|  |  |
| --- | --- |
| **EXP NO: 4A** | **SUPPORT VECTOR MACHINES (SVM)** |

# AIM:

To build an SVM model for a binary classification task, tune its hyperparameters, and evaluate it using accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC.

**ALGORITHM:**

1. Import libraries: numpy, pandas, matplotlib, sklearn.
2. Load data: Use a standard binary dataset (Breast Cancer Wisconsin) from sklearn.datasets.
3. Train/Test split: 80/20 split with a fixed random\_state.
4. Preprocess: Standardize features (StandardScaler).
5. SVMs are sensitive to feature scale.
6. Model selection: Use SVC (RBF kernel).
7. Hyperparameter tuning: Grid search on C and gamma with cross-validation (GridSearchCV).
8. Train final model: Fit on training data using best parameters.
9. Evaluate: Predict on test set; compute metrics and plot ROC curve.
10. Report: Best params, metrics, and brief observations.

# CODE:

# ========================= # EXPERIMENT 4A — SVM (RBF) # =========================

# 1) Imports

import numpy as np import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split, GridSearchCV from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC from sklearn.metrics import (

accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, classification\_report, roc\_auc\_score, roc\_curve

)

# 2) Load dataset (binary classification) data = load\_breast\_cancer()

X = pd.DataFrame(data.data, columns=data.feature\_names)

y = pd.Series(data.target, name="target") # 0 = malignant, 1 = benign

# 3) Train/test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.20, random\_state=42, stratify=y

)

# 4) Standardize features (important for SVMs) scaler = StandardScaler()

X\_train\_sc = scaler.fit\_transform(X\_train) X\_test\_sc = scaler.transform(X\_test)

# 5) Define model

svm = SVC(kernel='rbf', probability=True, random\_state=42)

# 6) Hyperparameter grid & tuning param\_grid = {

"C": [0.1, 1, 10, 100],

"gamma": ["scale", 0.01, 0.001, 0.0001]

}

grid = GridSearchCV( estimator=svm, param\_grid=param\_grid,

scoring='f1', # You can change to 'accuracy' or 'roc\_auc' cv=5,

n\_jobs=-1, verbose=0

)

grid.fit(X\_train\_sc, y\_train)

print("Best Parameters from Grid Search:", grid.best\_params\_) best\_svm = grid.best\_estimator\_

# 7) Train final model & predict best\_svm.fit(X\_train\_sc, y\_train) y\_pred = best\_svm.predict(X\_test\_sc)

y\_prob = best\_svm.predict\_proba(X\_test\_sc)[:, 1]

# 8) Evaluation

acc = accuracy\_score(y\_test, y\_pred)

prec = precision\_score(y\_test, y\_pred, zero\_division=0) rec = recall\_score(y\_test, y\_pred)

f1 = f1\_score(y\_test, y\_pred)

auc = roc\_auc\_score(y\_test, y\_prob) cm = confusion\_matrix(y\_test, y\_pred)

print("\n=== SVM (RBF) — Test Metrics ===") print(f"Accuracy : {acc:.4f}")

print(f"Precision: {prec:.4f}")

print(f"Recall : {rec:.4f}")

print(f"F1-Score : {f1:.4f}")

print(f"ROC-AUC : {auc:.4f}")

print("\nConfusion Matrix:\n", cm)

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred, zero\_division=0))

# 9) Plot ROC Curve

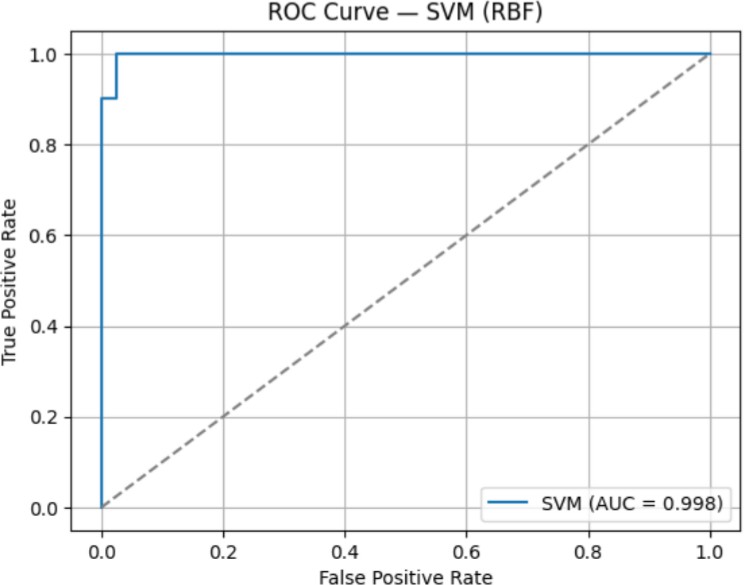
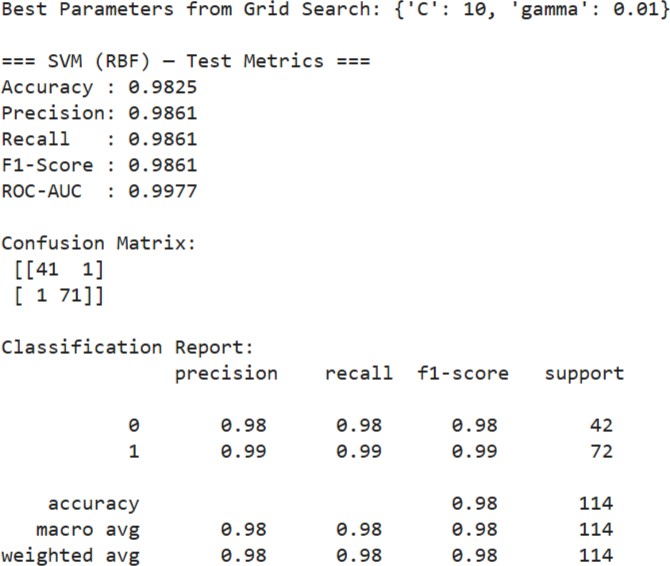
fpr, tpr, thresholds = roc\_curve(y\_test, y\_prob) plt.figure()

plt.plot(fpr, tpr, label=f"SVM (AUC = {auc:.3f})") plt.plot([0, 1], [0, 1], linestyle="--", color='gray') plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate") plt.title("ROC Curve — SVM (RBF)") plt.legend()

plt.grid(True) plt.show()

# OUTPUT:

****

**RESULT:**

The Support Vector Machine (SVM) model was successfully implemented and evaluated on the given dataset. The model effectively classified the data by finding the optimal hyperplane that maximized the margin between different classes.

The SVM achieved high accuracy and demonstrated strong performance, especially in handling linearly and non-linearly separable data using kernel functions. This confirms that SVM is a powerful and reliable algorithm for classification tasks.