Cahier des charges

Algorithme de détection automatique de faux billets

Sommaire

1. Contexte du projet

Nous souhaitons mettre en place un algorithme capable de **différencier automatiquement les vrais des faux billets** en euros, à partir de leurs **caractéristiques géométriques** mesurées par une machine.
Ce projet s'inscrit dans une démarche de lutte contre la **contrefaçon de billets**.

2. Objectifs

Une machine mesure plusieurs dimensions sur chaque billet.
Au fil des années, des **différences systématiques** ont été détectées entre les vrais et les faux billets, **invisibles à l'œil nu**, mais exploitables par un algorithme.

L'objectif est donc de construire un modèle capable, à partir de ces **mesures géométriques**, de prédire si un billet est **vrai ou faux**.

3. Modèle de données

Caractéristiques disponibles :

Chaque billet est décrit par **6 variables géométriques** :

- length : longueur du billet (en mm)
- height_left : hauteur à gauche (en mm)
- height_right : hauteur à droite (en mm)
- margin_up : marge entre le haut du billet et l'image (en mm)
- margin_low : marge entre le bas du billet et l'image (en mm)

- diagonal : diagonale du billet (en mm)

Un fichier d'exemple est fourni, contenant :

- **1 500 billets**
    - 1 000 vrais
    - 500 faux
- Une colonne spécifie la nature (vrai ou faux)

Une **analyse descriptive** des données est attendue dans un premier temps :

- Statistiques sur les dimensions
- Répartition vrais/faux
- Visualisations utiles

4. Algorithme

Langages acceptés :

- **Python**

Fonctionnement attendu :

L'algorithme devra :

- Prendre en entrée un fichier CSV contenant les **dimensions de plusieurs billets**
- Prédire (Probabilité) pour chacun s'il est **vrai ou faux**

Le fichier de production s'intitulera billets_production.csv et respectera le format mentionné plus haut (sans étiquette).

Méthodes à comparer :

Vous devrez implémenter **4 approches** :

1. **Régression logistique**
2. **K-means**
3. **KNN**
4. **Random Forest**

Évaluation :

Chaque méthode sera comparée à l'aide d'une **matrice de confusion** :

- Analyse des **faux positifs**
- Analyse des **faux négatifs**
- Comparaison des performances globales

5. Livrables attendus

- Un **notebook Python** contenant :
    - L'analyse exploratoire
    - L'entraînement des modèles
    - Les comparaisons de performances
    - Les prédictions finales
- Le code doit être **reproductible**
- L'algorithme doit fonctionner **sans la colonne "étiquette"**, uniquement sur les **6 dimensions**

6. Compléments fonctionnels supplémentaires

Création d'une API (FastAPI)

Le modèle devra être encapsulé dans une **API REST** développée avec **FastAPI**.
Cette API doit fournir un **endpoint** permettant :

- de **soumettre un fichier CSV** contenant uniquement les caractéristiques géométriques des billets
- de recevoir en retour les **prédictions** sous forme de **JSON**

Interface utilisateur (Streamlit)

Une **interface web simple** sera développée avec **Streamlit**, permettant à l'utilisateur de :

- **Uploader un fichier .csv**
- Lancer les prédictions en appelant l'API FastAPI

- Afficher les résultats :
    - Table de prédictions
    - Statistiques (nombre de vrais/faux)
    - Graphiques si pertinent

L'interface doit être **déployée** sur Streamlit Cloud

Modalité d'évaluation (Jour J)

Le **jour de l'évaluation** :

- L'évaluateur accédera à l'interface Streamlit via un **lien web**
- Il pourra **uploader un fichier CSV de test inconnu**
- Le système devra alors :
    - Envoyer le fichier à l'API
    - **Prédire** les classes
    - Afficher les résultats **en temps réel**