DETECTION DE FAUX BILLETS

PROJET 10 - DATA ANALYST

Arnaud CAILLE

ONCFM

1 .NETTOYAGE DES DONNEES

ONCFM

prévisualisation

```
bill_data.head()
```
✓ 0.0s

	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

1

Apperçu du dataset

2

Infos du dataset

```
bill_data.info()
```
✓ 0.0s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   is_genuine    1500 non-null   bool
 1   diagonal      1500 non-null   float64
 2   height_left   1500 non-null   float64
 3   height_right  1500 non-null   float64
 4   margin_low    1463 non-null   float64
 5   margin_up     1500 non-null   float64
 6   length        1500 non-null   float64
dtypes: bool(1), float64(6)
memory usage: 71.9 KB
```

3

Statistiques descriptives du dataset

```
descriptive_stats  = bill_data.describe().round(2)
descriptive_stats
```
✓ 0.0s

	diagonal	height_left	height_right	margin_low	margin_up	length
count	1500.00	1500.00	1500.00	1463.00	1500.00	1500.00
mean	171.96	104.03	103.92	4.49	3.15	112.68
std	0.31	0.30	0.33	0.66	0.23	0.87
min	171.04	103.14	102.82	2.98	2.27	109.49
25%	171.75	103.82	103.71	4.01	2.99	112.03
50%	171.96	104.04	103.92	4.31	3.14	112.96
75%	172.17	104.23	104.15	4.87	3.31	113.34
max	173.01	104.88	104.95	6.90	3.91	114.44

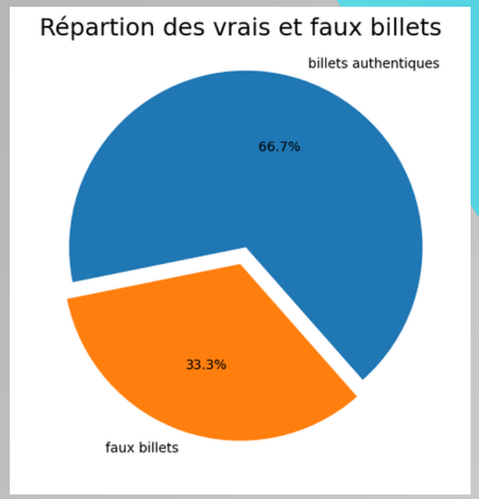
A la lecture des informations sur le dataset, on peut observer qu'il y a des valeurs manquantes pour la colonne "margin_low".

ONCFM

Visualisation graphique

1

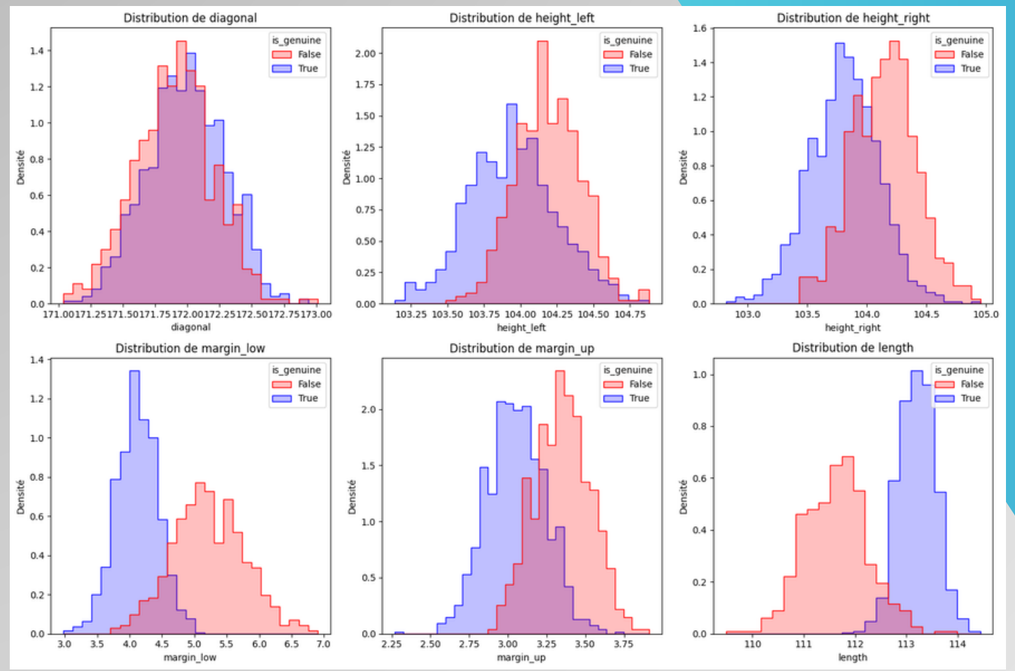
Répartition par authenticité



la répartition des billets du dataset fourni est d'un tiers pour les faux et de deux tiers pour les vrais

2

Distribution par caractéristique



- On remarque une tendance de chevauchement des faux et vrais billets, pour ce qui est de la diagonale.

- A l''inverse, on peut obeserver une nette dissociation pour ce qui est de la marge inférieure et encore plus marquée, concernant la longueur.

ONCFM

2 . IMPUTATION DES VALEURS MANQUANTE

ONCFM

Imputation des valeurs manquantes pour la colonne “margin_low”

1. **Imputation des valeurs manquantes par régression linéaire**

```
known_margin_low = bill_data.dropna(subset=["margin_low"])
missing_margin_low = bill_data[bill_data["margin_low"].isnull()]

X_known = known_margin_low.drop(columns=["is_genuine", "margin_low"])
y_known = known_margin_low["margin_low"]
X_missing = missing_margin_low.drop(columns=["is_genuine", "margin_low"])

# Création et entraînement du modèle de régression linéaire
lr_model = LinearRegression()
lr_model.fit(X_known, y_known)

# Prédiction des valeurs manquantes pour "margin_low"
predicted_values = lr_model.predict(X_missing)

# Imputation des valeurs manquantes dans bill_data
bill_data.loc[bill_data["margin_low"].isnull(), "margin_low"] = predicted_values

# Nouvelle vérification des valeurs manquantes
new_missing_values = bill_data.isnull().sum()
new_missing_values
```

2. **Contrôle d’absence de doublons**

```
duplicates = bill_data.duplicated().sum()
print("nombre de doublons dans le dataset après imputation des valeurs manquantes par régression linéaire : ",duplicates)
```

✓ 0.0s

```
nombre de doublons dans le dataset après imputation des valeurs manquantes par régression linéaire :  0
```

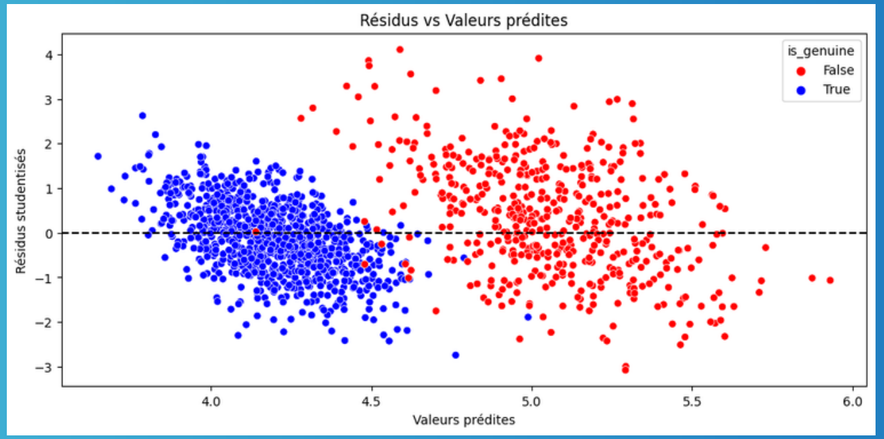
ONCFM

Evaluation de la régression linéaire

1

Vérification de la linéarité :

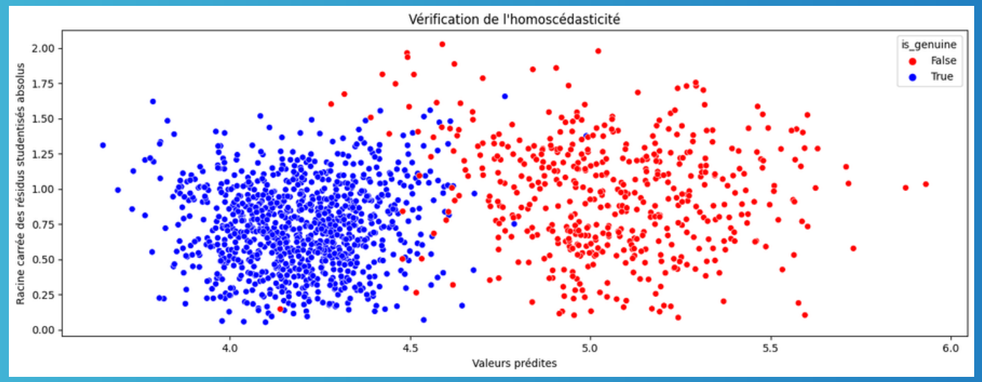
Les résidus se dispersent de manière relativement aléatoire autour de zéro, ce qui est bon signe pour la linéarité de la relation.



2

Vérification de l'homoscédasticité :

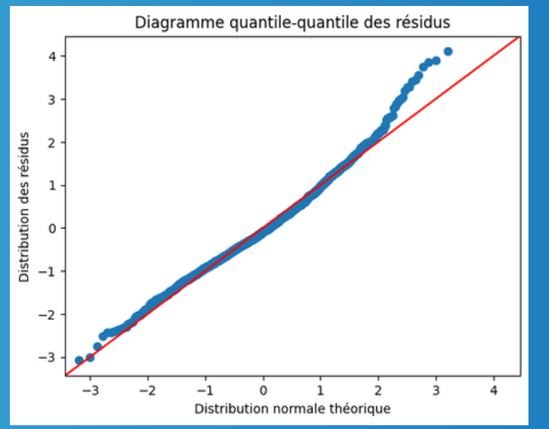
Les résidus montrent une dispersion constante sur toute la plage des valeurs prédites.



3

Normalité des résidus :

Les points du diagramme quantil-quantile des résidus suivent étroitement la diagonale à 45°, suggérant leur distribution normale.
On peut, toutefois, observer quelques déviations en queues (surtout supérieure), mais rien de très important



ONCFM

3 . ANALYSES UNIVARIEE ET MULTIVARIEE

ONCFM

Analyse univariée

	diagonal	height_left	height_right	margin_low	margin_up	length
count	1500.00	1500.00	1500.00	1500.00	1500.00	1500.00
mean	171.96	104.03	103.92	4.48	3.15	112.68
std	0.31	0.30	0.33	0.66	0.23	0.87
min	171.04	103.14	102.82	2.98	2.27	109.49
25%	171.75	103.82	103.71	4.02	2.99	112.03
50%	171.96	104.04	103.92	4.31	3.14	112.96
75%	172.17	104.23	104.15	4.87	3.31	113.34
max	173.01	104.88	104.95	6.90	3.91	114.44

- Diagonal:

- La diagonale des billets varie de 171.04 mm à 173.01 mm avec une moyenne de 171.96 mm.

- Height_left:

- La hauteur gauche des billets varie de 103.14 mm à 104.88 mm avec une moyenne de 104.03 mm.

- Height_right:

- La hauteur droite des billets varie de 102.82 mm à 104.95 mm avec une moyenne de 103.92 mm.

- Margin_low:

- La marge inférieure varie de 2.98 mm à 6.90 mm avec une moyenne de 4.49 mm.

- Margin_up:

La marge supérieure varie de 2.27 mm à 3.91 mm avec une moyenne de 3.15 mm.

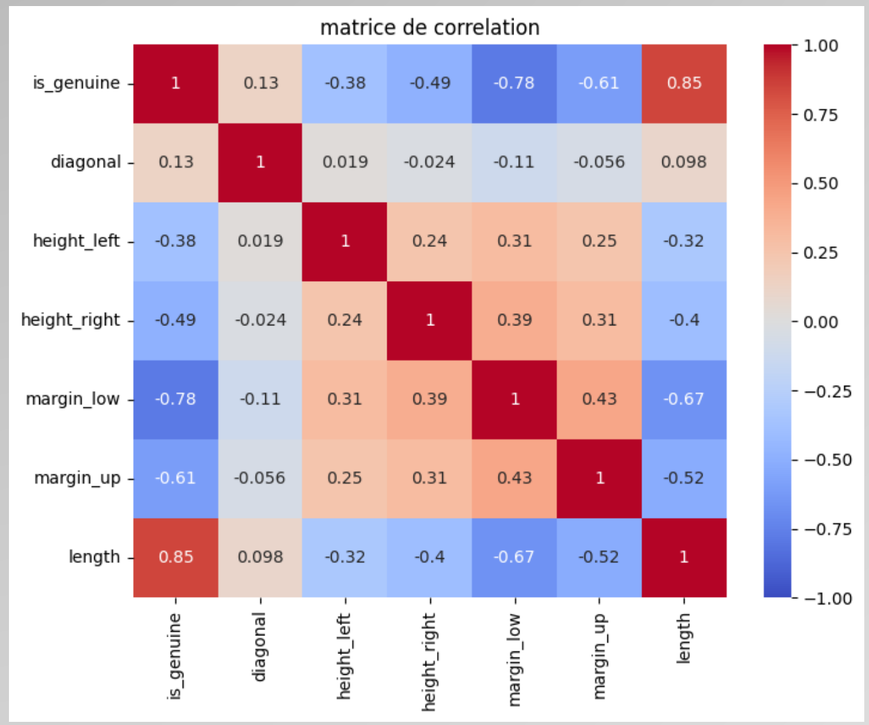
- Length:

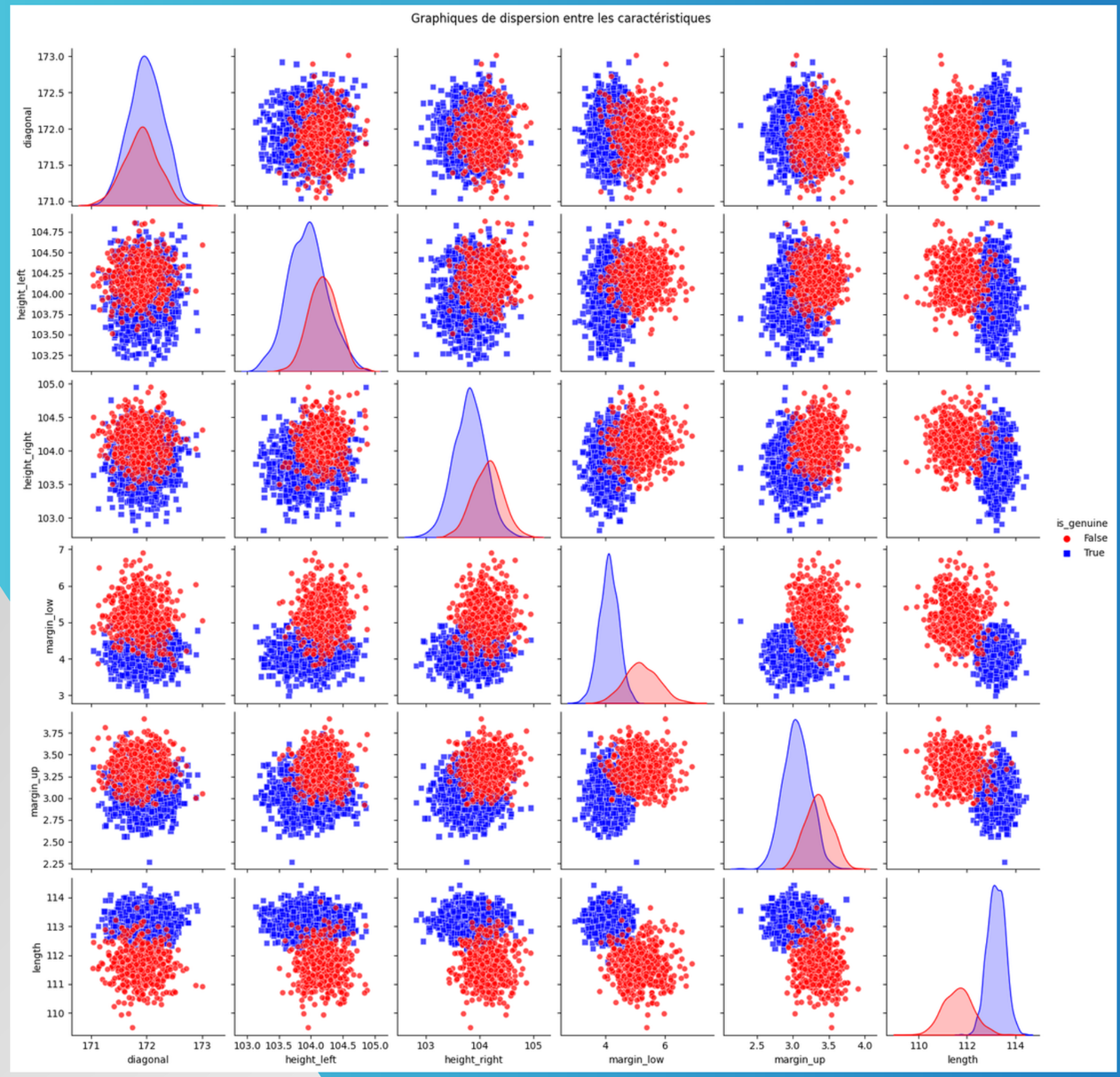
- La longueur des billets varie de 109.49 mm à 114.44 mm avec une moyenne de 112.68 mm.

ONCFM

Analyse multivariée

En observant la matrice de corrélation et les graphiques de dispersion, on observe que la longueur est très fortement et positivement corrélée avec l'authenticité, mais également que les marges inférieure et supérieure sont quant à elles, fortement corrélées négativement avec l'authenticité





ONCFM

4 . ALGORITHMES DE PRÉDICTION

ONCFM

4.1 - Régression logistique sur l'ensemble du dataset

1 **Séparation du dataset en ensembles d'entraînement et de test**

```
X = bill_data.drop("is_genuine", axis=1)
y = bill_data["is_genuine"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42, stratify = y)
```

2 **Entraînement du modèle**

```
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
# Prédiction sur l'ensemble de test
y_pred_train = logistic_model.predict(X_train)
y_pred_test = logistic_model.predict(X_test)
```

3 **Evaluation du modèle :**
Précision sur l'ensemble d'entraînement : 98.2 %
Précision sur l'ensemble de test : 99 %
Ces scores élevés indiquent que le modèle est capable de classer correctement un grand nombre d'observation

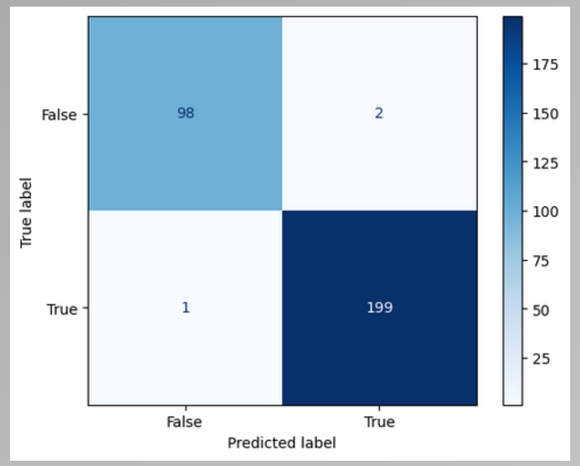
```
train_score = accuracy_score(y_train, y_pred_train)
test_score = accuracy_score(y_test, y_pred_test)

train_score, test_score
```

✓ 0.0s

```
(0.9891666666666666, 0.99)
```

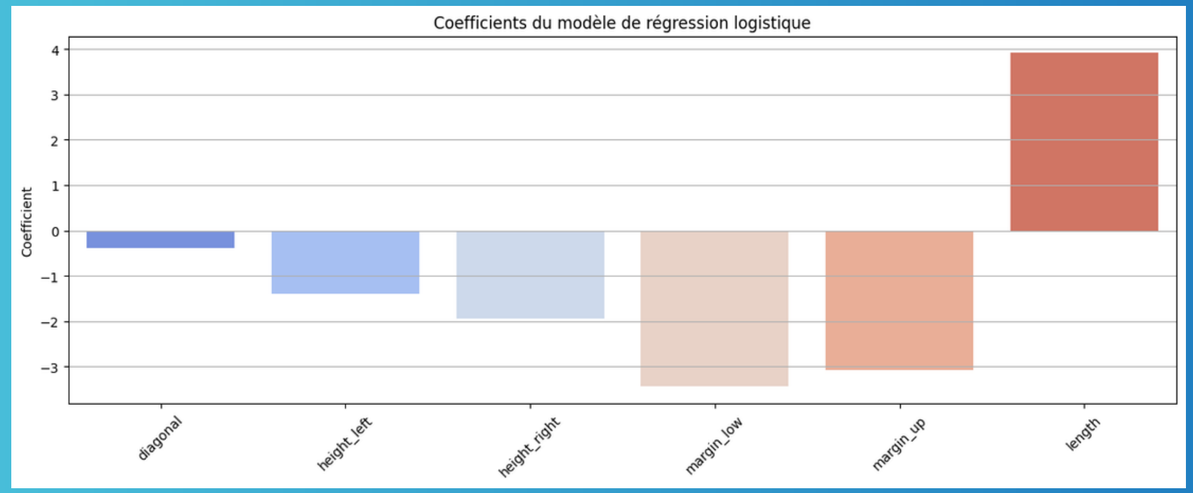
Vérification de la qualité du modèle



1

Matrice de confusion
- **Taux de faux positifs : 0.67 % du dataset .**
- **Taux de vrais négatifs : 0.33 % du dataset.**

2 **Coefficients du modèle**



3

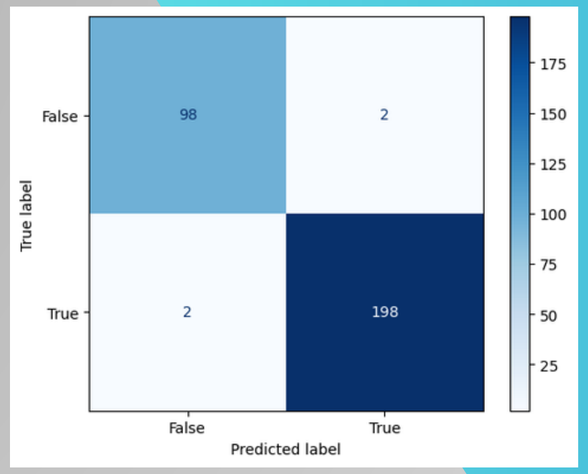
Validation croisée
- **résultats de la validation croisée :**
  - **1er fold : 99.00 %**
  - **2eme fold : 98.67 %**
  - **3eme fold : 99.67 %**
  - **4eme fold : 99.00 %**
  - **5eme fold : 99.00 %**
- **La performance moyenne sur les 5 folds est de 99.07 %.**
- **Ces scores suggèrent que le modèle de régression logistique est performant et généralise bien sur différentes partitions des données.**

ONCFM

4.2 - Vérification de la qualité du modèle pour une régression logistique avec les caractéristiques "length", "margin_low" et "margin_up".

1

Matrice de confusion
- Taux de faux positifs : 0.67 % du dataset .
- Taux de vrais négatifs : 0.67 % du dataset.



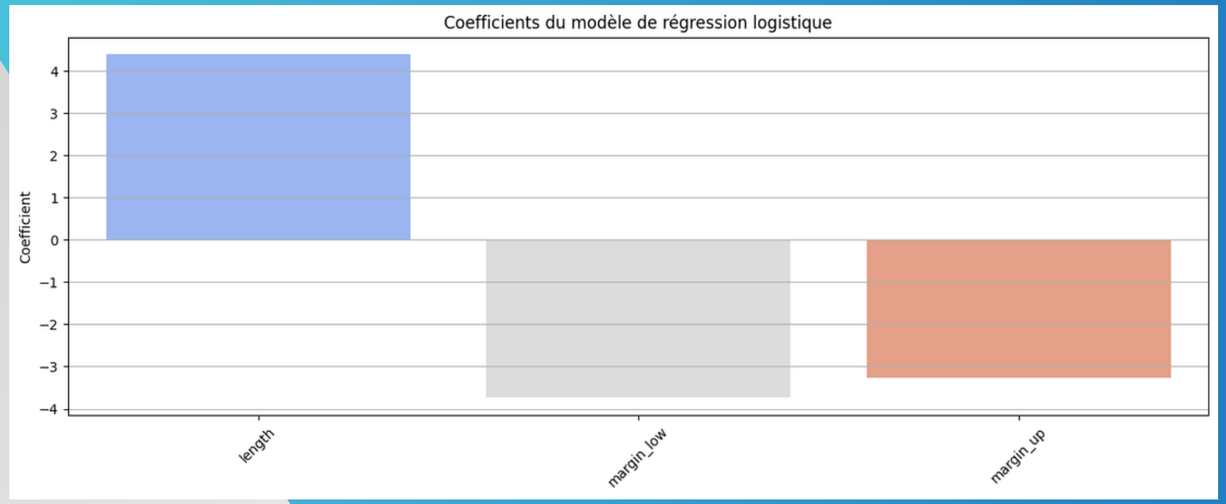
2

Coefficients du modèle pour chaque caractéristique :
- Length :
- Coefficient positif de 4.39, indiquant que l'augmentation de la longueur du billet augmente la probabilité qu'il soit authentique.
- Margin_up :
- Coefficient négatif de -3.27, signifiant que plus la marge inférieure augmentera, moins le billet aura de probabilité d'être authentique.
- Margin_low :
- Coefficient négatif de -3.74, signifiant, comme pour "margin_up" que plus la marge inférieure augmentera, moins le billet aura de probabilité d'être authentique.

3

- Exactitude sur l'ensembles d'entraînement : 99.17 % .
- Exactitude sur l'ensemble de test : 98.67 % .

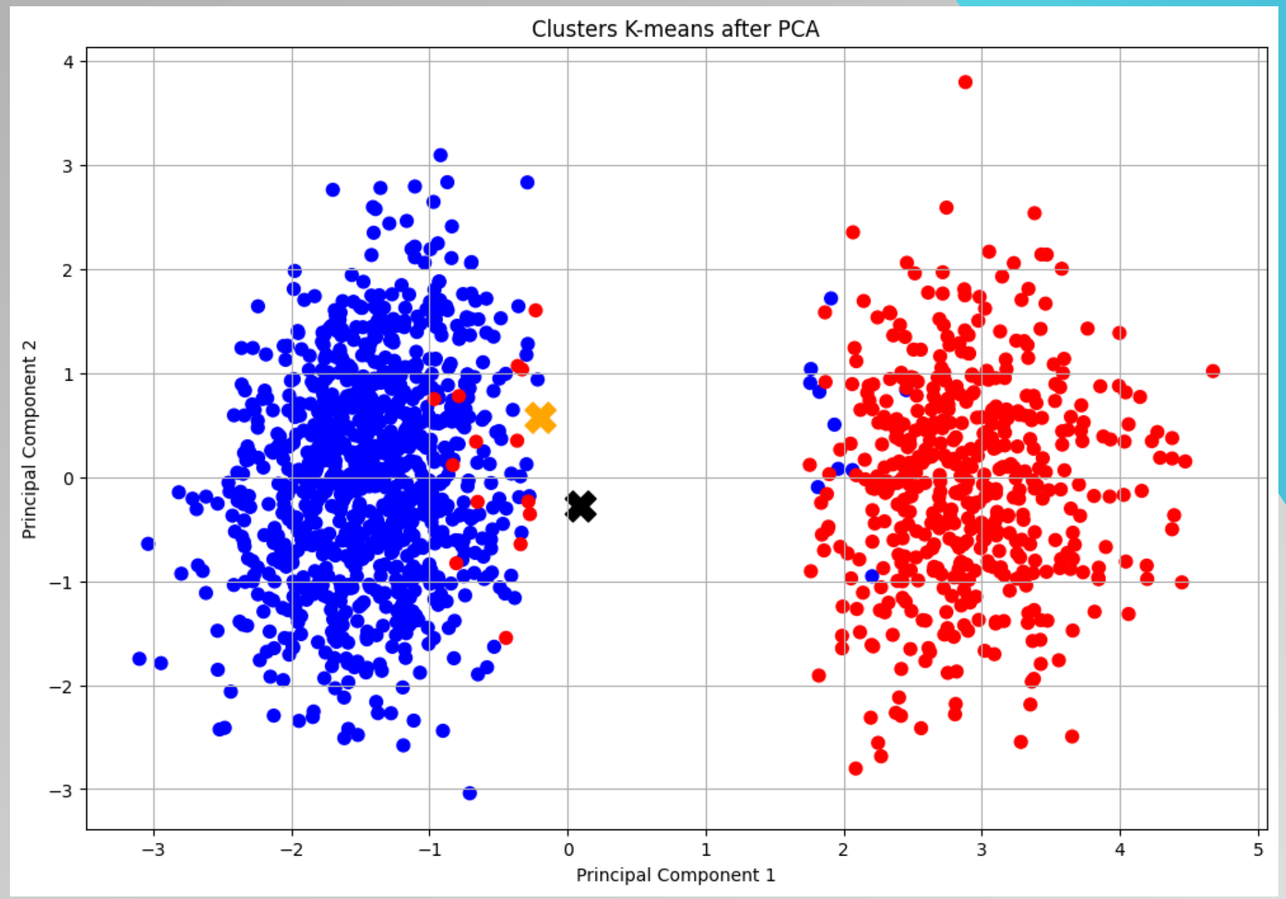
- Moyenne des scores de validation croisée (5-fold) : 98.53 % .
- Ces scores confirment la haute performance du modèle de régression logistique basé sur les caractéristiques "length", "margin_low" et "margin_up" pour distinguer les billets authentiques des faux billets. La validation croisée donne une estimation plus robuste de cette performance.



4.3 - Clustering K-means sur l'ensemble du dataset

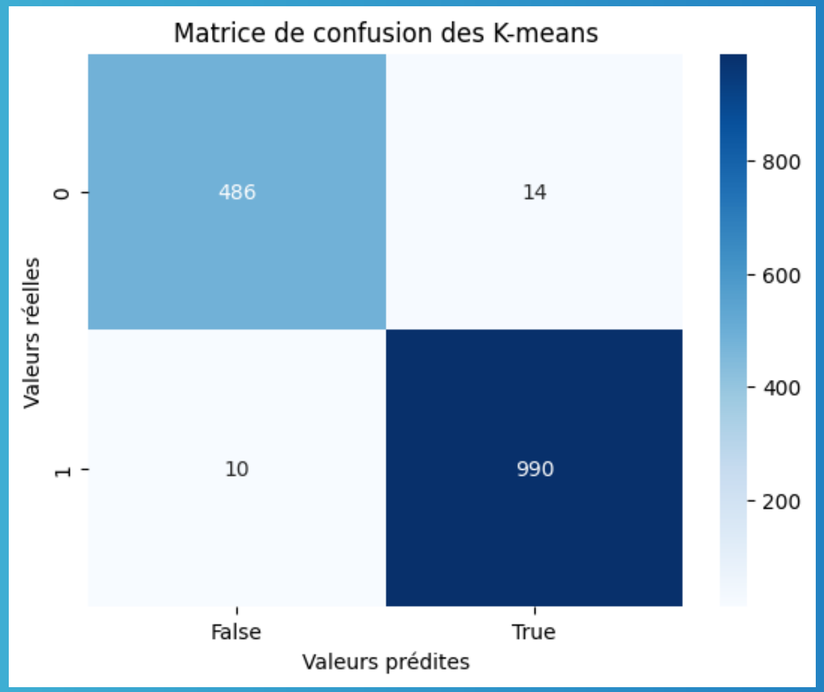
1

Visualisation des K-means et leurs centroïdes



2

Matrice de confusion
- Taux de faux positifs : 0.93 % du dataset .
- Taux de vrais négatifs : 0.67 % du dataset.



3

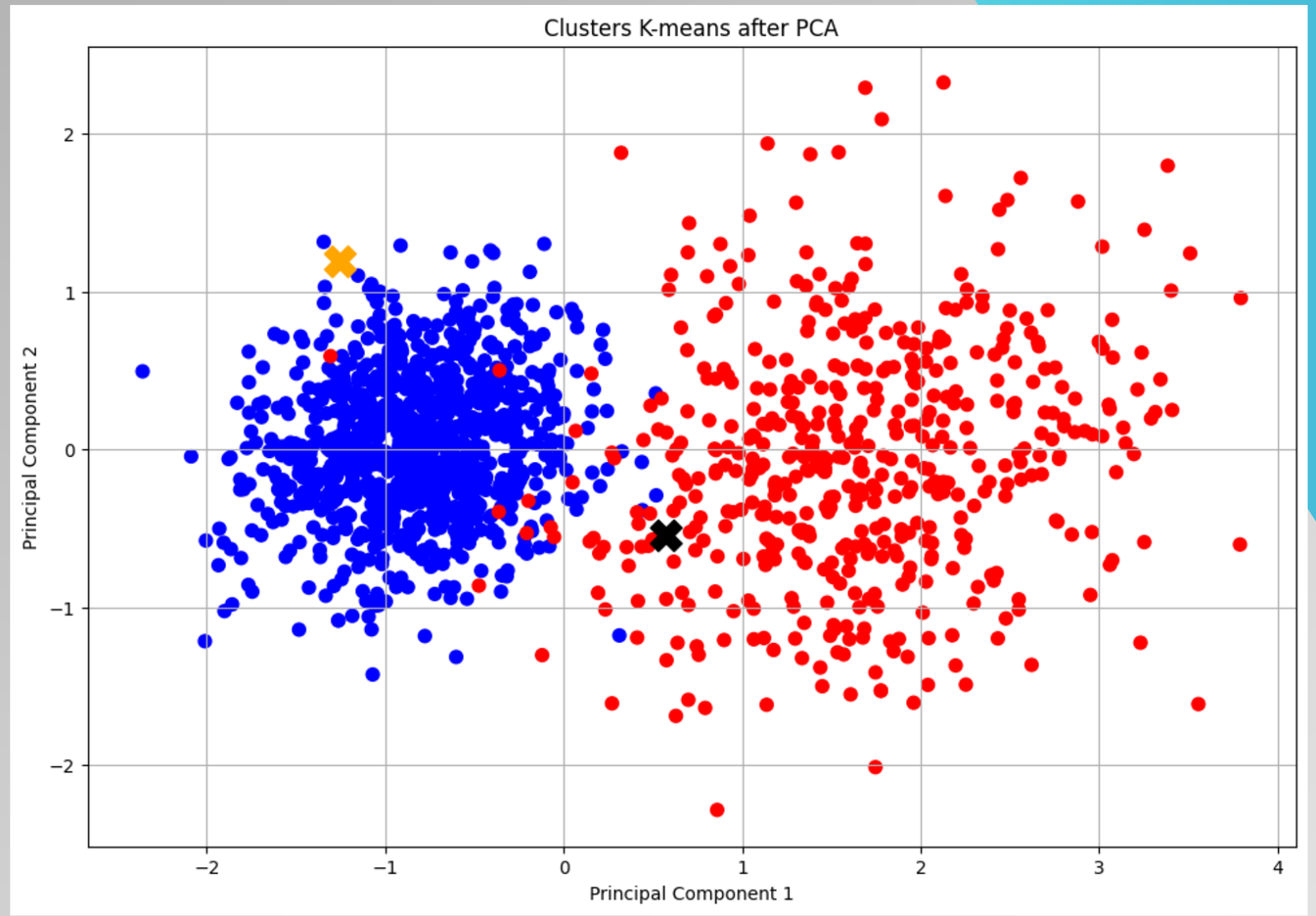
Score de Silhouette : 0.48
Ce score indique une bonne séparation dans les clusters

ONCFM

4.4 - Vérification de la qualité du modèle pour un clustering K-means avec les caractéristiques "length", "margin_low" et "margin_up".

1

Visualisation des K-means et leurs centroïdes

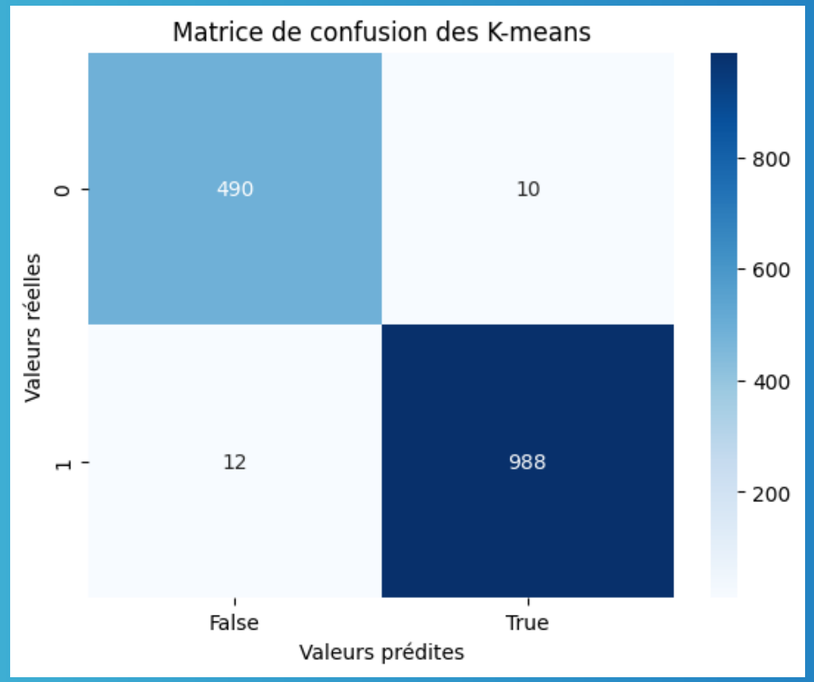


2

Matrice de confusion

- Taux de faux positifs : 0.67 % du dataset .

- Taux de vrais négatifs : 0.8 % du dataset.



3

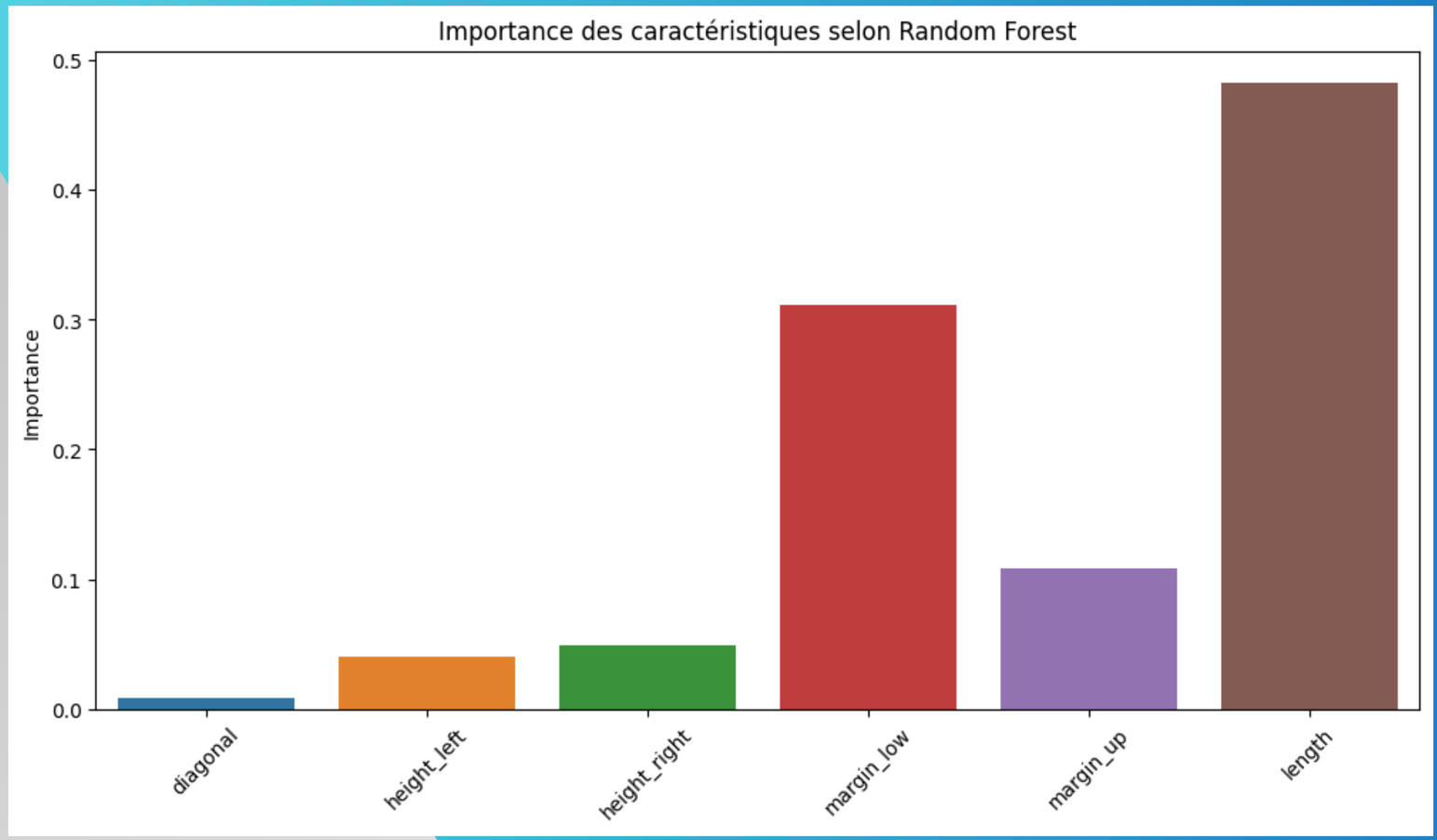
Score de Silhouette : 0.60

Ce score indique une meilleure séparation dans les clusters qu'avec l'ensemble du dataset

ONCFM

4.5 - Algorithme d’ensemble Random Forest

**Avec un taux de précision de 0.99 ,
la prédiction des arbres de décision désigne la caractéristique “length” comme la plus importante pour ce modèle, suivie de “margin_low” et “margin_up”**



ONCFM

5. CHOIX DU MODE D'APPRENTISSAGE ET CRÉATION DE LA FONCTION À EXPORTER

ONCFM

Mon choix se porte sur la régression logistique.
Cette dernière apporte plus de stabilité et de précision , en comparaison de la clusterisation par k-means.
J'encapsule donc mon algorithme dans une fonction nommée "modele_logistique_combined".

Je crée le fichier "reg_log.pkl" contenant l'entraînement du dataset et le modèle de régression logistique. (score d'entrainement : 0.9925 , score de test : 0.9867)

Je crée le fichier "bill_detect.py", permettant d'utiliser l'application streamlit en ligne, pour le test de l'algorithme :

https://arnaud-caille-p10-detection-faux-billets.streamlit.app/

J'ai également créé un simple fichier "algotest.ipynb" contenant l'algorithme à fin de test.

ONCFM

MERCI