TP 3 : Classification avec k-Nearest Neighbors (k-NN)

Durée: 4 heures

Thème principal: Classification supervisée

1. Introduction théorique

Le modèle des k plus proches voisins (*k-NN*) est une méthode de classification simple et efficace. Pour classifier un point, on identifie ses k plus proches voisins dans l'espace des caractéristiques en utilisant une mesure de distance (par exemple, la distance euclidienne) et on attribue la classe la plus représentée parmi ces voisins.

Dans ce TP, nous allons construire un modèle de classification supervisé pour identifier des types de moteurs électriques à partir de leurs paramètres (résistance, inductance, vitesse nominale, couple maximal).

2. Objectifs pédagogiques

- Comprendre le fonctionnement de l'algorithme k-NN.
- Implémenter un modèle k-NN avec Scikit-learn.
- Analyser les performances d'un modèle de classification dans un contexte mécatronique.

3. Prérequis et preparation

3.1 Connaissances nécessaires

- Notions sur les distances dans les espaces de caractéristiques.
- Concepts de classification supervisée.

3.2 Matériel nécessaire

Installer Scikit-learn et Matplotlib :

```
pip install scikit-learn matplotlib pandas
```

4. Exercices

Exercice 1 : *Chargement et préparation des données*

Le fichier CSV (nommé `moteurs_structured.csv`) contenant les colonnes suivantes :

- **Type de moteur** (classe) : valeurs possibles "DC", "Asynchrone", "Pas-à-pas"
- **Résistance (ohms)**, **Inductance (mH)**, **Vitesse nominale (RPM)**, **Couple maximal (Nm)**

1- Chargez le fichier avec pandas.
2- Encodez les types de moteurs en valeurs numériques avec `LabelEncoder`.

```
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data['Type de moteur'] = encoder.fit_transform(data['Type de moteur'])
```

3- Séparez les colonnes en entrées (X) et étiquettes (y).
4- Divisez les données en ensembles d'entraînment (80%) et de test (20%).
5- Affichez un résumé statistique des données préparées.

Exercice 2 : *Implémentation du modèle k-NN*

1- Importez le classifieur k-NN et créez le modèle avec `k=3`.

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
```

2- Entraînez le modèle avec les données d'entraînement.

```
model.fit(X_train, y_train)
```

Exercice 3 : *Evaluation du modèle*

1- Précisez les étiquettes pour l'ensemble de test.

```
y_pred = model.predict(X_test)
```

2- Générez une matrice de confusion.

```
from sklearn.metrics import confusion_matrix, accuracy_score
print(confusion_matrix(y_test, y_pred))
print(f'Accuracy: {accuracy_score(y_test, y_pred):.2f}')
```

3- Affichez la précision, le rappel et le score F1.
4- Interprétez les résultats.

Exercice 4 : *Visualisation des performances*

1- Réduisez les données à deux caractéristiques principales (par exemple, **résistance** et **couple**).
2- Tracez les points de données avec des couleurs différentes selon les classes prédites.
3- Ajoutez des annotations pour les étiquettes mal classées.

Exercice 5 : *Comparaison avec une régression linéaire multiple*

1. Créez un modèle de régression linéaire multiple.
2. Entraînez-le avec les mêmes données d'entraînement.
3. Évaluez les performances du modèle en termes d'erreur quadratique moyenne (MSE) et de coefficient de détermination R^2.
4. Comparez les résultats de la régression linéaire et du k-NN en termes de précision et de pertinence pour ce problème.

5. Résultats attendus

- Un modèle k-NN capable de classifier les moteurs.
- Une matrice de confusion montrant les précisions et erreurs.
- Un graphique illustrant les données classifiées par le modèle.

6. Travail à rendre

- Un notebook contenant : Le code complet pour chaque étape et les sorties graphiques et les évaluations du modèle.
- Une synthèse des performances du modèle et une discussion des paramètres influ