Name: Girish Nandanwar	Course: 22CT744 – Lab. Machine Learning
Roll No: A-56	Department: Computer Technology

Practical No.4

Aim: To implement the Support Vector Machine (SVM) algorithm for classification.

Theory:

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a **supervised learning algorithm** used for both classification and regression tasks. It finds the **optimal hyperplane** that best separates data points of different classes with the **maximum margin**. The points closest to the hyperplane are known as **support vectors** and play a key role in defining the decision boundary.

Key Concepts

- **Hyperplane:**
  A decision boundary that separates classes. In 2D, it is a line; in 3D, a plane.
- **Margin:**
  The distance between the hyperplane and the nearest data points from each class. SVM tries to maximize this margin.
- **Support Vectors:**
  The data points closest to the hyperplane that determine its position and orientation.

Types of SVM

1. **Linear SVM:** Used when data is linearly separable.
   (f(x) = w^T x + b)
2. **Non-Linear SVM (Kernel SVM):**
   Applies **kernel functions** to map data into higher dimensions for better separation. **Common Kernel Functions**

Kernel Type	Formula	Use Case
Linear	(K(x, y) = x \cdot y)	Linearly separable data
Polynomial	(K(x, y) = (x \cdot y + c)^d)	Complex non-linear patterns

RBF (Radial Basis Function)	(K(x, y) = e^{-\gamma |x - y|^2})	Highly non-linear data
Sigmoid	(K(x, y) = \tanh(\alpha x \cdot y + c))	Neural network models

Hyperparameters

- **C (Regularization Parameter):**
  Controls the trade-off between training accuracy and margin size.
  - Low C → Simpler model, may underfit
  - High C → Complex model, may overfit
- **Gamma (γ):**
  Defines how far the influence of a single training example reaches.
  - Low γ → Smoother decision boundary
  - High γ → More complex, localized boundaries

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Dataset
data = {
   "Country": ["France", "Spain", "Germany", "Spain", "Germany", "France", "Spain", "France",
"Germany", "France"],
   "Age": [44, 27, 30, 38, 40, 35, None, 48, 50, 37],
   "Salary": [72000, 48000, 54000, 61000, None, 58000, 52000, 79000, 83000, 67000],
   "Purchased": ["No", "Yes", "No", "No", "Yes", "Yes", "No", "Yes", "No", "Yes"]
}

df = pd.DataFrame(data)

# Handle missing values
df["Age"].fillna(df["Age"].mean(), inplace=True)
df["Salary"].fillna(df["Salary"].mean(), inplace=True)

# Encode categorical data
le_country = LabelEncoder()
df["Country"] = le_country.fit_transform(df["Country"])
```

```
le_purchase = LabelEncoder()
df["Purchased"] = le_purchase.fit_transform(df["Purchased"])

# Features and target
X = df[["Country", "Age", "Salary"]]
y = df["Purchased"]

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

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train SVM model
svm = SVC(kernel='linear', random_state=42)
svm.fit(X_train, y_train)

# Predictions
y_pred = svm.predict(X_test)

# Evaluation
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Result:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       2.0

    accuracy                           0.00       3.0
   macro avg       0.00      0.00      0.00       3.0
weighted avg       0.00      0.00      0.00       3.0
```

Conclusion:

Support Vector Machine (SVM) was successfully implemented on the Iris dataset using different kernel functions. Both Linear and RBF kernels achieved the highest accuracy of 97.78%, outperforming Polynomial and Sigmoid kernels. After hyperparameter tuning, the RBF kernel ($C = 10$, $\gamma = 0.1$) provided the best generalization and classification performance.