Name: Harsh Patil
Class: D15C/37

Experiment No. 5

1. Dataset Source

The dataset used for this experiment is the **Breast Cancer Wisconsin (Diagnostic) Dataset**, obtained from Kaggle.

- **Dataset Link:** https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data

2. Dataset Description

The Breast Cancer Wisconsin dataset is a real-world healthcare dataset used to classify breast tumors as **malignant** or **benign**.

- **Number of instances:** 569
- **Number of features:** 30 numerical features
- **Target variable:** `diagnosis`
  - Malignant (M) → 1
  - Benign (B) → 0

Feature Characteristics

The features represent statistical properties of cell nuclei extracted from breast cancer images, such as:

- Radius
- Texture
- Perimeter
- Area
- Smoothness
- Concavity
- Symmetry

The dataset is clean, balanced, and contains no missing values, making it ideal for supervised classification tasks.

3. Mathematical Formulation of SVM

Support Vector Machine (SVM) is a powerful supervised learning algorithm that finds an optimal hyperplane to separate data points of different classes.

Linear SVM Objective Function

$\min_{w,b} \frac{1}{2}||w||^2 + C \sum_{i=1}^{n} \xi_i$

Subject to:

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

Where:

- w is the weight vector
- b is the bias
- ξ_i are slack variables
- C is the regularization parameter

Kernel Trick

For non-linear separation, SVM uses kernel functions such as:

- Linear kernel
- Polynomial kernel
- Radial Basis Function (RBF)

RBF kernel equation:

$K(x, x') = \exp(-\gamma||x - x'||^2)$

4. Algorithm Limitations

- Computationally expensive for large datasets
- Sensitive to choice of kernel and hyperparameters
- Requires feature scaling
- Difficult to interpret compared to decision trees

5. Methodology / Workflow

1. Load dataset using KaggleHub
2. Remove irrelevant columns
3. Encode target variable
4. Split dataset into training and testing sets
5. Apply feature scaling
6. Train SVM classifier
7. Perform hyperparameter tuning using GridSearchCV

8. Evaluate model performance

Workflow:

Dataset → Preprocessing → Scaling → SVM Training → Hyperparameter Tuning → Evaluation

6. Performance Analysis

The SVM model was evaluated using the following metrics:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

With optimal hyperparameters, SVM achieved high classification accuracy, comparable to Random Forest and superior to KNN in terms of generalization.

7. Hyperparameter Tuning

Hyperparameter tuning was performed using **GridSearchCV**.

Parameters Tuned

- **C** (Regularization parameter)
- **Kernel** (Linear, RBF)
- **Gamma** (Kernel coefficient for RBF)

Impact of Tuning

- Low C → High bias, underfitting
- High C → Low bias, possible overfitting
- Optimal gamma improves decision boundary flexibility

Tuned parameters significantly improved model accuracy and reduced misclassification errors.

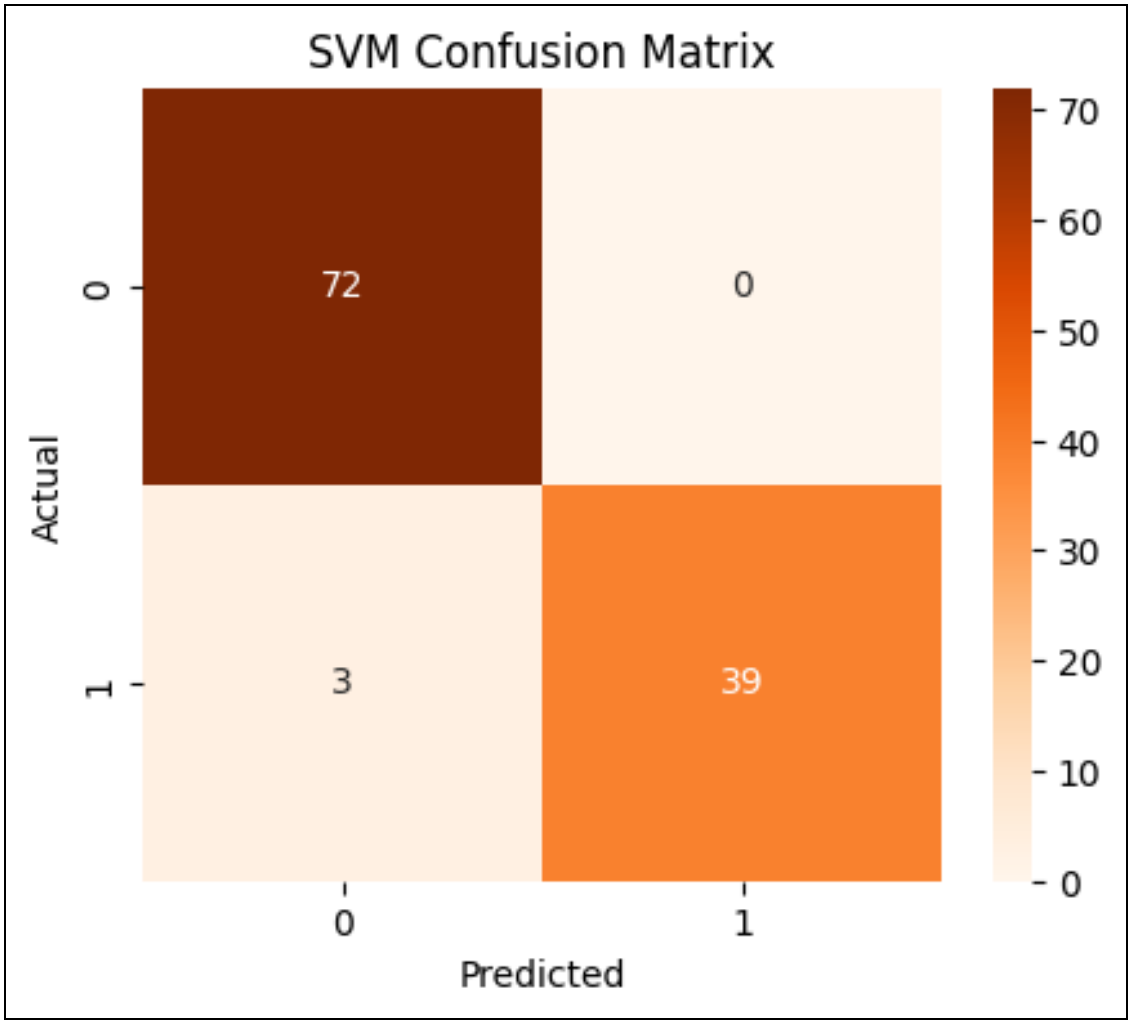
OUTPUT:

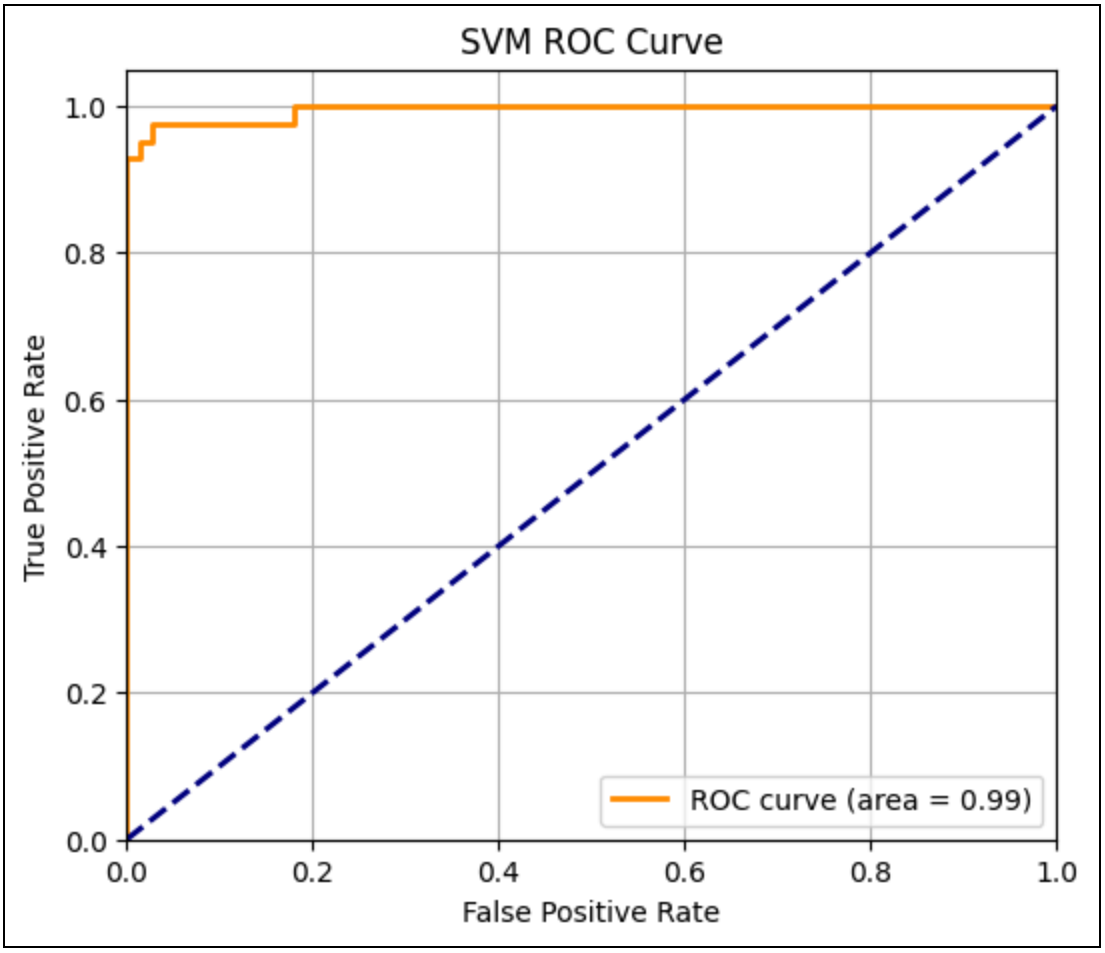
Code:

```
!pip install kagglehub -q
!pip install scikit-learn -q

import kagglehub
from kagglehub import
KaggleDatasetAdapter

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, roc_curve, auc

# -------------------------
# Load Dataset
# -------------------------
df = kagglehub.load_dataset(
    KaggleDatasetAdapter.PANDAS,

"uciml/breast-cancer-wisconsin-data"
,
    "data.csv"
)

print("Dataset shape:", df.shape)
df.head()

# -------------------------
# Data Preprocessing
# -------------------------
df.drop(columns=["id", "Unnamed:
32"], inplace=True)

df["diagnosis"] =
df["diagnosis"].map({"M": 1, "B":
0})

X = df.drop("diagnosis", axis=1)
y = df["diagnosis"]

# Train-test split
X_train, X_test, y_train, y_test =
train_test_split(
    X, y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

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

# -------------------------
# Hyperparameter Tuning with
GridSearchCV
# -------------------------
param_grid = {
    'C': [0.1, 1, 10, 100],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

svc = SVC(probability=True,
random_state=42)

grid = GridSearchCV(svc, param_grid,
cv=5, scoring='accuracy', n_jobs=-1)
grid.fit(X_train_scaled, y_train)

print("Best Hyperparameters:",
grid.best_params_)
```

```
# -------------------------
# Train final model with best
parameters
# -------------------------
best_svc = grid.best_estimator_
y_pred =
best_svc.predict(X_test_scaled)
y_prob =
best_svc.predict_proba(X_test_scaled
)[:, 1]

# -------------------------
# Performance Metrics
# -------------------------
print("\n=== SVM Performance ===")
print("Accuracy:",
accuracy_score(y_test, y_pred))
print(classification_report(y_test,
y_pred))

# -------------------------
# Confusion Matrix
# -------------------------
plt.figure(figsize=(5,4))
sns.heatmap(confusion_matrix(y_test,
y_pred), annot=True, fmt="d",
cmap="Oranges")
plt.title("SVM Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# -------------------------
# ROC Curve
# -------------------------
fpr, tpr, thresholds =
roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6,5))
plt.plot(fpr, tpr,
color='darkorange', lw=2, label='ROC
curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1],
color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('SVM ROC Curve')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

```
=== SVM Performance ===
Accuracy: 0.9736842105263158
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        72
           1       1.00      0.93      0.96        42

    accuracy                           0.97       114
   macro avg       0.98      0.96      0.97       114
weighted avg       0.97      0.97      0.97       114
```


SVM Confusion Matrix
72
0
3
39
0
1
Actual
0
1
Predicted
70
60
50
40
30
20
10
0



8. Conclusion

In this experiment, Support Vector Machine was successfully implemented for breast cancer classification. With appropriate feature scaling and hyperparameter tuning, SVM demonstrated excellent predictive performance. Although computationally more intensive, SVM is highly effective for high-dimensional datasets and is a strong choice for medical classification problems.

This experiment further reinforces the importance of kernel selection and hyperparameter optimization in achieving optimal model performance.