```
# Lab 10: SVM Algorithm implementation
# 0. Installation and Imports

# Uncomment to install ucimlrepo if not installed
!pip install ucimlrepo

from ucimlrepo import fetch_ucirepo
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
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.12/dist-packages (0.0.7)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2025.1
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrep
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlre
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->panda
```

```
# Lab 10: SVM Algorithm implementation
# 1. Load Dataset and Inspect

balance_scale = fetch_ucirepo(id=12)
X = balance_scale.data.features
y = balance_scale.data.targets

print("Metadata:\n", balance_scale.metadata)
print("\nVariable Information:\n", balance_scale.variables)
print("Sample features:\n", X.head())
print("Sample targets:\n", y.head())
```

```
Metadata:
 {'uci_id': 12, 'name': 'Balance Scale', 'repository_url': 'https://archive.ics.uci.edu/dataset/12/balance+scale', 'da

Variable Information:
              name     role         type demographic    description units  \
0  right-distance  Feature  Categorical        None        L, B, R  None
1    right-weight  Feature  Categorical        None  1, 2, 3, 4, 5  None
2   left-distance  Feature  Categorical        None  1, 2, 3, 4, 5  None
3     left-weight  Feature  Categorical        None  1, 2, 3, 4, 5  None
4           class   Target  Categorical        None  1, 2, 3, 4, 5  None

  missing_values
0             no
1             no
2             no
3             no
4             no
Sample features:
    right-distance  right-weight  left-distance  left-weight
0               1             1              1            1
1               2             1              1            1
2               3             1              1            1
3               4             1              1            1
4               5             1              1            1
Sample targets:
   class
0     B
1     R
2     R
3     R
4     R
```

```
# Lab 10: SVM Algorithm implementation
# 2. Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(
    X, y.values.ravel(), test_size=0.2, random_state=42
)

print("Training set size:", X_train.shape)
print("Testing set size:", X_test.shape)
```

```
Training set size: (500, 4)
Testing set size: (125, 4)
```

```
# Lab 10: SVM Algorithm implementation
# 3. SVM Training and Prediction

svm_model = SVC(kernel='rbf', C=1.0, random_state=42)  # You can try 'linear', 'poly', or 'sigmoid' kernels
svm_model.fit(X_train, y_train)
y_pred = svm_model.predict(X_test)
```

```
# Lab 10: SVM Algorithm implementation
# 4. Evaluation

print("Accuracy Score:", 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))
```

```
Accuracy Score: 0.904
Classification Report:
              precision    recall  f1-score   support

           B       0.00      0.00      0.00        11
           L       0.87      1.00      0.93        55
           R       0.94      0.98      0.96        59

    accuracy                           0.90       125
   macro avg       0.60      0.66      0.63       125
weighted avg       0.83      0.90      0.86       125

Confusion Matrix:
 [[ 0  7  4]
 [ 0 55  0]
 [ 0  1 58]]
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
# Lab 10: SVM Algorithm implementation
# 5. Confusion Matrix Visualization

from sklearn.metrics import ConfusionMatrixDisplay

cm = confusion_matrix(y_test, y_pred)
ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=svm_model.classes_).plot(cmap="Blues")
plt.title("SVM Confusion Matrix")
plt.show()
```