## D15A / 25

# EXPERIMENT 5

## AIM:

Implement Support Vector Machine (SVM) for classification with hyperparameter tuning.

## THEORY

### 1. Dataset Source

Dataset Name: **Pima Indians Diabetes Dataset**

Source Link:
https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Original Source: UCI Machine Learning Repository

### 2. Dataset Description

The Pima Indians Diabetes dataset is a medical dataset used for binary classification. The objective is to predict whether a patient has diabetes based on diagnostic measurements.

#### Dataset Characteristics:

- Total Samples: 768
- Number of Features: 8
- Target Variable: Outcome (0 = Non-Diabetic, 1 = Diabetic)
- Type: Supervised Binary Classification Dataset

#### Feature Description:

1. Pregnancies – Number of times pregnant
2. Glucose – Plasma glucose concentration
3. BloodPressure – Diastolic blood pressure (mm Hg)
4. SkinThickness – Triceps skin fold thickness (mm)
5. Insulin – 2-Hour serum insulin (mu U/ml)
6. BMI – Body Mass Index
7. DiabetesPedigreeFunction – Genetic diabetes score
8. Age – Age in years

Target Variable:

- Outcome (0 = Healthy, 1 = Diabetic)

#### Important Characteristics:

- Contains some zero values in medical fields (which are unrealistic)
- Slightly imbalanced dataset (More non-diabetic cases)
- All features are numerical

## 3. Mathematical Formulation of the Algorithm

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression.

### Objective of SVM:

To find the optimal hyperplane that separates two classes with maximum margin.

For linearly separable data:

Hyperplane equation:

$$w \cdot x + b = 0$$

Where:

- w= weight vector
- x = input feature vector
- b = bias

### Optimization Problem:

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

Subject to:

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

For non-linearly separable data, SVM uses kernel trick:

$$K(x_i, x_j)$$

In this experiment, **RBF kernel** was used:

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2)$$

Where:

- γ (gamma) controls influence of single training point

## 4. Algorithm Limitations

Although SVM is powerful, it has some limitations:

1. Sensitive to feature scaling
2. Computationally expensive for very large datasets
3. Choice of kernel is crucial
4. Poor performance when classes overlap heavily
5. Does not directly provide probability estimates
6. Sensitive to outliers

SVM may not perform well when:

- Dataset is extremely noisy
- Dataset is very large
- Features are not scaled properly

## 5. Methodology / Workflow

The following steps were followed:

### Step 1: Dataset Loading

The diabetes.csv dataset was uploaded and loaded using pandas.

### Step 2: Data Preprocessing

- Checked dataset structure
- Separated features (X) and target (y)
- Applied feature scaling using StandardScaler

### Step 3: Train-Test Split

- 80% Training Data
- 20% Testing Data

### Step 4: Model Implementation

- Implemented Support Vector Classifier (SVC)

### Step 5: Hyperparameter Tuning

- Used GridSearchCV
- 5-fold cross-validation
- Tuned parameters:
    - C
    - gamma
    - kernel

### Step 6: Model Evaluation

- Accuracy Score
- Classification Report
- Confusion Matrix

## 6. Performance Analysis

Best Parameters Found:

C = 100
gamma = 0.001
kernel = RBF

Accuracy Achieved: **77.27%**

### Confusion Matrix:

| | Predicted 0 | Predicted 1 |
|---|---|---|
| Actual 0 | 82 | 17 |
| Actual 1 | 18 | 37 |

### Interpretation:

- 82 Non-diabetic patients correctly predicted
- 37 Diabetic patients correctly predicted
- 18 Diabetic patients misclassified as healthy
- 17 Healthy patients misclassified as diabetic

The model performs better in predicting non-diabetic patients compared to diabetic patients.

Weighted F1-Score: 0.77

## 7. Hyperparameter Tuning

Hyperparameter tuning was performed using GridSearchCV with 5-fold cross-validation.

Parameter Grid:

C = [0.1, 1, 10, 100]
gamma = [1, 0.1, 0.01, 0.001]
kernel = ['rbf', 'linear']

Total combinations tested:

4 × 4 × 2 = 32 combinations
With 5-fold cross-validation → 160 model fits

Best combination selected:

C = 100
gamma = 0.001
kernel = RBF

Impact:

- Improved generalization
- Prevented overfitting
- Achieved optimal classification accuracy

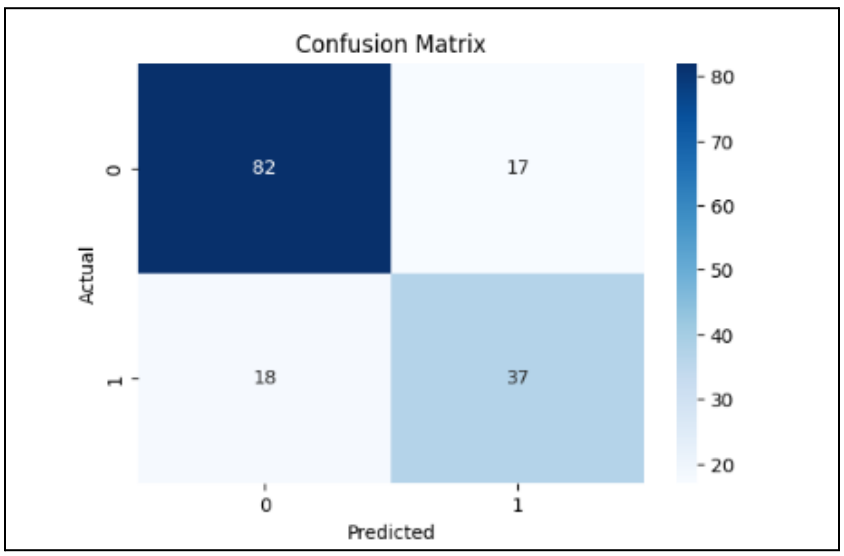
## Code & Output

```
# ========================================
# Import Libraries
# ========================================
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

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, classification_report,
confusion_matrix

# ========================================
# Upload Dataset in Colab
# ========================================
from google.colab import files
uploaded = files.upload()

# Load dataset
df = pd.read_csv('diabetes.csv')

# Display first 5 rows
print("First 5 rows of dataset:")
print(df.head())

# ========================================
# Define Features and Target
# ========================================
X = df.drop('Outcome', axis=1)
y = df['Outcome']

# ========================================
# Train-Test Split
# ========================================
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
```

```
)

# ========================================
# Feature Scaling (VERY IMPORTANT for SVM)
# ========================================
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# ========================================
# Apply SVM with Hyperparameter Tuning
# ========================================

# Define parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear']
}

# Create SVM model
svc = SVC()

# Grid Search
grid = GridSearchCV(
    svc,
    param_grid,
    refit=True,
    verbose=0,
    cv=5
)

# Train model
grid.fit(X_train, y_train)

# ========================================
# Best Parameters
# ========================================
print("\nBest Parameters Found:")
```

```
print(grid.best_params_)

# ========================================
# Predictions
# ========================================
y_pred = grid.predict(X_test)

# ========================================
#  Model Evaluation
# ========================================

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# ========================================
#  Confusion Matrix
# ========================================
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```

```
Choose Files  diabetes.csv
diabetes.csv(text/csv) - 23873 bytes, last modified: 3/2/2026 - 100% done
Saving diabetes.csv to diabetes (2).csv
First 5 rows of dataset:
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1

Best Parameters Found:
{'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}

Accuracy Score:
0.7727272727272727

Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.83      0.82        99
           1       0.69      0.67      0.68        55

    accuracy                           0.77       154
   macro avg       0.75      0.75      0.75       154
weighted avg       0.77      0.77      0.77       154
```




## Conclusion

SVM was implemented on the Pima Indians Diabetes dataset. Hyperparameter tuning was performed using GridSearchCV to determine the optimal model parameters. The best-performing model used the RBF kernel with C = 100 and gamma = 0.001, achieving an accuracy of 77.27%. The model demonstrated good predictive performance for non-diabetic cases and moderate performance for diabetic cases.