BAKU HIGHER OIL SCHOOL

INFORMATION TECHNOLOGY DEPARTMENT
INFORMATION SECURITY
DIVISION

Machine Learning

Home Assignment 4

Assignment name: SVM Algorithm

Student name: Huseyn Abdullayev

Group number: CS 25

Instructor: Leyla Muradkhanli

MASTER OF CYBER SECURITY BAKU 2025

Dataset Description and Preprocessing Steps

The dataset used for this assignment is the Iris Dataset. It contains 150 samples of iris flowers, with information on their measurements and species.

The dataset includes the following key columns:

• sepal.length: Sepal length in cm.

• sepal.width: Sepal width in cm.

• petal.length: Petal length in cm.

• petal.width: Petal width in cm.

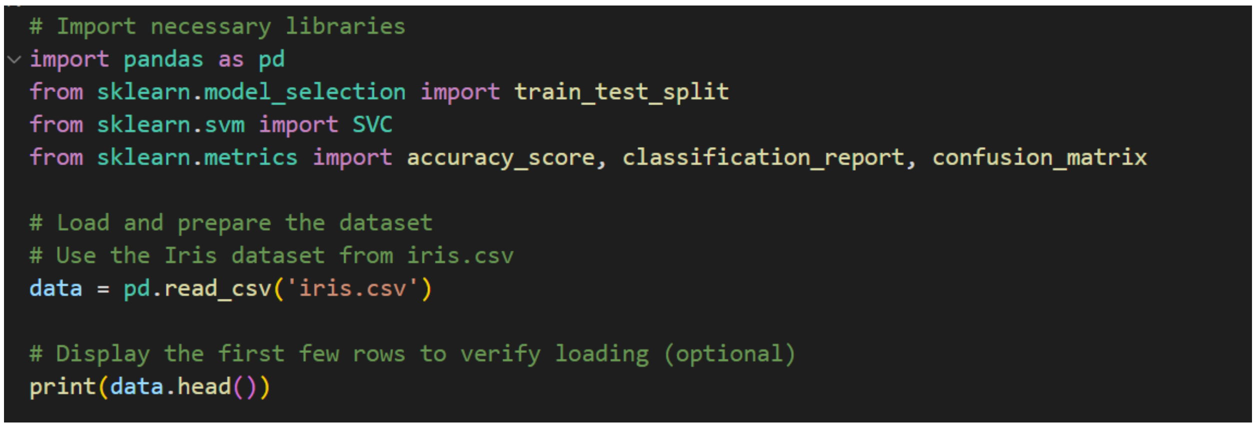
• variety: Target variable (Setosa, Versicolor, Virginica).

The target variable is "variety," which is categorical (multiclass).

Data Preprocessing Steps:

1. **Loading Data**:

The dataset was imported using pandas.read_csv('iris.csv').



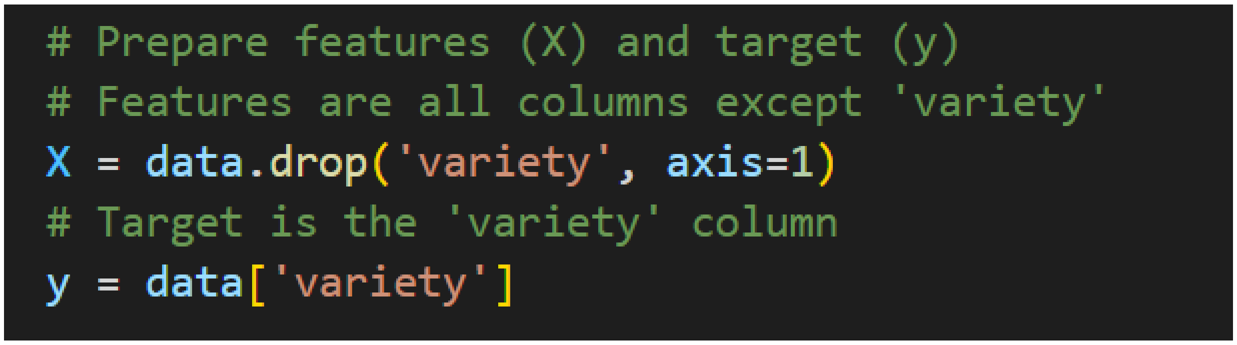
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load and prepare the dataset
# Use the Iris dataset from iris.csv
data = pd.read_csv('iris.csv')

# Display the first few rows to verify loading (optional)
print(data.head())
```

2. **Data Cleaning**:

No missing values were present in the dataset. All features are numeric, so no further cleaning was required.

3. **Encoding Categorical Labels**:

The target "variety" is categorical but handled directly by scikit-learn models. No encoding was necessary for features as they are all numeric.



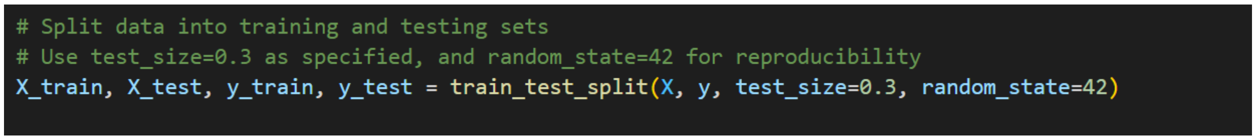
```python
# Prepare features (X) and target (y)
# Features are all columns except 'variety'
X = data.drop('variety', axis=1)
# Target is the 'variety' column
y = data['variety']
```

Feature Selection and Splitting:

Selected features: sepal.length, sepal.width, petal.length, petal.width (as X).

Target: variety (as y).

The dataset was then split into 70% training data (105 samples) and 30% testing data (45 samples) using train_test_split() with random_state=42 for reproducibility.



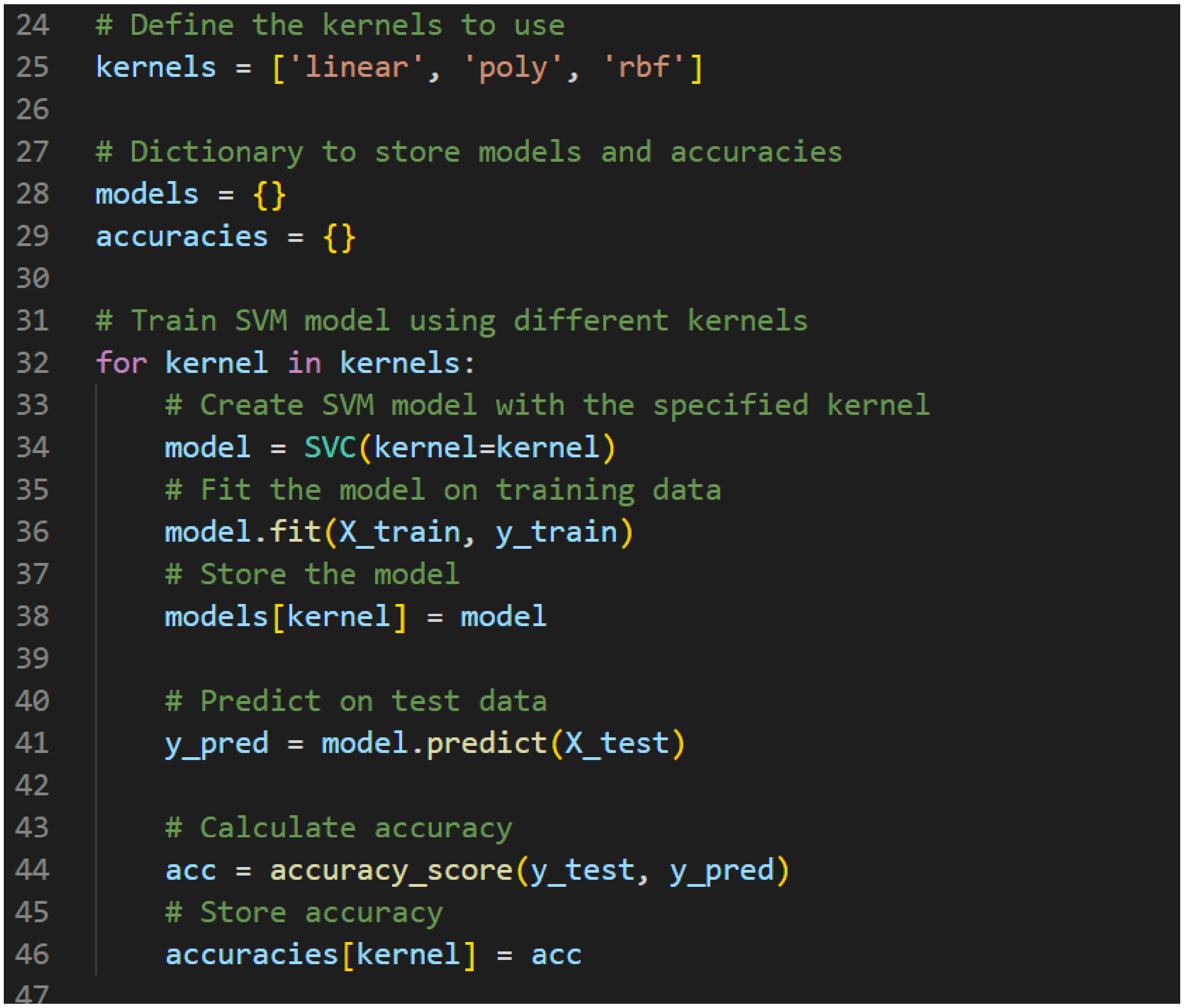
```python
# Split data into training and testing sets
# Use test_size=0.3 as specified, and random_state=42 for reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Summary of Model Implementation

Since SVM can benefit from scaling, but for the Iris dataset, features are on similar scales, scaling was not applied in this implementation.

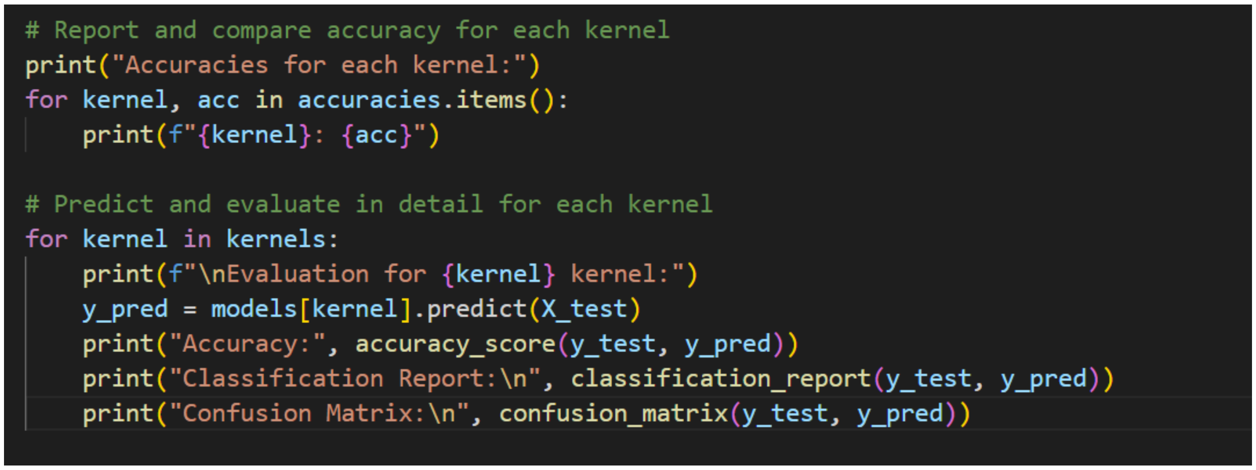
A Support Vector Machine (SVM) model from sklearn.svm was chosen for this multiclass classification task. Models were trained using the training data (105 samples) with different kernels: linear, poly, and rbf.

SVM classifies by finding the hyperplane that best separates classes, with kernels allowing for non-linear boundaries.



```python
# Define the kernels to use
kernels = ['linear', 'poly', 'rbf']

# Dictionary to store models and accuracies
models = {}
accuracies = {}

# Train SVM model using different kernels
for kernel in kernels:
    # Create SVM model with the specified kernel
    model = SVC(kernel=kernel)
    # Fit the model on training data
    model.fit(X_train, y_train)
    # Store the model
    models[kernel] = model

    # Predict on test data
    y_pred = model.predict(X_test)

    # Calculate accuracy
    acc = accuracy_score(y_test, y_pred)
    # Store accuracy
    accuracies[kernel] = acc

```

Predictions were made on the test data, and performance metrics were calculated.



```python
# Report and compare accuracy for each kernel
print("Accuracies for each kernel:")
for kernel, acc in accuracies.items():
    print(f"{kernel}: {acc}")

# Predict and evaluate in detail for each kernel
for kernel in kernels:
    print(f"\nEvaluation for {kernel} kernel:")
    y_pred = models[kernel].predict(X_test)
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Model Accuracies:

• Linear kernel: 1.0

• Poly kernel: 0.9778

• RBF kernel: 1.0

Classification Report (for Linear kernel):

precision recall f1-score support

Setosa 1.00 1.00 1.00 19

Versicolor 1.00 1.00 1.00 13

Virginica 1.00 1.00 1.00 13

accuracy 1.00 45

macro avg 1.00 1.00 1.00 45

weighted avg 1.00 1.00 1.00 45

Confusion Matrix (for Linear kernel):

[[19 0 0]

[0 13 0]

[0 0 13]]

Classification Report (for Poly kernel):

precision recall f1-score support

Setosa 1.00 1.00 1.00 19

Versicolor 1.00 0.92 0.96 13

Virginica 0.93 1.00 0.96 13

accuracy 0.98 45

macro avg 0.98 0.97 0.97 45

weighted avg 0.98 0.98 0.98 45

Confusion Matrix (for Poly kernel):

[[19 0 0]

[0 12 1]

[0 0 13]]

Classification Report (for RBF kernel):

precision recall f1-score support

Setosa 1.00 1.00 1.00 19

Versicolor 1.00 1.00 1.00 13

Virginica 1.00 1.00 1.00 13

accuracy 1.00 45

macro avg 1.00 1.00 1.00 45

weighted avg 1.00 1.00 1.00 45

Confusion Matrix (for RBF kernel):

[[19 0 0]

[0 13 0]

[0 0 13]]

Interpretation and Conclusion

The models perform excellently, with linear and rbf kernels achieving perfect accuracy, indicating the Iris dataset is linearly separable in some transformations. The poly kernel is slightly lower at 97.78%, with one misclassification from Versicolor to Virginica. Petal measurements likely influence separation strongly. The SVM classifier achieved near-perfect accuracy on the Iris dataset. This demonstrates that SVM is highly suitable for classification tasks with clear separability. Compared to KNN, SVM can handle higher dimensions better and is effective with kernels for non-linear data. In real-world applications, SVM's robustness is an advantage, but it can be computationally intensive for large datasets. Further improvements could include tuning hyperparameters like C or gamma. This experiment shows SVM's effectiveness with structured, small datasets.