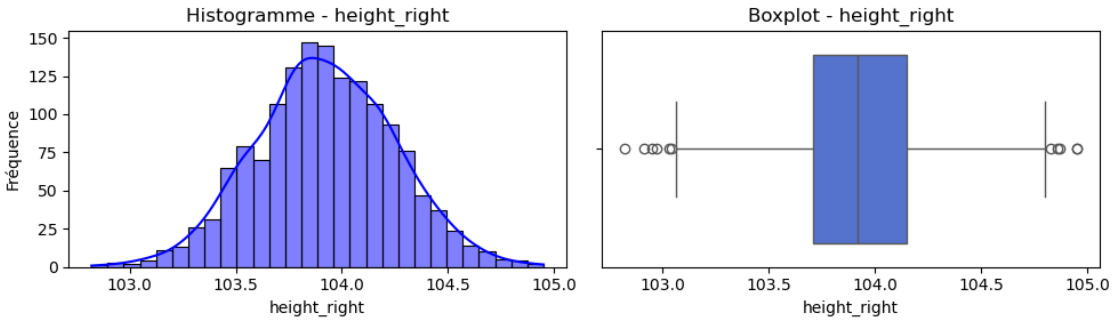



Histogramme - diagonal
Fréquence
diagonal

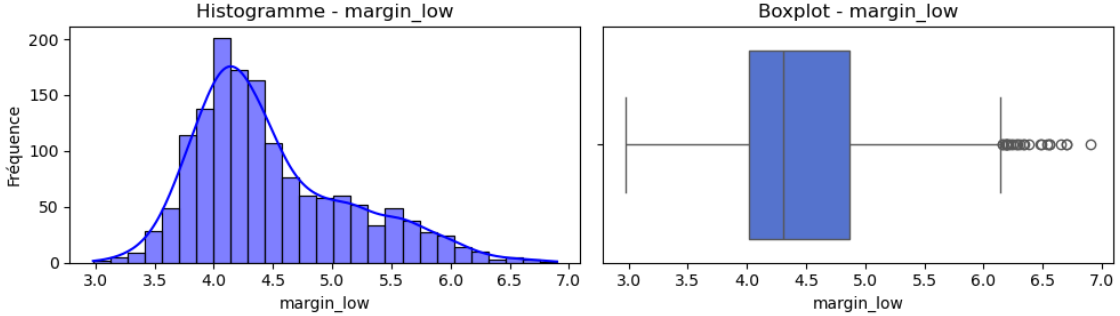
Boxplot - diagonal
diagonal


Histogramme - height_left
Fréquence
height_left

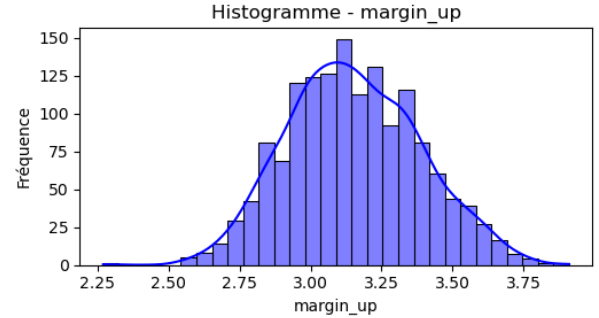
Boxplot - height_left
height_left

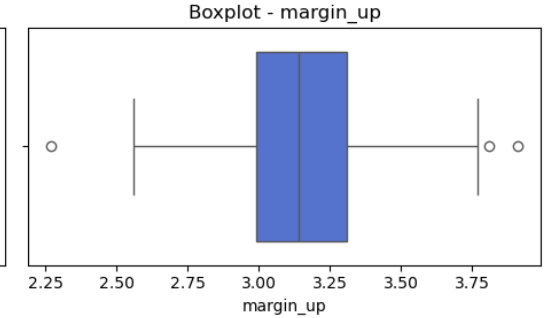

Histogramme - height_right
Fréquence
height_right

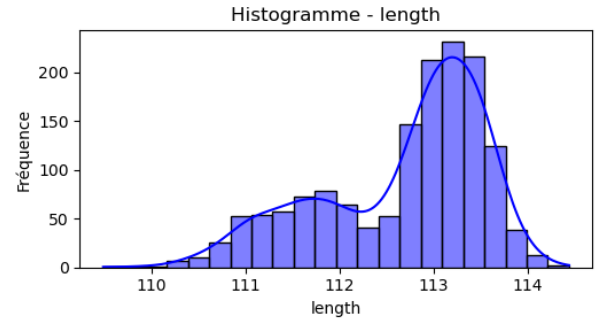
Boxplot - height_right
height_right

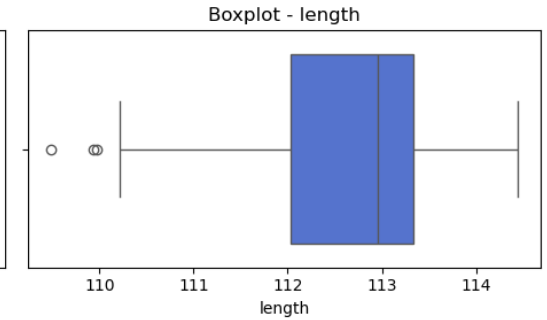

Histogramme - margin_low
Fréquence
margin_low

Boxplot - margin_low
margin_low

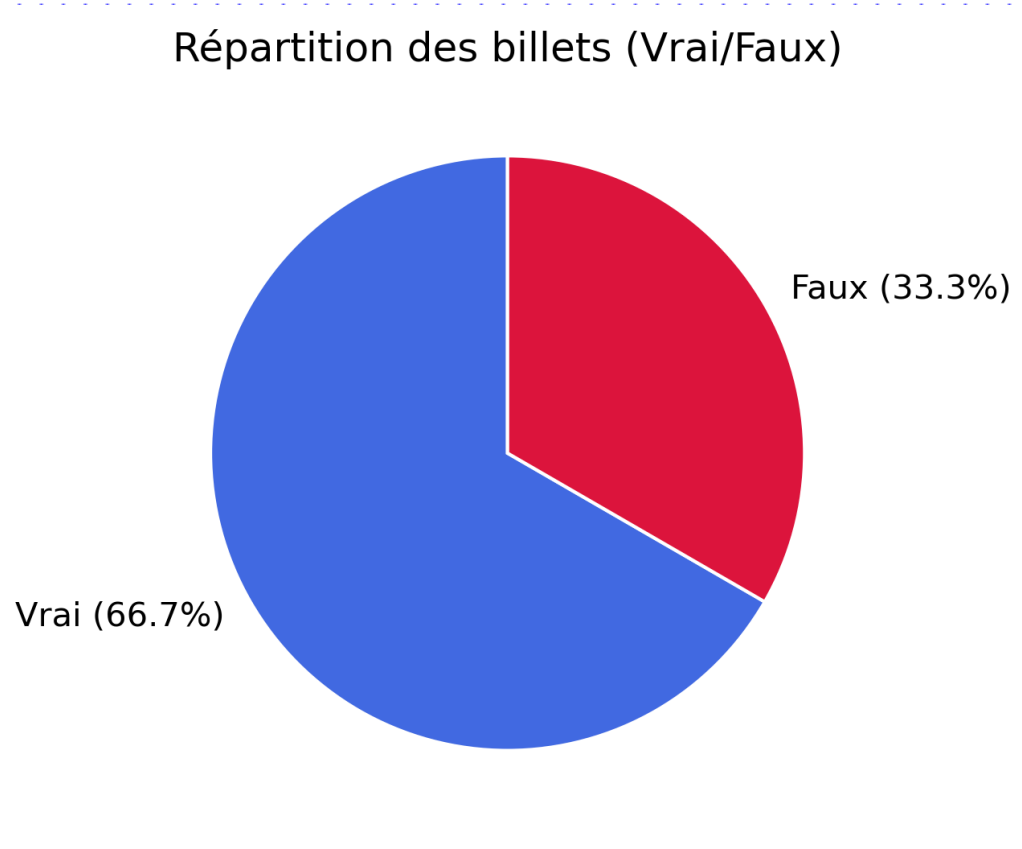

Histogramme - margin_up
Fréquence
margin_up

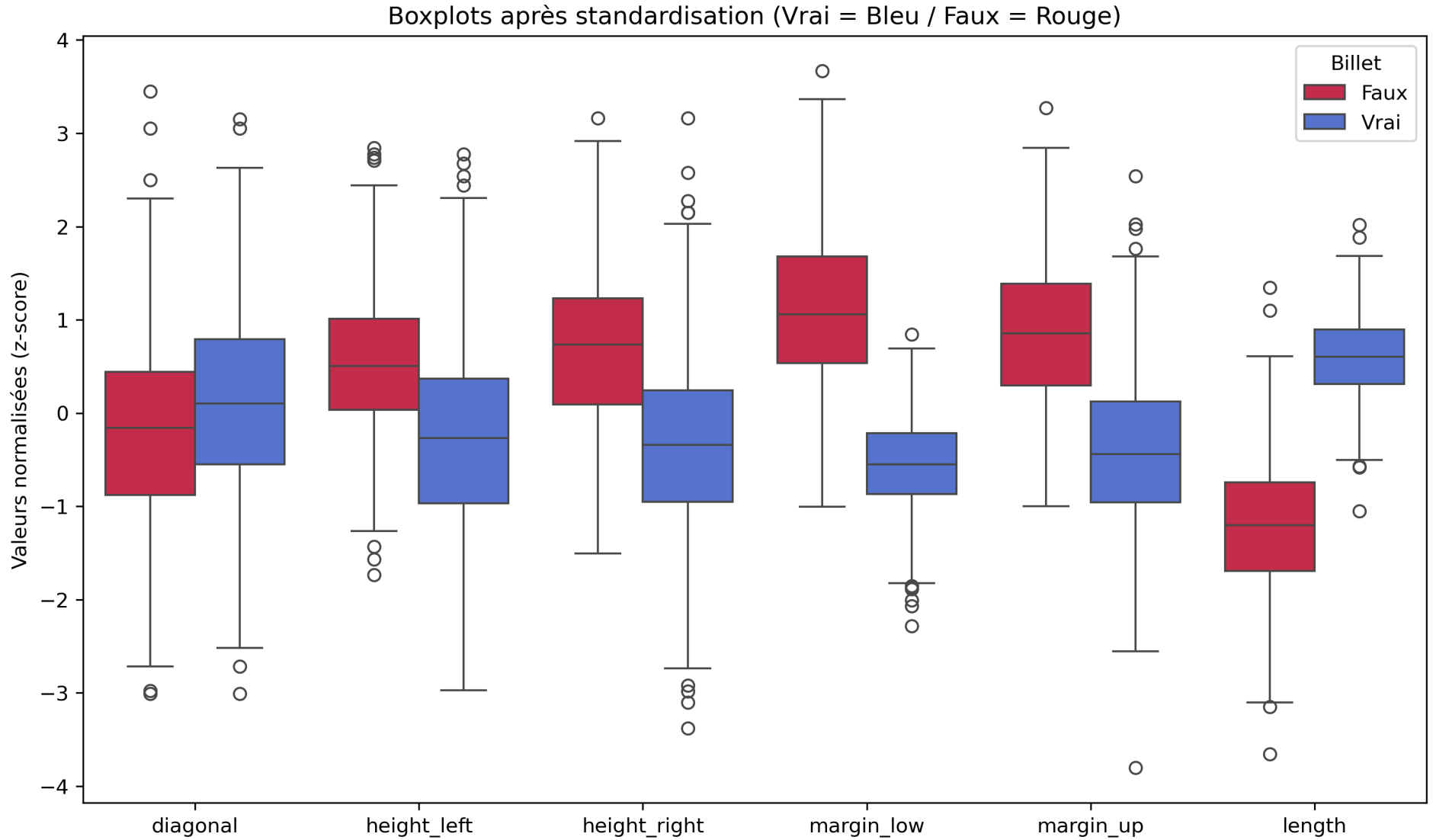

Boxplot - margin_up
margin_up

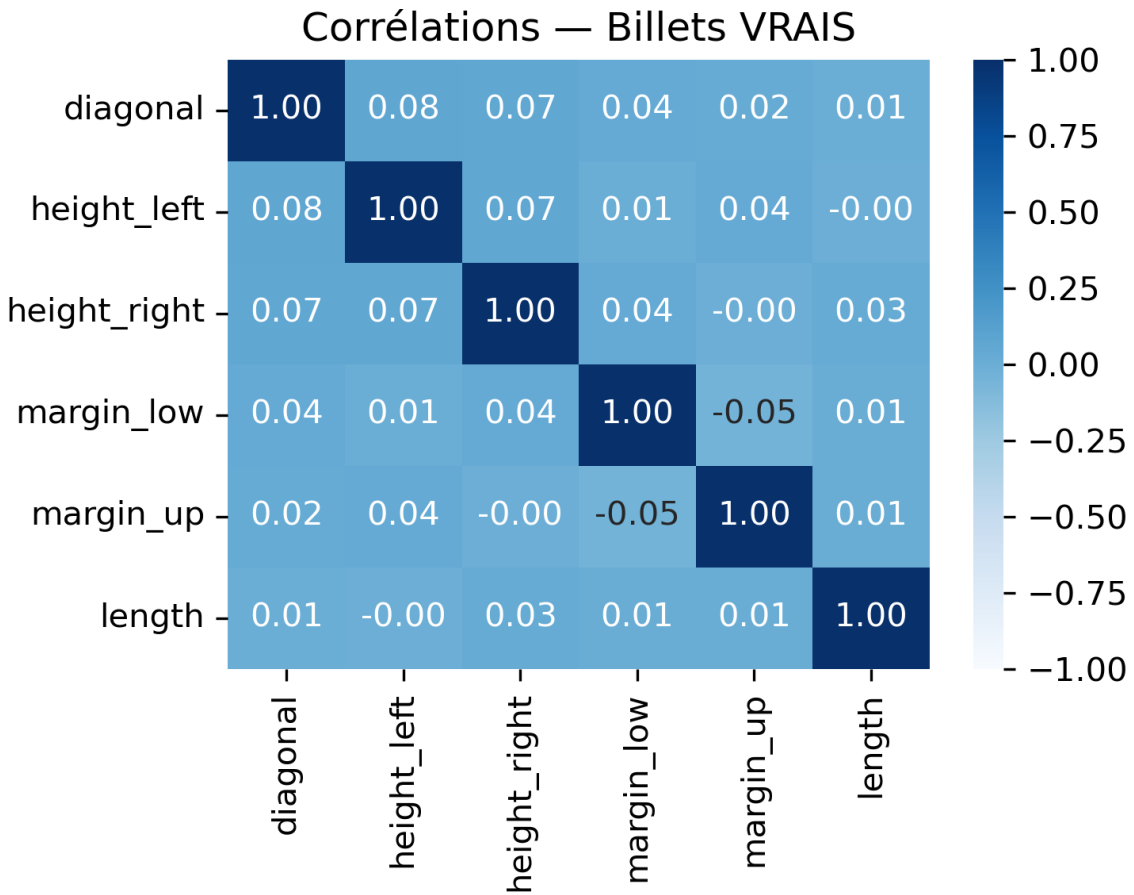

Histogramme - length
Fréquence
length

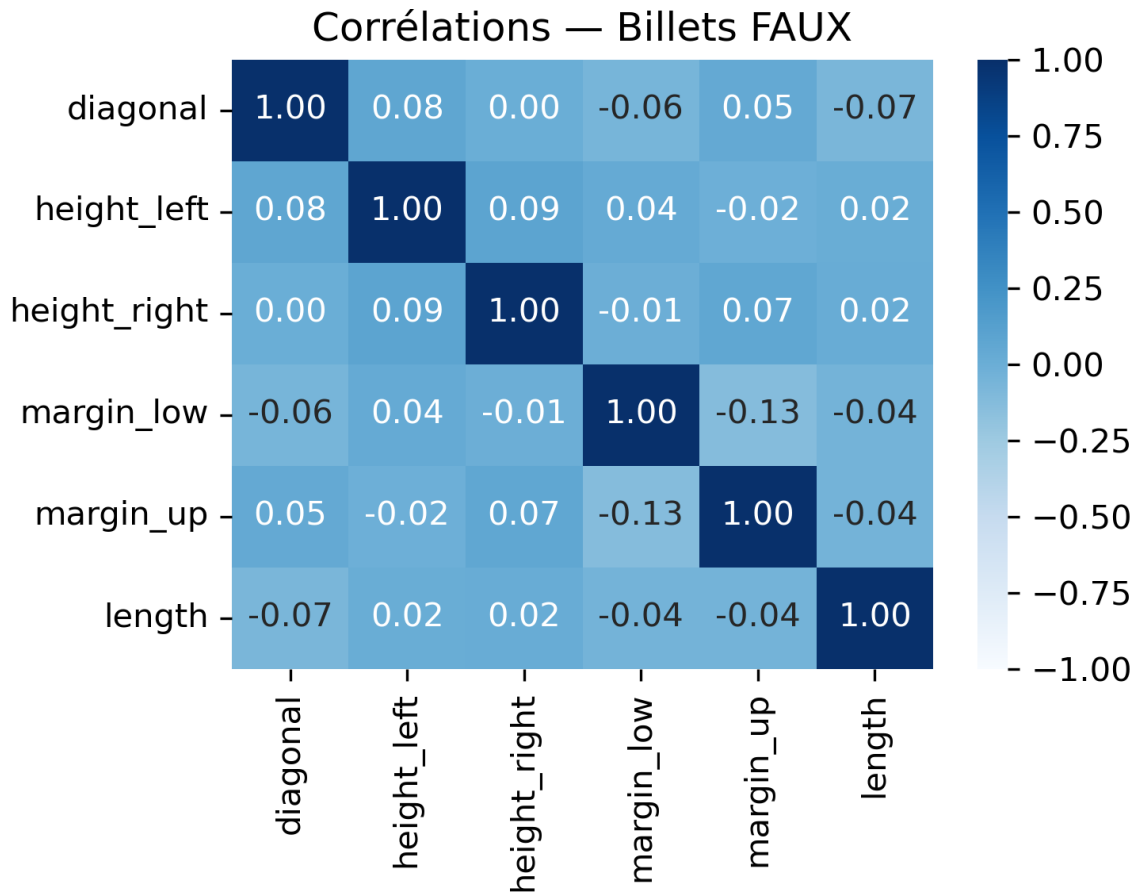

Boxplot - length
length

is_genuine	Effectif	Pourcentage (%)
Vrai	1000	66.7
Faux	500	33.3



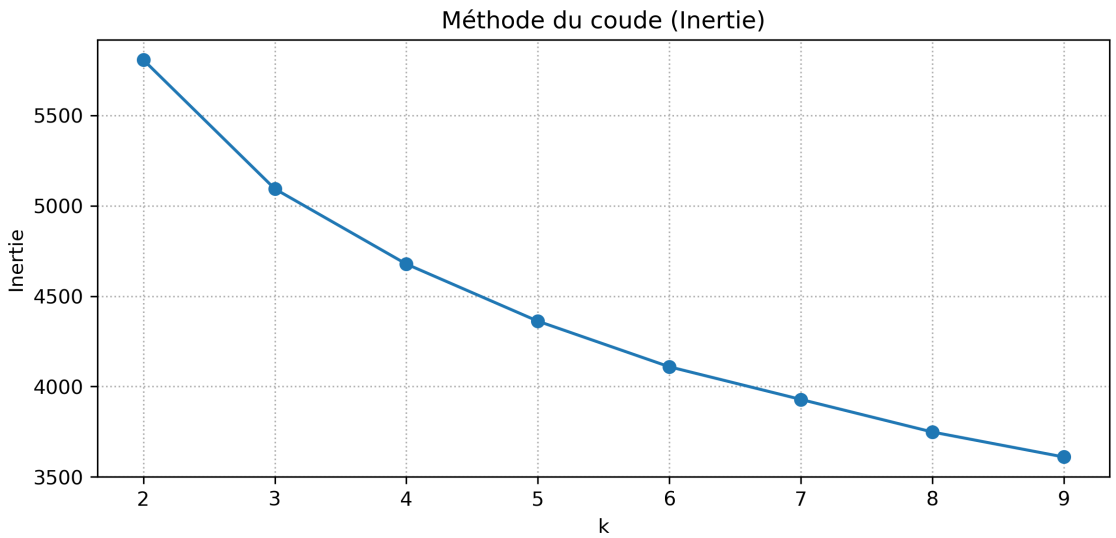

Boxplots après standardisation (Vrai = Bleu / Faux = Rouge)
Billet
Faux
Vrai
Valeurs normalisées (z-score)
4
3
2
1
0
−1
−2
−3
−4
diagonal
height_left
height_right
margin_low
margin_up
length

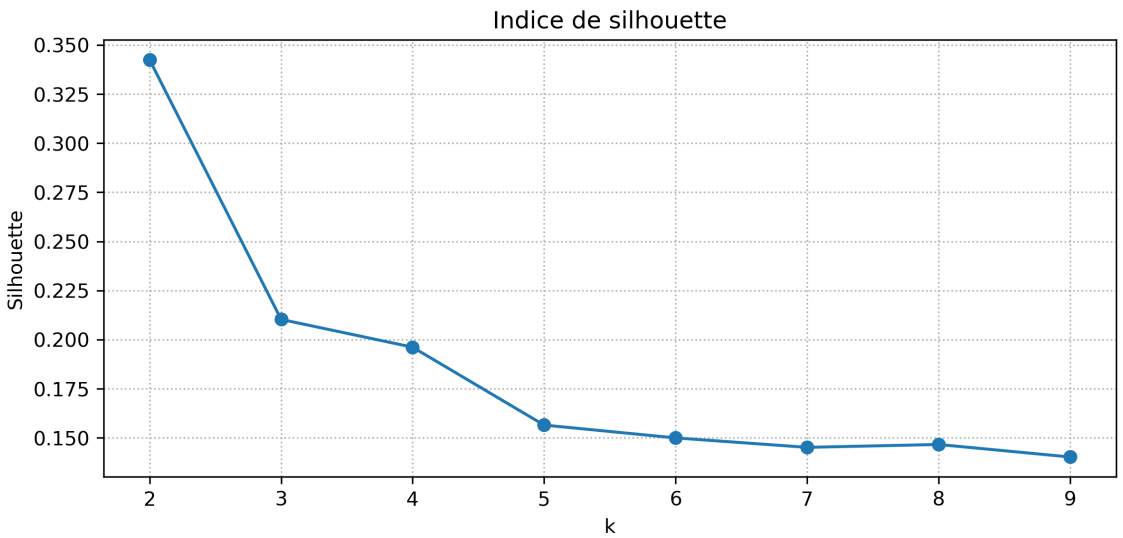




Corrélations légèrement plus fortes pour les faux billets, ce qui laisse penser que certains défauts de fabrication sur une dimension peuvent s’accompagner d’erreurs systématiques sur d’autres mesures.

Approche Non supervisée : K-Means

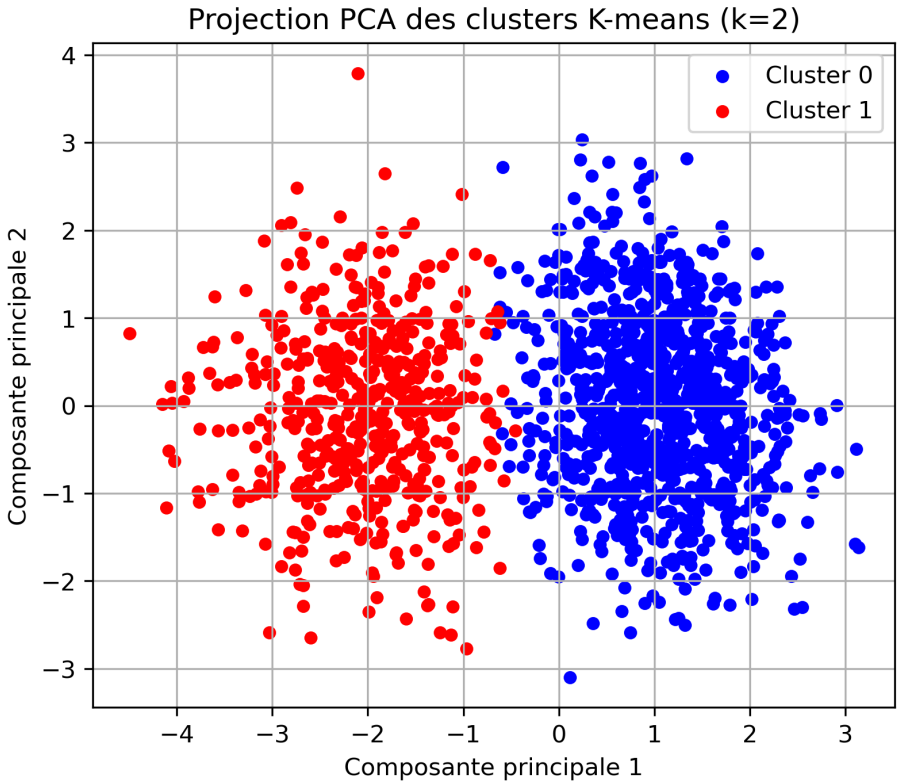


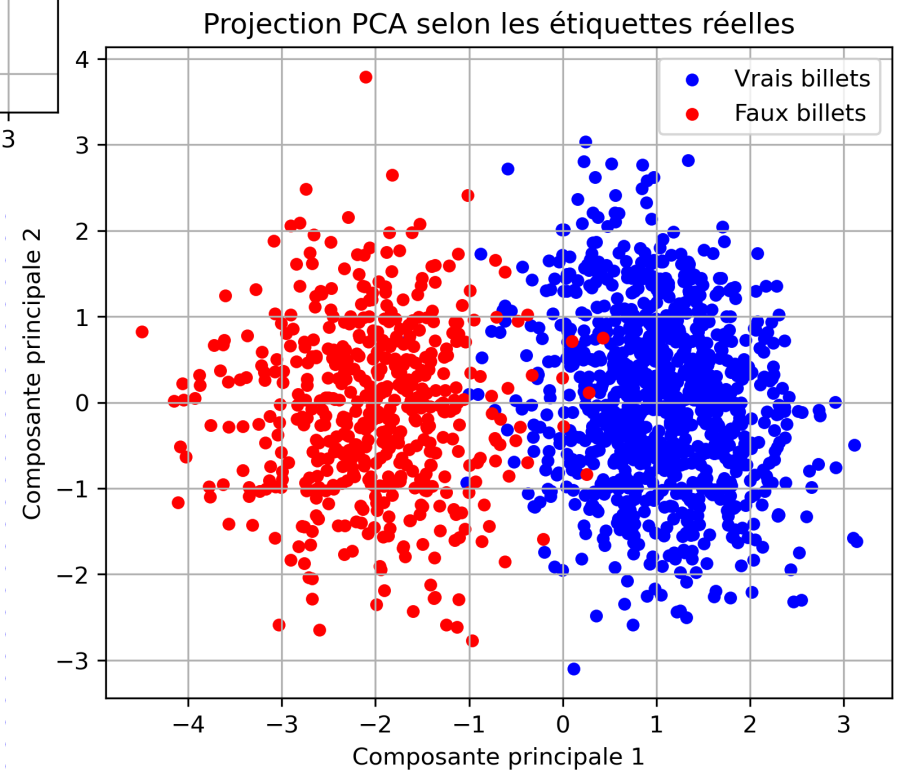


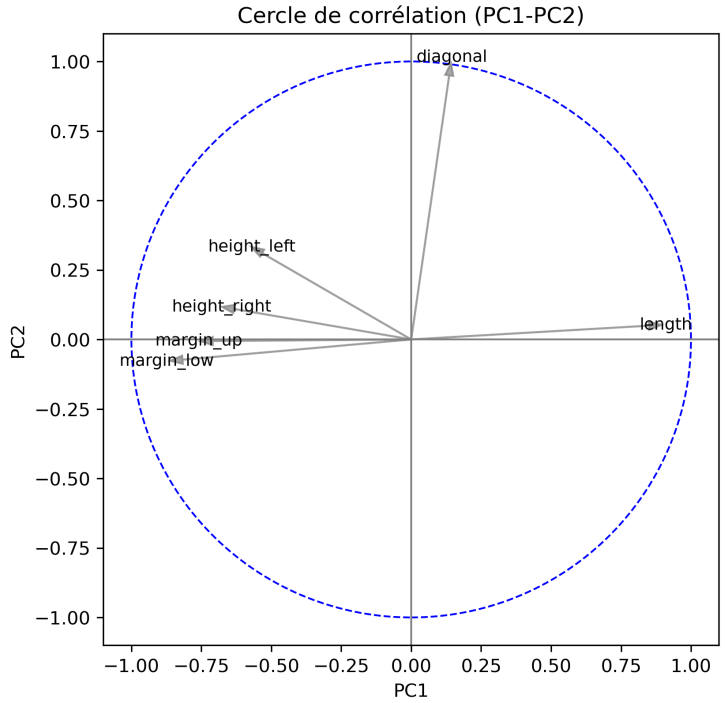

```
Cluster_kmeans
0    1003
1     497
Name: count, dtype: int64
```

Cluster	0	1
Vrai/Faux		
False	13	487
True	990	10

```
Accuracy (ajustée au bon sens) : 0.985
ARI (Adjusted Rand Index) : 0.939
NMI (Normalized Mutual Information) : 0.877
```


Projection PCA des clusters K-means (k=2)
Cluster 0
Cluster 1
Composante principale 2
Composante principale 1

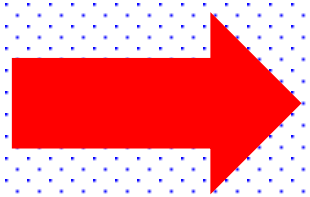

Projection PCA selon les étiquettes réelles
Vrais billets
Faux billets
Composante principale 2
Composante principale 1


Cercle de corrélation (PC1-PC2)
diagonal
height_left
height_right
margin_up
margin_low
length
PC2
PC1

	diagonal	height_left	height_right	margin_low	margin_up	length
0	0.096124	-0.281762	-0.352065	-0.552229	-0.428505	0.594584
1	-0.193988	0.568627	0.710506	1.114458	0.864769	-1.199935

Cluster_kmeans	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	0.99	171.99	103.95	103.81	4.12	3.05	113.20
1	0.02	171.90	104.20	104.15	5.22	3.35	111.63

is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
False	171.90116	104.19034	104.14362	5.21270	3.35016	111.63064
True	171.98708	103.94913	103.80865	4.11841	3.05213	113.20243



K-means arrive à séparer les Faux billets des Vrais. Mais il fait encore des erreurs et présente beaucoup de limites !

Approche supervisée : Régression Logistique

```python
#variables explicatives et variable cible
feature_cols = [ 'diagonal', 'height_left', 'height_right', 'margin_low', 'margin_up','length']
target_col = 'is_genuine'    # 0 = vrai, 1 = faux

X = df_Rlogistique[feature_cols]
y = df_Rlogistique[target_col]
```

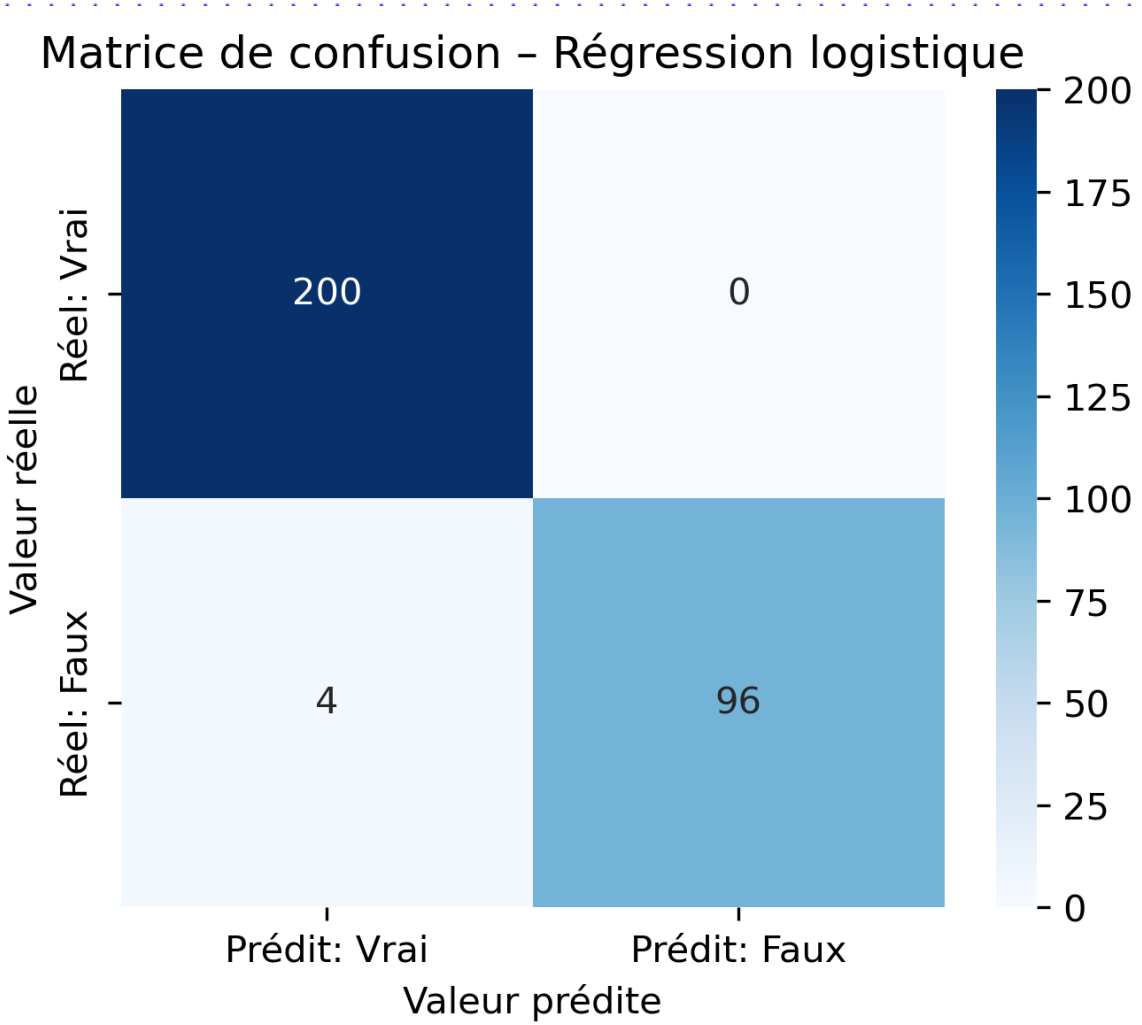
```
Taille jeu d'entraînement : (1200, 6)
Taille jeu de test : (300, 6)
```

Création du pipeline :

Le pipeline enchaîne automatiquement plusieurs étapes :

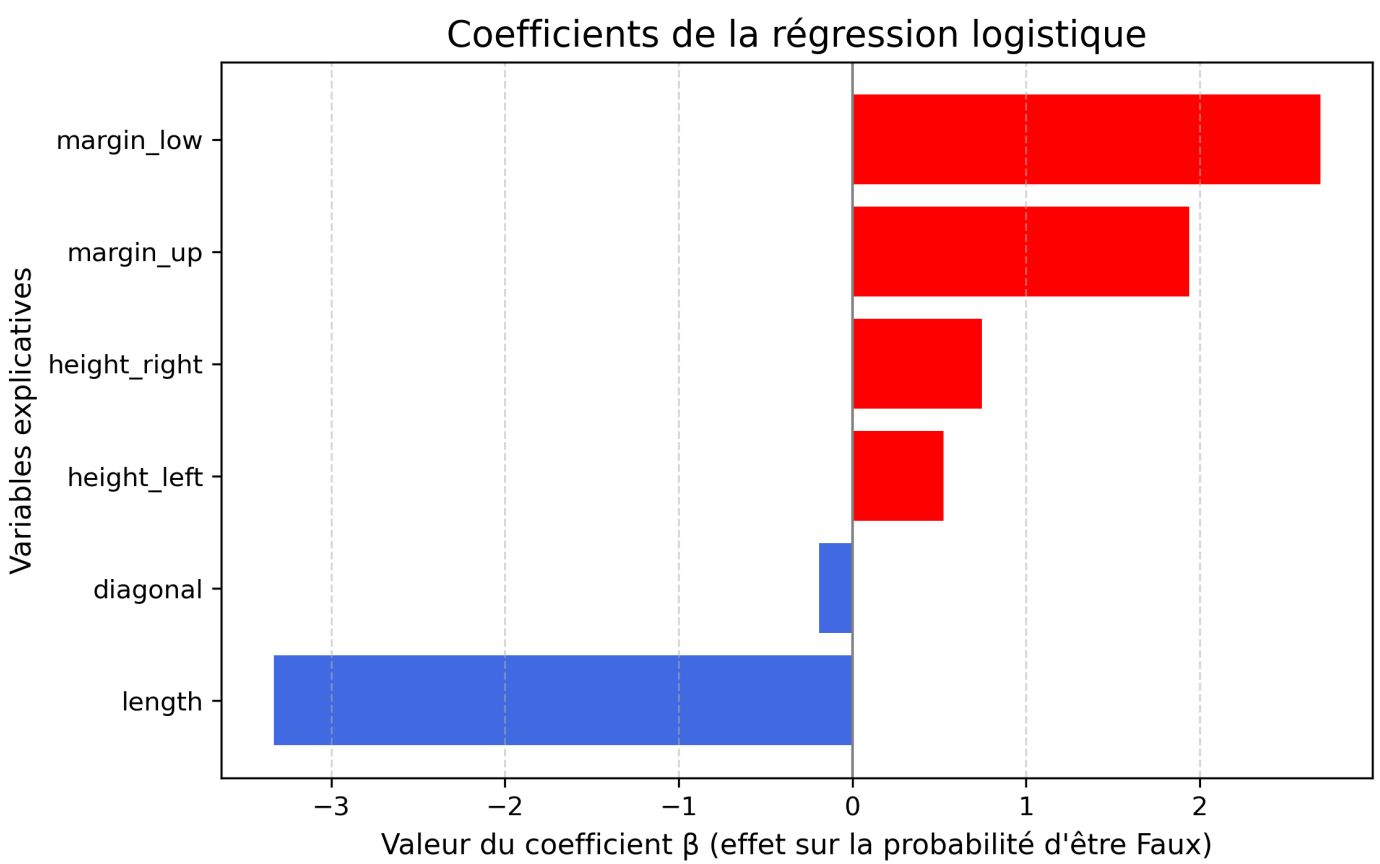
1. Imputation : remplace les valeurs manquantes (sécurité, même s'il n'y en a pas actuellement)
2. Standardisation : met toutes les variables sur la même échelle
3. Régression logistique : apprentissage du modèle

=> Cela garantit une exécution propre, cohérente et reproductible.



Taux de bonne classification (accuracy) : 0.987

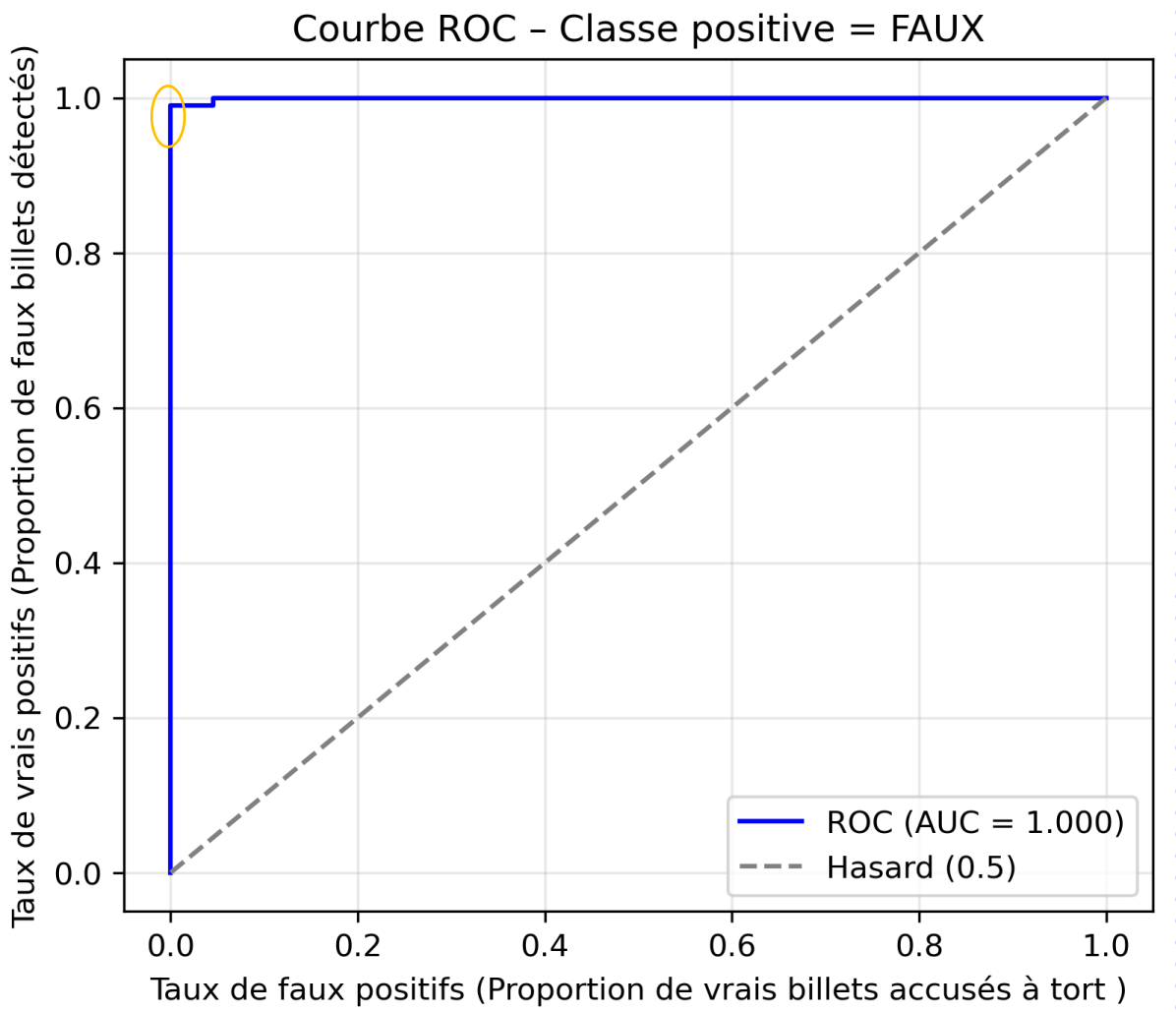
	coef_beta	importance_abs	odds_ratio
length	-3.331755	3.331755	0.035730
margin_low	2.692955	2.692955	14.775272
margin_up	1.937791	1.937791	6.943399
height_right	0.744308	0.744308	2.104985
height_left	0.525643	0.525643	1.691546
diagonal	-0.192549	0.192549	0.824854



Interprétation des odds ratios :

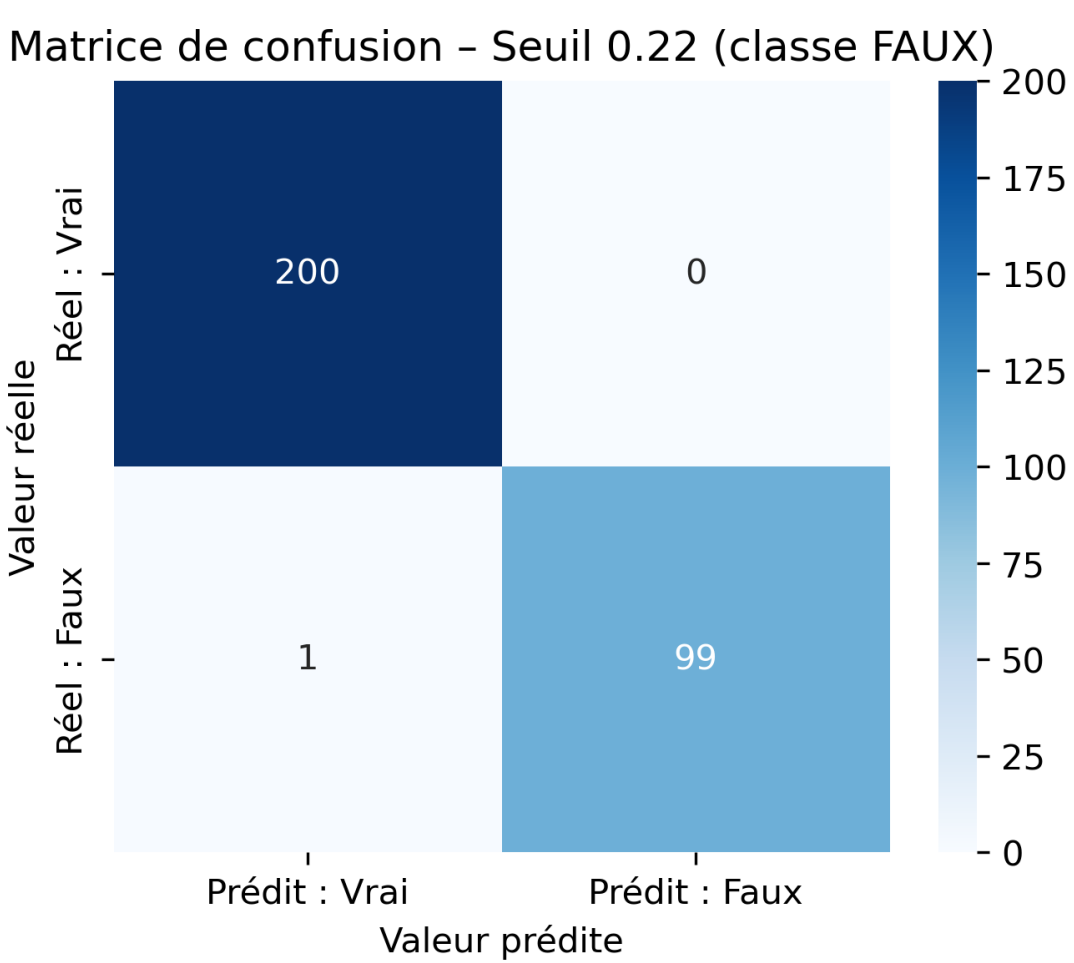
- length (OR = 0.036) → quand la longueur augmente d'un écart-type, les chances d'être un billet faux sont divisées par ≈ 33 (1 / 0.036) !
- margin_low (OR = 14.78) → quand la marge basse augmente, les chances d'être faux sont multipliées par ≈ 15 !
- margin_up (OR = 6.94) → multiplie les chances d'être faux par ≈ 7.

```
AUC (classe positive = FAUX) : 1.0
```




```
Seuil : inf --> FPR=0.00, TPR=0.00
Seuil : 1.00 --> FPR=0.00, TPR=0.01
Seuil : 0.23 --> FPR=0.00, TPR=0.99
Seuil : 0.08 --> FPR=0.04, TPR=0.99
Seuil : 0.08 --> FPR=0.04, TPR=1.00
Seuil : 0.00 --> FPR=1.00, TPR=1.00
```

```
Seuil optimal (F1) : 0.22, F1-score : 0.995
```



Taux de bonne classification (accuracy) : 0.997

Test de robustesse :

Afin de vérifier la robustesse du modèle,
10 séparations aléatoires du jeu de données
(80 % apprentissage / 20 % test) ont été réalisées.

À chaque itération, le modèle de régression logistique a été réentraîné et évalué sur un jeu de test différent, afin de s’assurer que les performances ne dépendent pas du hasard de la découpe.


```
Accuracy moyenne : 0.988
Écart-type       : 0.0052
```

Excellente stabilité du modèle

Application de l'Algorithme de détection de faux billets

Sauvegarde du modèle final + configuration (seuil 0.22)

Fonction pour prédire un fichier csv

Application de l'algorithme avec un nouveau jeu de données

```
Sauvegardé: artifacts/model.joblib + artifacts/config.json
```

```python
import os, json, joblib, sklearn
os.makedirs("artifacts", exist_ok=True)

joblib.dump(pipeline_logistic, "artifacts/model.joblib")

config = {
    "threshold": 0.22, #seuil à 0.22
    "expected_features": expected_features,
    "sklearn_version": sklearn.__version__,
    "random_state": 0,
    "notes": "pipeline: imputer(median) pour NaN -> scaler -> logistic_regression(class_weight=balanced)"
}
with open("artifacts/config.json", "w", encoding="utf-8") as f:
    json.dump(config, f, ensure_ascii=False, indent=2)

print("Sauvegardé: artifacts/model.joblib + artifacts/config.json")
```

```
def predict_file(path_csv, id_col=None)
```

```python
def predict_file(path_csv, id_col=None):
    # Charger artefacts
    pipe = joblib.load("artifacts/model.joblib")
    with open("artifacts/config.json", "r", encoding="utf-8") as f:
        cfg = json.load(f)

    thr = cfg["threshold"]
    expected = cfg["expected_features"]



    # Lire données
    df = pd.read_csv(path_csv)

    # Vérifs de schéma
    missing = [c for c in expected if c not in df.columns]
    extra   = [c for c in df.columns if c not in expected and c != id_col]
    if missing:
        print(f"Attention! Colonnes manquantes: {missing} (remplies par NaN → imputées par la pipeline)")
    if extra:
        print(f"Infos! Colonnes ignorées: {extra}")

    # Aligner l'ordre des colonnes attendues
    X = df.reindex(columns=expected)



    # Probas + seuil
    proba = pipe.predict_proba(X)[:, 1]
    label = (proba >= thr).astype(int)

    out = pd.DataFrame({"proba_pos": proba, f"label_{thr}": label})
    if id_col and id_col in df.columns:
        out.insert(0, id_col, df[id_col].values)
    return out
```

	id	proba_pos	label_0.22	prediction_text
0	A_1	0.998427	1	faux_billet
1	A_2	0.999841	1	faux_billet
2	A_3	0.999753	1	faux_billet
3	A_4	0.034642	0	vrai_billet
4	A_5	0.000244	0	vrai_billet

```
Fini. Résultats dans outputs/predictions_production.csv
```

```
Billets dans la zone d'incertitude :
```

id	proba_pos	label_0.22	prediction_text

```
Fichier Annexe. Résultats dans outputs/billets_zone_incertaine.csv
```