K-Nearest Neighbors (KNN) Practical Session

Objective

Apply the K-Nearest Neighbors method with a configurable number of classes.

Part 1: Data Generation with a Variable Number of Classes

We add a parameter to specify the number of classes when generating the data.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Paramètre pour le nombre de classes
n_classes = 3  # Vous pouvez changer cette valeur

# Génération des données synthétiques avec un nombre de classes
variable
X, y = make_classification(n_samples=300, n_features=2,
n_informative=2, n_redundant=0,
                           n_classes=n_classes, n_clusters_per_class=1,
random_state=42)

# Visualisation des données générées
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='coolwarm', s=50)
plt.title(f"Données générées pour la classification avec {n_classes}
classes")
plt.xlabel("Caractéristique 1")
plt.ylabel("Caractéristique 2")
plt.show()

# Division des données en ensemble d'entraînement et de test (80/20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Normalisation des données
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Part 2: Applying the KNN Method

We use scikit-learn's KNN classifier with a variable number of classes.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Création du modèle KNN avec k=5
knn = KNeighborsClassifier(n_neighbors=5)

# Entraînement du modèle sur les données d'entraînement
knn.fit(X_train, y_train)

# Prédiction sur l'ensemble de test
y_pred = knn.predict(X_test)

# Évaluation du modèle
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision du modèle KNN avec {n_classes} classes:
{accuracy:.2f}")

# Visualisation des résultats sur l'ensemble de test
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, cmap='coolwarm',
s=50)
plt.title(f"Prédictions KNN sur l'ensemble de test ({n_classes}
classes)")
plt.xlabel("Caractéristique 1")
plt.ylabel("Caractéristique 2")
plt.show()
```

Part 3: Tuning the Parameter k with a Variable Number of Classes

We test different numbers of neighbors while keeping the specified number of classes.

```
k_values = [1, 3, 5, 7, 9]
accuracies = []

for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
    print(f"k={k}, Précision={accuracy:.2f}")

# Visualisation de l'impact de k sur la précision
plt.plot(k_values, accuracies, marker='o')
plt.title(f"Impact du nombre de voisins sur la précision ({n_classes}
```

```
classes)")
plt.xlabel("Nombre de voisins (k)")
plt.ylabel("Précision")
plt.grid(True)
plt.show()
```