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

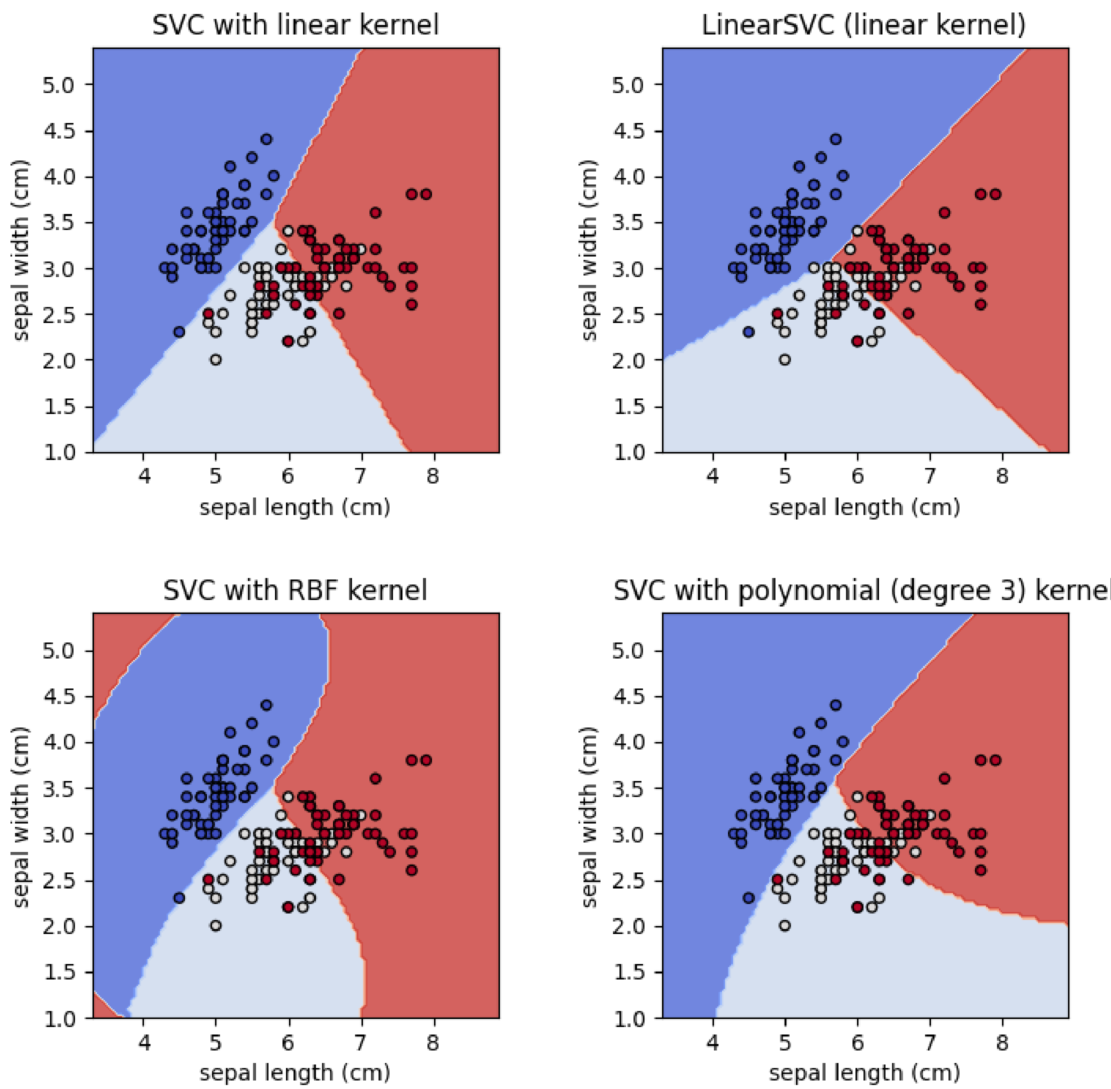
Recognition Accuracy Results:

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



CODE

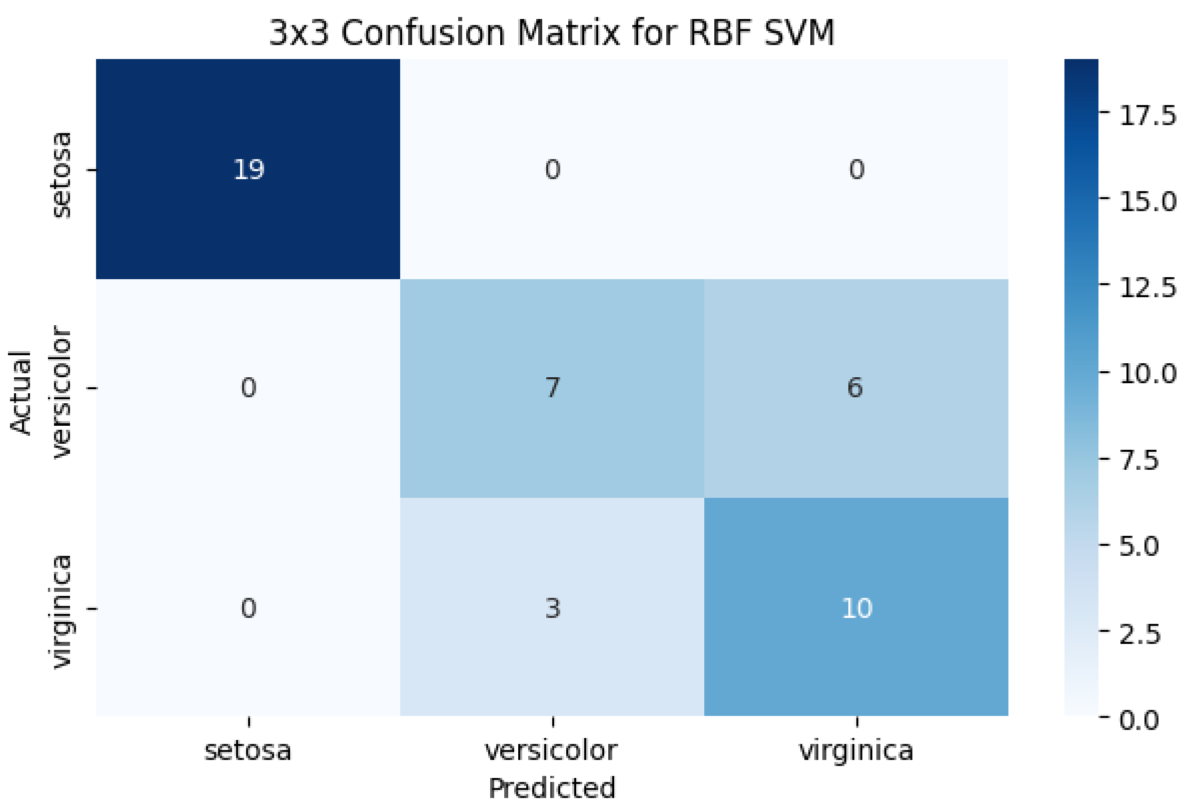
```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 10:14:14 2024

@author: turunenj
```

```
"""

#https://scikit-
learn.org/stable/auto_examples/svm/plot_iris_svc.html#sphx-glr-
auto-examples-svm-plot-iris-svc-py



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



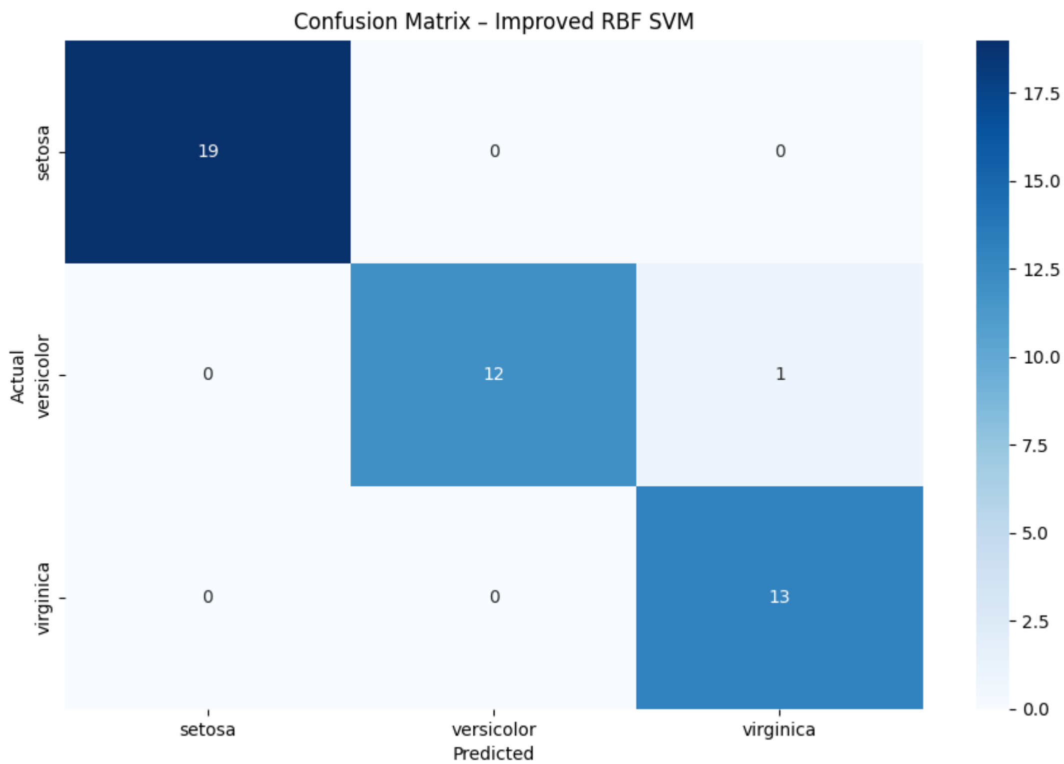
Code

```
# -*- coding: utf-8 -*-
"""
Modified SVM_example.py to include 3x3 confusion matrix
"""

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split

# Load Iris dataset
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Train SVM (RBF kernel example)
clf = svm.SVC(kernel="rbf", gamma=0.7, C=1.0)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

```
# Create and plot 3x3 confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=iris.target_names,
yticklabels=iris.target_names)
plt.title("3x3 Confusion Matrix for RBF SVM")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Question 4

Improved Recognition Accuracy (RBF kernel): 0.98



CODE

```
# -*- coding: utf-8 -*-
"""
Improved SVM_example.py – higher recognition accuracy
"""


from sklearn import datasets, svm, metrics
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Load full Iris dataset (use all 4 features)
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Feature scaling for better SVM performance
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into train/test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)

# Choose one improved model (RBF kernel)
clf = svm.SVC(kernel="rbf", gamma=0.5, C=10)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Compute and display recognition accuracy
acc = metrics.accuracy_score(y_test, y_pred)
print(f"Improved Recognition Accuracy (RBF kernel): {acc:.2f}")

# Plot 3x3 confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
```

```
plt.figure(figsize=(5, 4))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=iris.target_names,
yticklabels=iris.target_names)
plt.title("Confusion Matrix – Improved RBF SVM")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.tight_layout()
plt.show()
```

Question 5

Classification Accuracy: 0.9778

Cross-validation is where we repeatedly split the dataset into training and testing subsets, training the model on the training data, and evaluating it testing data

Code

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug  6 13:58:40 2024

@author: turunenj
"""


from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.metrics import accuracy_score

# Load iris dataset
iris = load_iris()

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    iris.data, iris.target, test_size=0.3, random_state=0
)

# Create and train the classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Calculate and print accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Classification Accuracy: {accuracy:.4f}")
```

Question 6

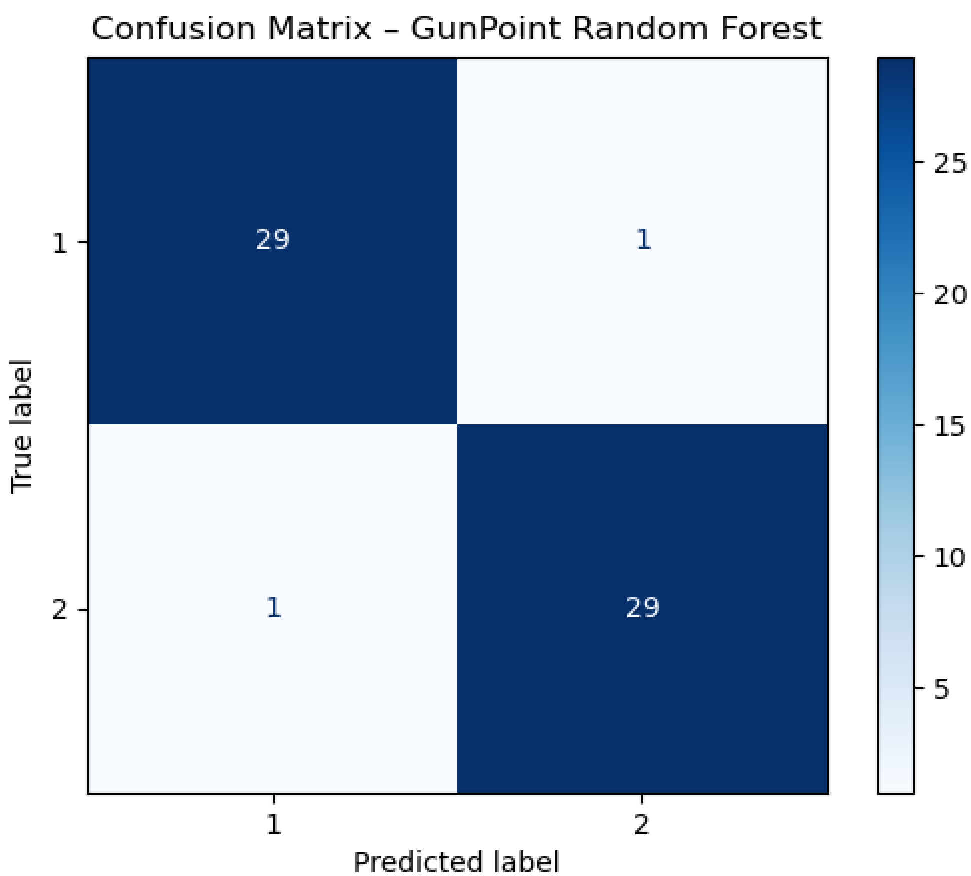
I chose the GunPoint dataset (via aeon) from the TSC archive and it has 2 classes.
I trained a scikit-learn RandomForestClassifier on the data.

Accuracy: 92.50 %

Modifications

Used all available time-series channels (shape: n_samples × n_channels × n_timepoints) rather than flattening to 2D

Applied z-normalisation per series (mean=0, std=1) to reduce scale differences.



Code

```
from aeon.datasets import load_classification

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay, accuracy_score
```

```python
import matplotlib.pyplot as plt
import numpy as np

# Load GunPoint dataset
X, y = load_classification("GunPoint")

# Flatten time series so RandomForest can handle it (2D input)
n_samples, n_channels, n_timepoints = X.shape
X_reshaped = X.reshape(n_samples, n_channels * n_timepoints)

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X_reshaped, y, test_size=0.3, random_state=42, stratify=y
)

# Train Random Forest
clf = RandomForestClassifier(n_estimators=200, max_depth=10,
random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

# Accuracy
acc = accuracy_score(y_test, y_pred)
print(f"Accuracy: {acc * 100:.2f}%")
```

```
# Confusion matrix figure

cm = confusion_matrix(y_test, y_pred)

disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=clf.classes_)

disp.plot(cmap="Blues")

plt.title("Confusion Matrix – GunPoint Random Forest")

plt.show()
```