Détection de faux billets

Projet 12

sommaire

Contexte

Contexte

L'Organisation nationale de lutte contre le faux-monnayage (ONCFM) souhaite développer une solution permettant de différencier automatiquement les vrais des faux billets en euros, à partir de leurs caractéristiques géométriques, afin de renforcer la lutte contre la contrefaçon.

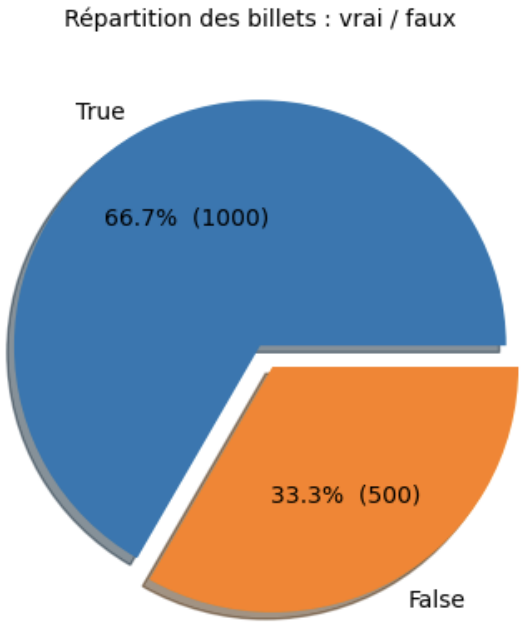
Objectif

L'objectif est de construire un algorithme capable de classifier un billet comme "vrai" ou "faux" en se basant sur des dimensions précises (longueur, hauteur, marges, diagonale). Ces différences, invisibles à l'œil nu, peuvent être détectées par une machine et analysées par un modèle de classification.

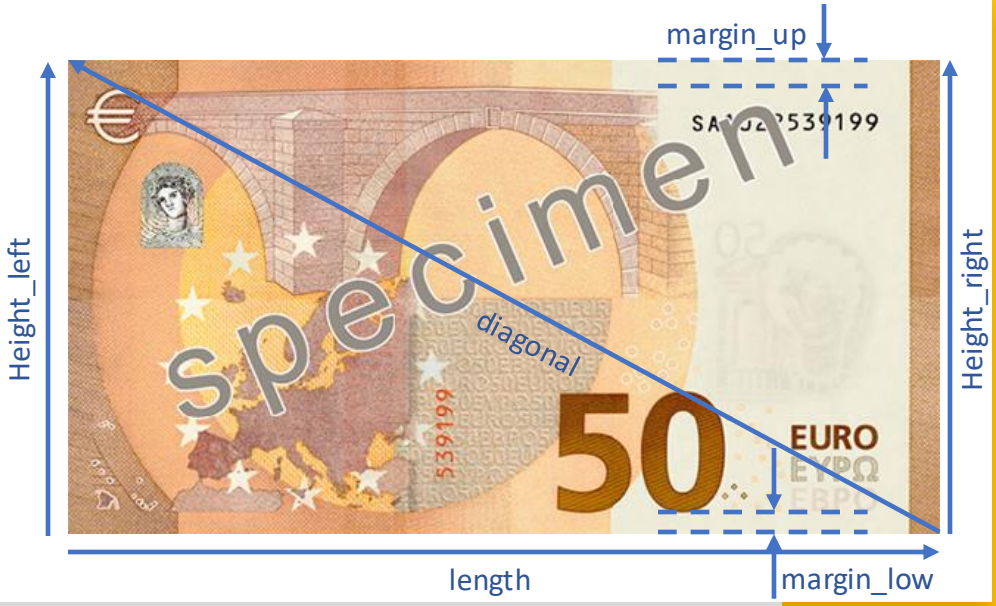
Analyse des données

1500 billets avec 6 variables géométriques:
- 1000 Vrais
- 500 Faux

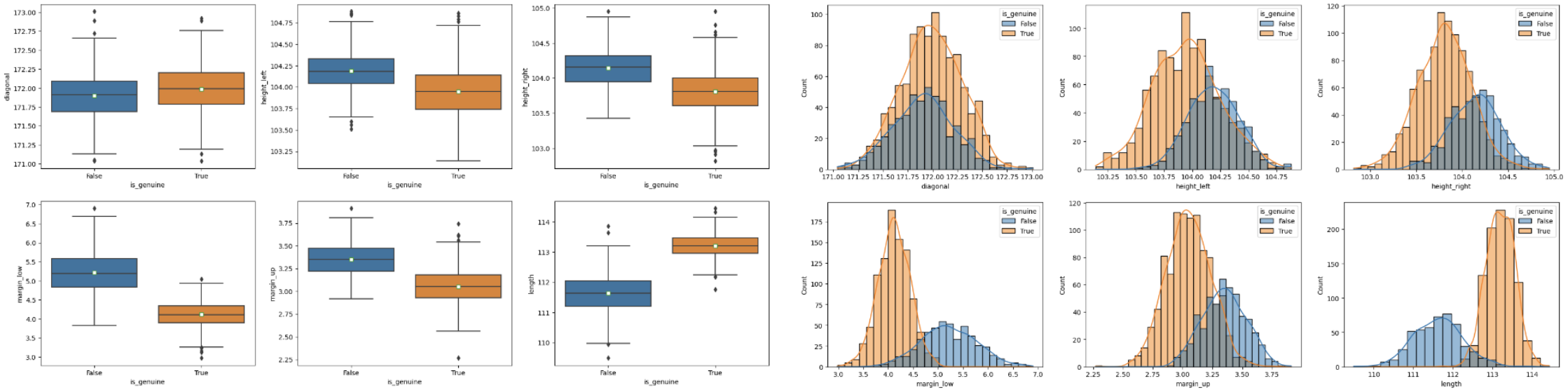
37 valeurs manquantes dans la colonne ‘margin_low’



	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



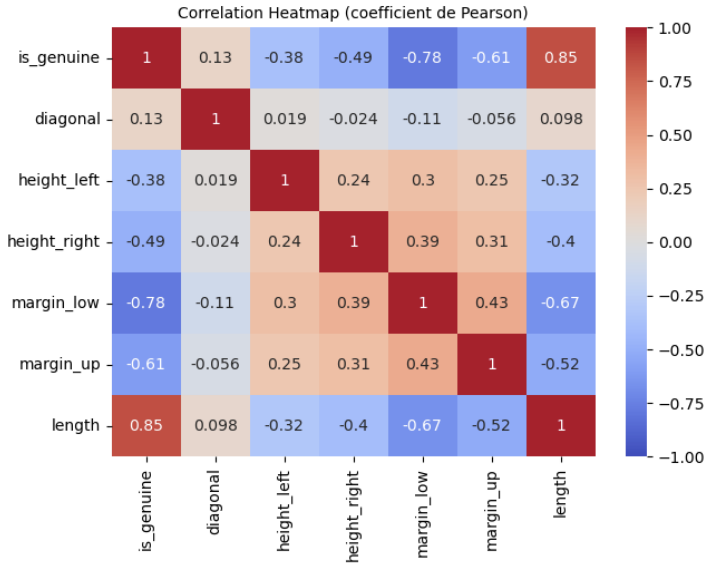
Analyse des données

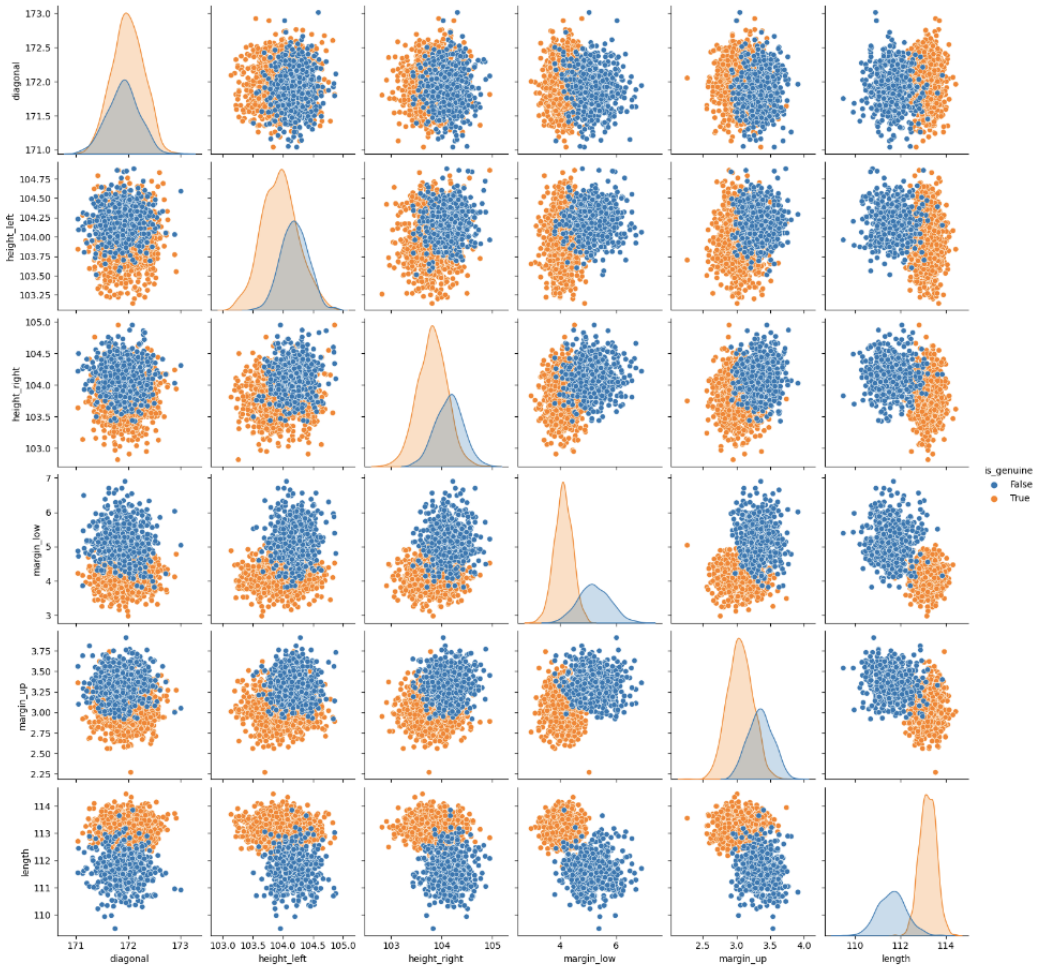


Caractéristiques des faux billets :

- moins long et plus haut (diagonale équivalente aux vrais)
- marges supérieure et inférieure plus grandes
- Les écarts de distributions vrais/faux billets sont plus marqués pour margin_low et length.

Corrélation entre les variables





- **length** a la plus forte corrélation positive avec **is_genuine (+0,85)**
- **margin_low** et **margin_up** présentent des corrélations négatives notables avec **is_genuine (-0,78 et -0,61)**,

Ces caractéristiques sont importantes pour différencier les vrais billets des faux.

Traitement des valeurs manquantes

Remplacement ou suppression?

	+	-
Remplacement	**Préserve la taille de l'échantillon** : Aucune donnée n'est perdue, ce qui est important si le dataset est petit. **Améliore la robustesse des modèles** : Évite de biaiser les résultats en maintenant la cohérence du dataset.	**Introduit de l'incertitude** : Les valeurs imputées ne reflètent pas les données réelles, ce qui peut fausser l'analyse. **Méthodes simplistes** : L'imputation par la moyenne/médiane peut masquer des relations importantes dans les données.
Suppression	**Simple et direct** : Facile à mettre en œuvre sans ajout de complexité. **Évite les biais d'imputation** : Les données sont strictement réelles sans ajout d'estimations artificielles. **Pertinent pour les petites proportions de données manquantes**	**Perte d'information** : Si beaucoup de données sont manquantes, cela peut réduire significativement la taille du dataset. **Biais potentiel** : La suppression peut biaiser les résultats si les données manquantes sont liées à des caractéristiques spécifiques.

Remplacement par régression linéaire multiple

Echantillonage → Entrainement → Evaluation → Analyse des resultats

Train_data = 1467 billets
Sans valeurs manquantes

Test_data = 37 billets
Avec les valeurs manquantes cible ‘margin_low’

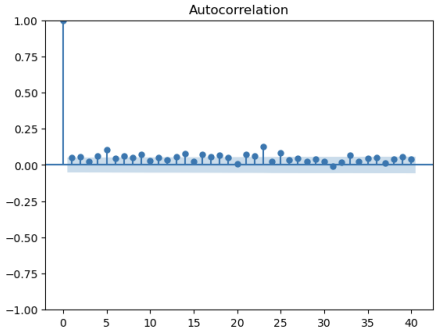
Tous les coefficients associés aux variables ont des p-values très faibles (< 0.05), ce qui signifie qu'ils sont statistiquement significatifs pour expliquer ‘margin_low’

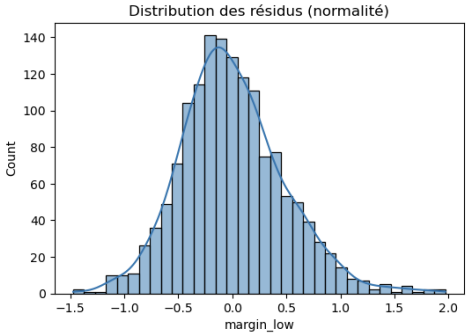
$R^2 = 0,47$

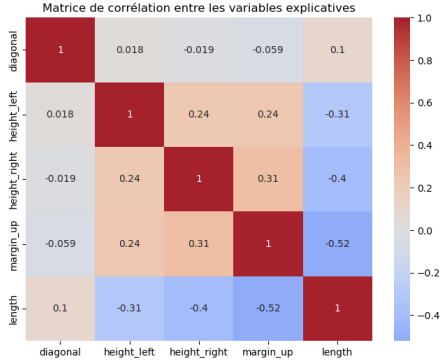
Le modèle explique 47%
De la variation totale

Analyse des résidus:
- Indépendance
- Normalité
- Homoscédasticité
- Colinéarité

RLM : analyse des résultats

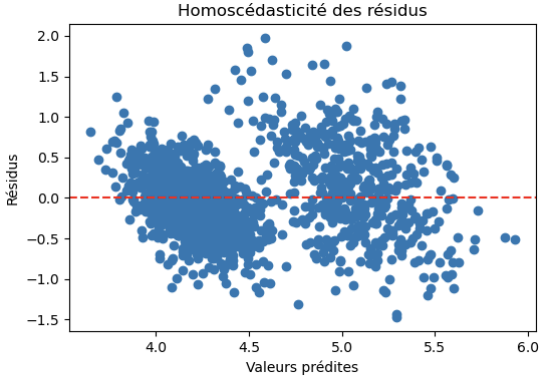






Variance Inflation Factor (VIF) pour chaque variable :

	Variable	VIF
0	const	590198.238883
1	diagonal	1.013613
2	height_left	1.138261
3	height_right	1.230115
4	margin_up	1.404404
5	length	1.576950



Indépendance

Normalité

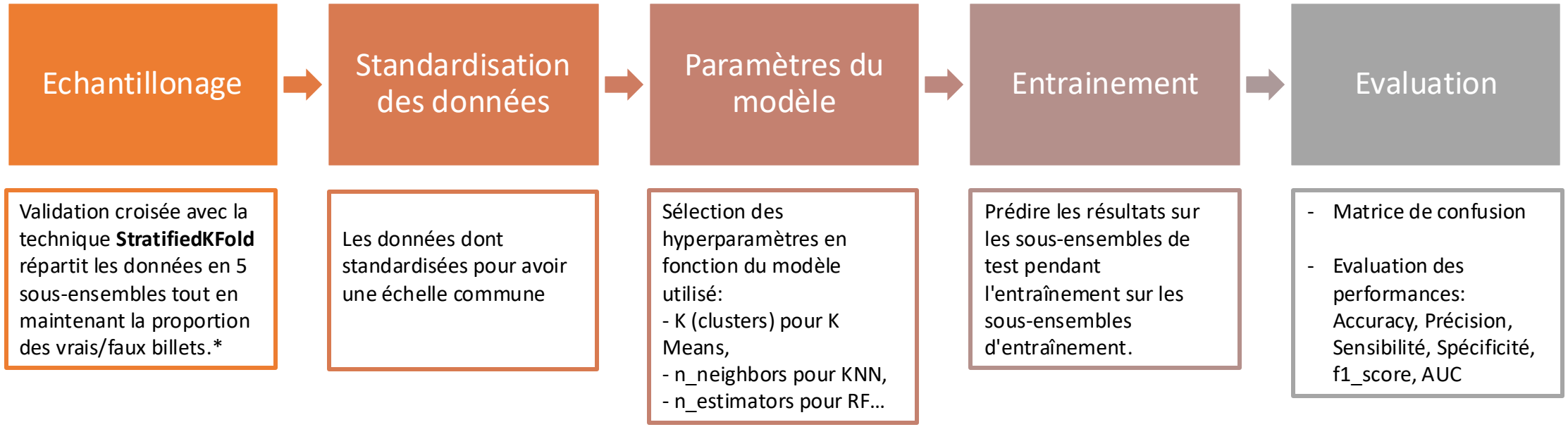
Colinéarité

Homoscédasticité

Les algorithmes utilisés

- K Means
- Regression logistique
- KNN
- Random Forest
- Gradient Boosting

Etapes clés de la mise en place d'un modèle



*Pour le K Means, aucun échantillonnage effectué sur la data.

K Means

Simple et rapide : Facile à implémenter et généralement rapide pour des jeux de données de taille modérée.

Efficace pour des clusters bien séparés : Fonctionne bien quand les clusters sont globulaires et distincts.

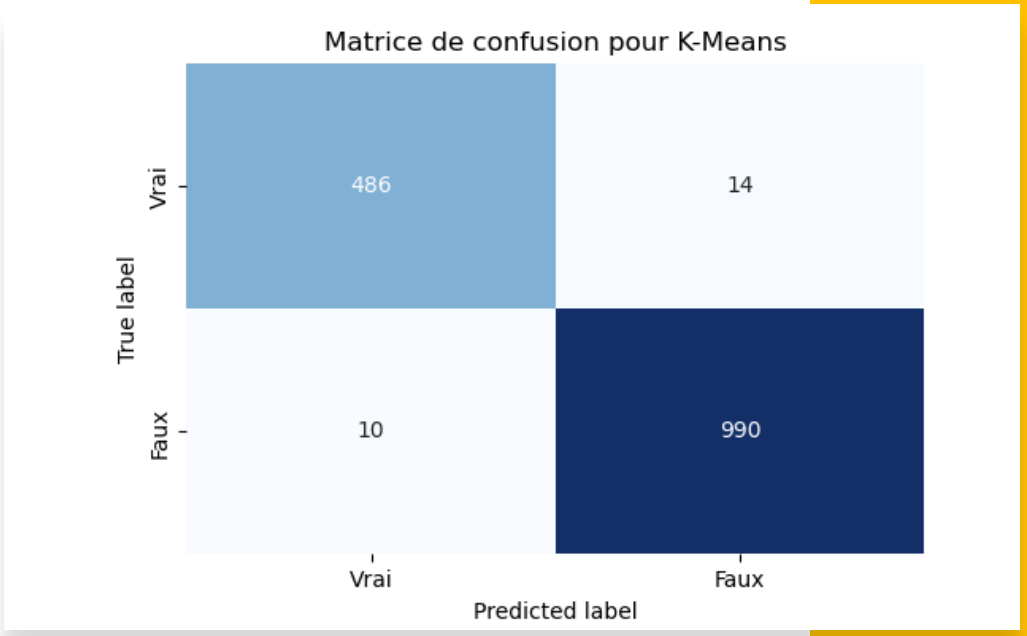
Scalable : Peut être appliqué à de grands ensembles de données avec une complexité linéaire.

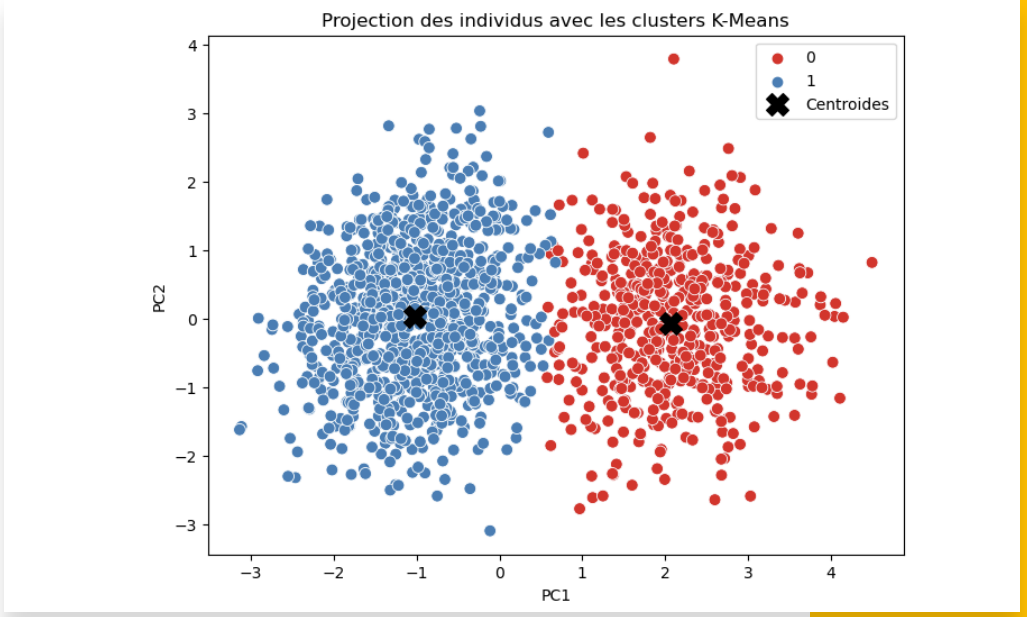
Sensibilité aux outliers : Les valeurs extrêmes peuvent fortement affecter les résultats.

Nécessite de définir K : Le nombre de clusters (K) doit être spécifié à l'avance.

Coordonnées des centroides des clusters :

	diagonal	height_left	height_right	margin_up	margin_low	length
0	171.987729	103.945129	103.805588	3.052540	4.117843	113.196066
1	171.899153	104.200383	104.152520	3.351734	5.221734	111.630847





Regression logistique

Rapide et efficace : Convient bien pour des jeux de données de taille modérée et converge rapidement.

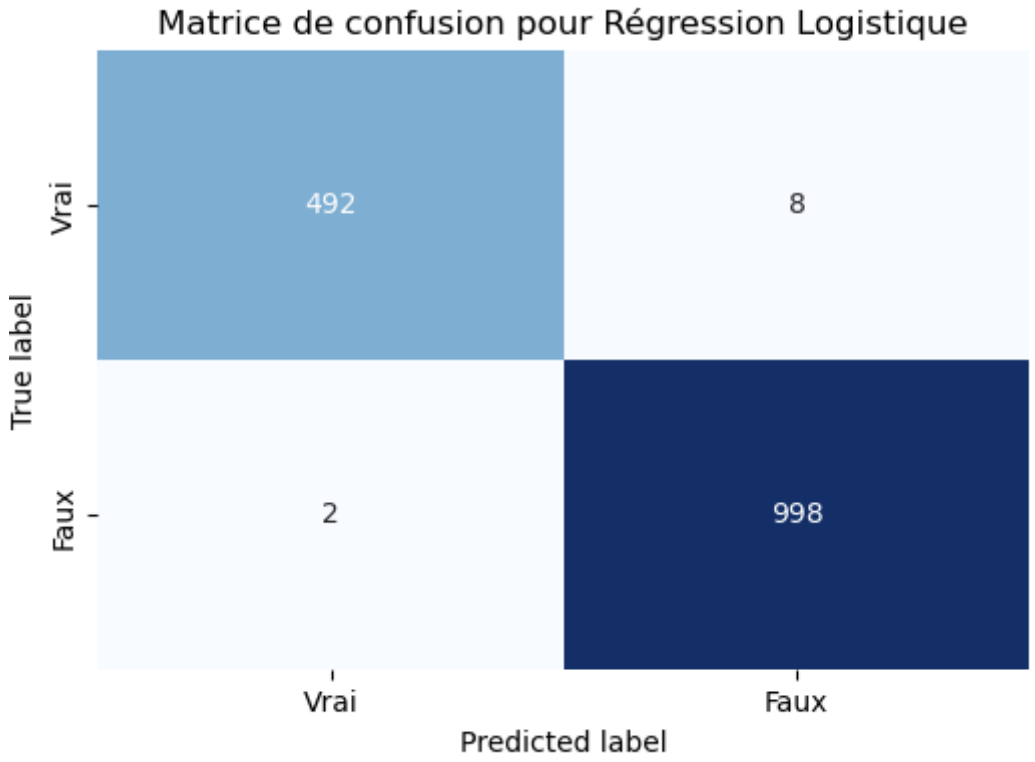
Probabilités : Fournit des probabilités de classe, utiles pour évaluer l'incertitude des prédictions.

Bonne performance sur des données linéaires : Fonctionne bien lorsque la relation entre les variables indépendantes et la variable cible est linéaire.

Sensibilité aux outliers : Les valeurs extrêmes peuvent avoir un effet important sur les coefficients.

Pas adapté aux grandes dimensions : Moins performant lorsque le nombre de variables est beaucoup plus élevé que le nombre d'observations.

Ne gère pas bien les classes déséquilibrées : Peut avoir des performances limitées sur des ensembles de données avec un fort déséquilibre entre les classes.



KNN

Simple et intuitif : Le concept de "voisinage" est facile à comprendre et à expliquer.

Pas d'hypothèses sur les données : Contrairement à d'autres algorithmes comme la régression logistique, KNN ne fait pas d'hypothèses sur la distribution des données.

Flexible : Peut s'adapter aussi bien aux problèmes de classification qu'aux problèmes de régression.

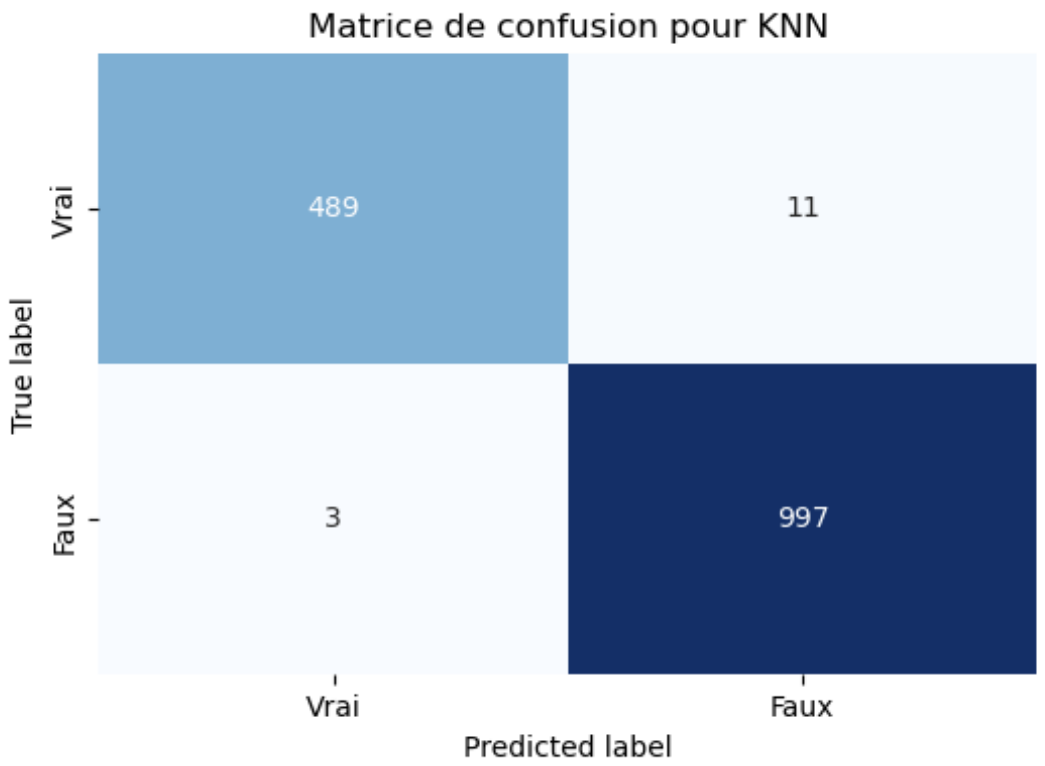
Capture les relations non linéaires : Peut modéliser des relations non linéaires entre les variables.

Lent avec de grandes données : Le temps de calcul devient élevé avec de grands jeux de données.

Sensibilité au choix de K : Le choix du nombre de voisins (K) peut affecter fortement la précision.

Sensibilité aux outliers : Les valeurs aberrantes peuvent fortement influencer les résultats.

Nécessite une standardisation : KNN est sensible à l'échelle des variables, ce qui nécessite de standardiser les données pour de meilleures performances.



Random Forest

Robuste aux outliers et au bruit

Bonne performance :
Très performant sur une grande variété de problèmes (classification et régression) et offre une précision élevée.

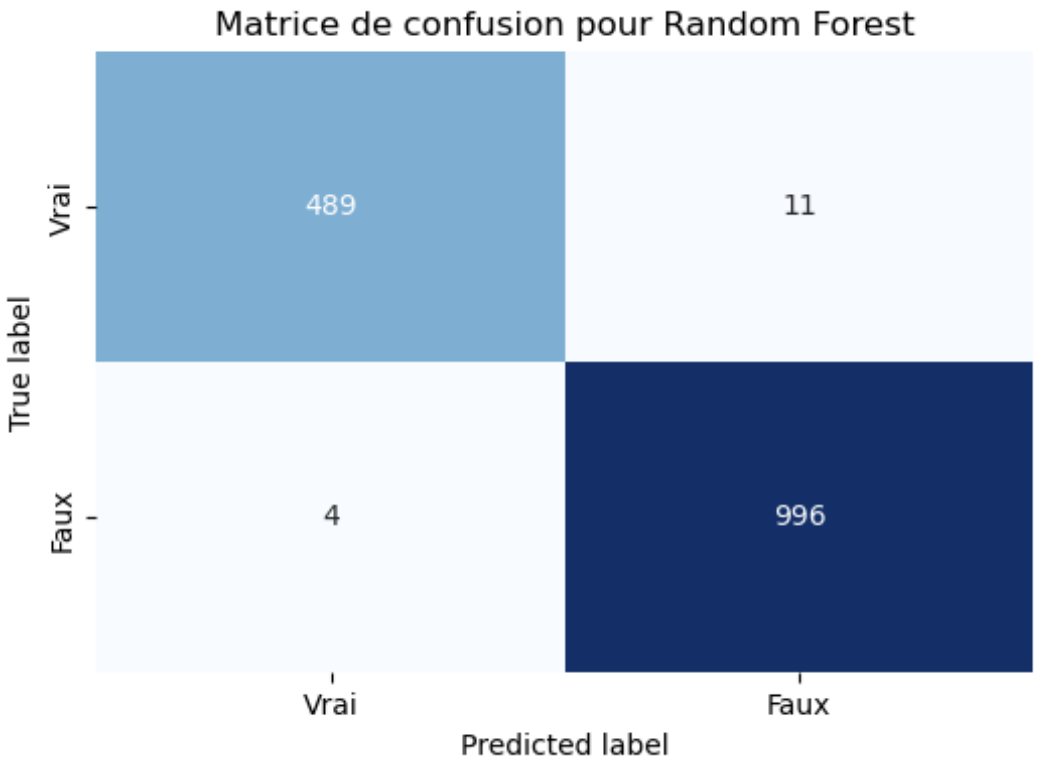
Réduction du l'overfitting : Le Random Forest a moins tendance à surapprendre, car il moyenne les prédictions de plusieurs arbres.

Gère bien les données déséquilibrées : En ajustant les poids des classes ou en modifiant les critères d'échantillonnage, Random Forest peut bien gérer les classes déséquilibrées.

Temps de calcul : L'algorithme peut être lent pour de très grands ensembles de données.

Peu interprétable : Contrairement à un seul arbre de décision, qui est facile à interpréter visuellement, un Random Forest est constitué de centaines d'arbres, ce qui rend le modèle difficile à expliquer.

Sensibilité à un très grand nombre de variables : Si le jeu de données contient trop de variables non pertinentes, cela peut diluer l'importance des variables pertinentes et impacter la performance.



Gradient Boosting

Haute précision : Le Gradient Boosting est souvent l'un des algorithmes de machine learning les plus performants, capable de minimiser l'erreur de manière itérative et progressive.

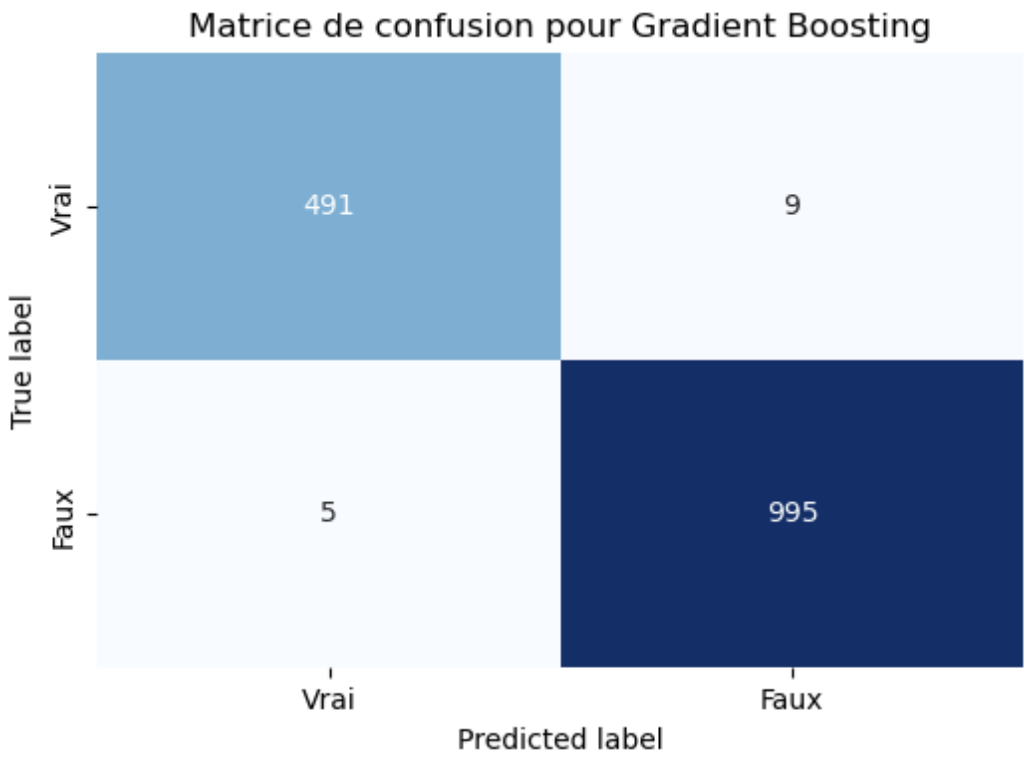
Gère les données non linéaires : Il fonctionne bien sur les relations complexes et non linéaires entre les variables.

Flexibilité : Il peut être adapté à des problèmes de classification et de régression, et supporte différents types de fonctions de perte.

Temps de calcul : Il est généralement plus lent que Random Forest en raison de son processus séquentiel (les modèles sont construits les uns après les autres).

Sensibilité aux hyperparamètres : Nécessite un ajustement précis des hyperparamètres (comme le taux d'apprentissage et le nombre d'arbres)

Peu interprétable : Similaire à Random Forest, il est difficile d'expliquer les prédictions individuelles en raison de la complexité du modèle final.



Comparaison des performances

Modèle	Accuracy	Précision	Sensibilité	Spécificité	f1_score	AUC
K-Means	0.984000	0.986056	0.990	0.028	0.988024	0.9810
Régression Logistique	0.993333	0.992048	0.998	0.984	0.995015	0.9910
KNN	0.990667	0.989087	0.997	0.978	0.993028	0.9875
Random Forest	0.990000	0.989076	0.996	0.978	0.992526	0.9870
Gradient Boosting	0.990667	0.991036	0.995	0.982	0.993014	0.9885

Application fonctionnelle

```python
def detection_fx_billets(model, nom_fichier):
    """
    Fonction permettant la détection de faux billets à partir d'un algorithme de classification déjà entraîné.

    Paramètres:
    - model: Modèle de classification déjà entraîné
    - nom_fichier: Chemin vers le fichier CSV contenant les informations sur les billets

    Retourne:
    - DataFrame avec les prédictions et les probabilités de chaque billet
    """

    # Importation des données
    df = pd.read_csv(nom_fichier)

    # Sélection des données significatives pour la prédiction
    X = df[['diagonal', 'height_left', 'height_right', 'margin_low', 'margin_up', 'length']]

    # Standardisation des données
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X.values)

    # Prédictions et probabilités
    y_pred = model.predict(X_scaled)  # Utilisation des prédictions directes du modèle
    predict_proba = model.predict_proba(X_scaled)[:, 1]  # Probabilité d'être un vrai billet

    # Création d'une copie du DataFrame pour ajouter les résultats
    df_pred = df.copy()
    df_pred['prediction'] = y_pred
    df_pred['probabilité vrai %'] = np.round(predict_proba * 100, 2)

    return df_pred
```

Merci pour votre attention