SENTONGO HAMZA

MASTER’S IN DATA SCIENCE

QUESTION 1

First, in order to have a good K-means classifier we need to have time series data in a 2d format(Shape: **(n\_samples, n\_timestamps)**).Instead of random x and y data points. We don’t include the z because we are dealing with unsupervised learning.

Precautions

-same length of time series

-Data should be normalized or standardized to avoid scale bias.

-We should use feature extraction (mean, variance, frequency components) should be applied before clustering.

We have already practiced the above in the previous exercises

QUESTION 2

Recognition Accuracy Results:

SVC with linear kernel: 0.80

LinearSVC (linear kernel): 0.78

SVC with RBF kernel: 0.80

SVC with polynomial (degree 3) kernel: 0.78

A diagram of different sizes and colors

AI-generated content may be incorrect.

**CODE**

import matplotlib.pyplot as plt

from matplotlib import colormaps

from sklearn import datasets, svm, metrics

from sklearn.model\_selection import train\_test\_split

from sklearn.inspection import DecisionBoundaryDisplay

# Load Iris dataset

iris = datasets.load\_iris()

X = iris['data'][:, :2]   # Use only first two features

y = iris['target']

# Split into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

C = 1.0  # SVM regularization parameter

models = [

    ("SVC with linear kernel", svm.SVC(kernel="linear", C=C)),

    ("LinearSVC (linear kernel)", svm.LinearSVC(C=C, max\_iter=10000)),

    ("SVC with RBF kernel", svm.SVC(kernel="rbf", gamma=0.7, C=C)),

    ("SVC with polynomial (degree 3) kernel", svm.SVC(kernel="poly", degree=3, gamma="auto", C=C)),

]

# Train, test, and display accuracy for each model

print("Recognition Accuracy Results:")

for name, clf in models:

    clf.fit(X\_train, y\_train)

    y\_pred = clf.predict(X\_test)

    acc = metrics.accuracy\_score(y\_test, y\_pred)

    print(f"{name}: {acc:.2f}")

# Plot decision boundaries

fig, sub = plt.subplots(2, 2, figsize=(8, 8))

plt.subplots\_adjust(wspace=0.4, hspace=0.4)

X0, X1 = X[:, 0], X[:, 1]

for (name, clf), ax in zip(models, sub.flatten()):

    disp = DecisionBoundaryDisplay.from\_estimator(

        clf,

        X,

        response\_method="predict",

        cmap=colormaps['coolwarm'],

        alpha=0.8,

        ax=ax,

        xlabel=iris.feature\_names[0],

        ylabel=iris.feature\_names[1],

    )

    ax.scatter(X0, X1, c=y, cmap=colormaps['coolwarm'], s=20, edgecolors="k")

    ax.set\_title(name)

plt.show()

Question 3