Experiment No : 7 **Roll no :**

Aim : To study and implement support vector machine(SVM) in machine learning.

Theory :.

Support Vector Machine (SVM) – Theory for Study and Implementation

Introduction

Support Vector Machine (SVM) is a **supervised machine learning algorithm** primarily used for **classification tasks**, although it can also handle **regression** problems (SVR). SVMs are powerful, especially in high-dimensional spaces, and are known for their ability to create **robust decision boundaries**.

Core Idea

The main goal of SVM is to **find a hyperplane** that best separates the data points of different classes. In a 2D space, this hyperplane is simply a line. In higher dimensions, it becomes a plane or hyperplane.

Let's say we have a binary classification task with classes labeled as `+1` and `-1`. The SVM tries to:

- Maximize the **margin** between the two classes.
- Choose the hyperplane that is **equidistant** from the nearest data points of each class (called **support vectors**).

Margin and Support Vectors

- **Margin**: The distance between the hyperplane and the nearest data points from either class.
- **Support Vectors**: The data points that lie closest to the decision boundary. These points are critical in defining the optimal hyperplane.

The intuition is: a **larger margin** implies better generalization on unseen data.

Mathematical Formulation

For a linearly separable dataset, SVM solves the following optimization problem:

Objective:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

Subject to:

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

Where:

- xix_i are feature vectors
- yi∈{−1,+1}y_i \in \{-1, +1\} are labels
- ww is the weight vector
- bb is the bias

This is a **convex quadratic optimization problem** and has a unique global minimum.

Advantages of SVM

1. **Effective in high-dimensional spaces**.
2. **Works well for clear margin of separation**.
3. **Robust to overfitting**, especially with proper regularization (C).
4. **Versatile**: Can be used for both linear and non-linear classification using kernels.

Disadvantages of SVM

1. **Not suitable for very large datasets** (training time is long).
2. **Performance is sensitive** to choice of kernel and hyperparameters.
3. **Less interpretable** than simpler models like decision trees or logistic regression.
4. **Requires feature scaling** (standardization or normalization).

Implementation Steps

1. **Load and preprocess data** (handle missing values, encode labels).
2. **Split dataset** into training and testing sets.
3. **Standardize features** (important for SVM performance).
4. **Train SVM** with chosen kernel and parameters.
5. **Evaluate model** using accuracy, confusion matrix, etc.
6. **Tune hyperparameters** (e.g., `C`, `gamma`) using cross-validation.

Real-World Applications

- **Image classification** (e.g., face recognition)
- **Text classification** (e.g., spam filtering)
- **Bioinformatics** (e.g., protein classification)

Source Code :

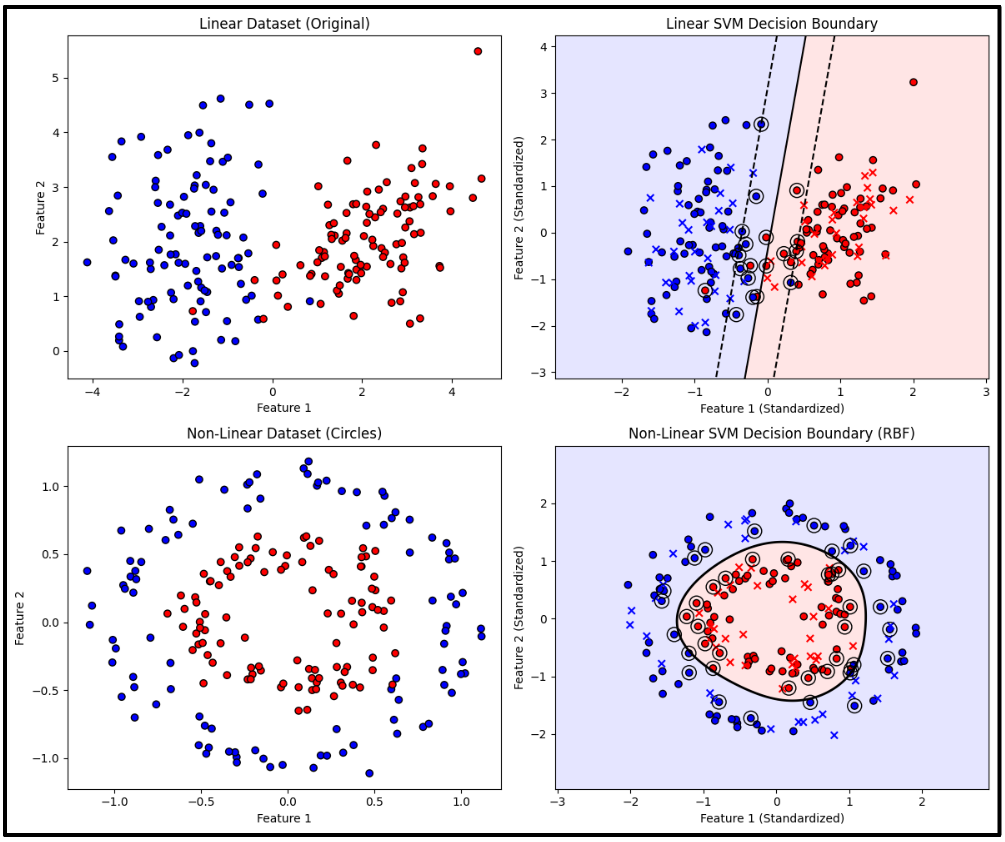
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification, make_circles
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# ---------------------------
#  Linear Dataset
```

```
# ---------------------------
X_lin, y_lin = make_classification(
    n_samples=200, n_features=2, n_informative=2, n_redundant=0,
    n_clusters_per_class=1, class_sep=2.0, random_state=42
)

X_train_lin, X_test_lin, y_train_lin, y_test_lin = train_test_split(
    X_lin, y_lin, test_size=0.3, random_state=42, stratify=y_lin
)
sc_lin = StandardScaler()
X_train_lin_std = sc_lin.fit_transform(X_train_lin)
X_test_lin_std = sc_lin.transform(X_test_lin)

svc_linear = SVC(kernel="linear", C=1.0)
svc_linear.fit(X_train_lin_std, y_train_lin)

xx_lin, yy_lin = np.meshgrid(
    np.linspace(X_train_lin_std[:,0].min()-1,
X_train_lin_std[:,0].max()+1, 200),
    np.linspace(X_train_lin_std[:,1].min()-1,
X_train_lin_std[:,1].max()+1, 200)
)
Z_lin = svc_linear.decision_function(np.c_[xx_lin.ravel(),
yy_lin.ravel()])
Z_lin = Z_lin.reshape(xx_lin.shape)

# ---------------------------
# Non-Linear Dataset (make_circles)
# ---------------------------
X_non, y_non = make_circles(n_samples=200, noise=0.1, factor=0.5,
random_state=42)

X_train_non, X_test_non, y_train_non, y_test_non = train_test_split(
    X_non, y_non, test_size=0.3, random_state=42, stratify=y_non
)
sc_non = StandardScaler()
X_train_non_std = sc_non.fit_transform(X_train_non)
X_test_non_std = sc_non.transform(X_test_non)
```

```
svc_rbf = SVC(kernel="rbf", gamma=1, C=1)
svc_rbf.fit(X_train_non_std, y_train_non)

xx_non, yy_non = np.meshgrid(
    np.linspace(X_train_non_std[:,0].min()-1,
X_train_non_std[:,0].max()+1, 200),
    np.linspace(X_train_non_std[:,1].min()-1,
X_train_non_std[:,1].max()+1, 200)
)
Z_non = svc_rbf.decision_function(np.c_[xx_non.ravel(),
yy_non.ravel()])
Z_non = Z_non.reshape(xx_non.shape)

# ---------------------------
#  Plotting (4 sections)
# ---------------------------
fig, axes = plt.subplots(2, 2, figsize=(12, 10))

# Linear dataset (original)
axes[0,0].scatter(X_lin[:,0], X_lin[:,1], c=y_lin, cmap=plt.cm.bwr,
edgecolors='k')
axes[0,0].set_title("Linear Dataset (Original)")
axes[0,0].set_xlabel("Feature 1")
axes[0,0].set_ylabel("Feature 2")



# Non-linear dataset (SVM boundary)
axes[1,1].contourf(xx_non, yy_non, Z_non > 0, alpha=0.2,
cmap=plt.cm.bwr)
axes[1,1].contour(xx_non, yy_non, Z_non, levels=[0], colors='k',
linewidths=2)
axes[1,1].scatter(X_train_non_std[:,0], X_train_non_std[:,1],
c=y_train_non, cmap=plt.cm.bwr, edgecolors='k', marker='o')
axes[1,1].scatter(X_test_non_std[:,0], X_test_non_std[:,1],
c=y_test_non, cmap=plt.cm.bwr, edgecolors='k', marker='x')
axes[1,1].scatter(svc_rbf.support_vectors_[:,0],
svc_rbf.support_vectors_[:,1], s=150, facecolors='none',
edgecolors='k')
axes[1,1].set_title("Non-Linear SVM Decision Boundary (RBF)")
```

```
axes[1,1].set_xlabel("Feature 1 (Standardized)")
axes[1,1].set_ylabel("Feature 2 (Standardized)")

plt.tight_layout()
plt.show()
```

Output :



Conclusion : We have successfully implemented SVM in machine learning.