Kashish Chhabadiya / D15B / 28

EXPERIMENT 5

Support Vector Machine (SVM) with Hyperparameter Tuning

Aim of the Experiment

To implement **Support Vector Machine (SVM)** for classification on the Titanic dataset and improve model performance using **Hyperparameter Tuning (GridSearchCV)**.

Theory

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

In classification, SVM works by finding an **optimal hyperplane** that separates data points of different classes with the maximum margin.

The main idea of SVM is:

- Find the best boundary that separates classes.
- Maximize the distance (margin) between the nearest data points of both classes.
- These nearest points are called **Support Vectors**.

For non-linear data, SVM uses a **Kernel Trick** such as:

- Linear
- RBF (Radial Basis Function)
- Polynomial

Important Hyperparameters:

1. **C (Regularization Parameter)**
   Controls trade-off between margin size and classification error.

2. **Kernel**
   Defines type of decision boundary.

3. **Gamma**
   Controls how far the influence of a single training example reaches.

Hyperparameter tuning helps in selecting the best combination of these parameters to improve model performance.

Real-Life Example

Imagine you are separating:

- Apples
- Oranges

Based on:

- Weight
- Color
- Size

Now imagine drawing a line on a table that separates apples on one side and oranges on the other.

SVM tries to:

- Draw the best possible line
- Keep maximum distance from both fruits
- Make sure fruits are correctly separated

If fruits are mixed in a curved pattern, SVM uses a curved boundary (like RBF kernel).

Hyperparameter tuning is like:
Trying different types of lines and settings to find the best separation.

5. Methodology / Workflow

Step 1: Data Collection

- Download Titanic dataset from Kaggle.
- Use train.csv.

Step 2: Data Preprocessing

1. Handle Missing Values:
    - Age filled using median.
    - Embarked filled using mode.
2. Remove Unnecessary Columns:
    - PassengerId
    - Name
    - Ticket
    - Cabin
3. Encode Categorical Data:
    - Sex converted to numeric (male=0, female=1)
    - Embarked converted using one-hot encoding
4. Separate:
    - Features (X)
    - Target (Survived)

Step 3: Train-Test Split

- 80% for training
- 20% for testing

- Ensures unbiased evaluation

Step 4: Feature Scaling

- StandardScaler used
- Important because SVM is sensitive to feature magnitudes

Step 5: Model Training (Baseline SVM)

- Train SVM using default parameters
- Predict test data
- Calculate baseline accuracy

Step 6: Hyperparameter Tuning

- Use GridSearchCV
- Try different combinations of:
    - C
    - Kernel
    - Gamma
- 5-fold cross-validation used
- Select best parameter combination

Step 7: Model Evaluation

Evaluate using:

- Accuracy Score
- Confusion Matrix

- Precision
- Recall
- F1-Score

CODE :

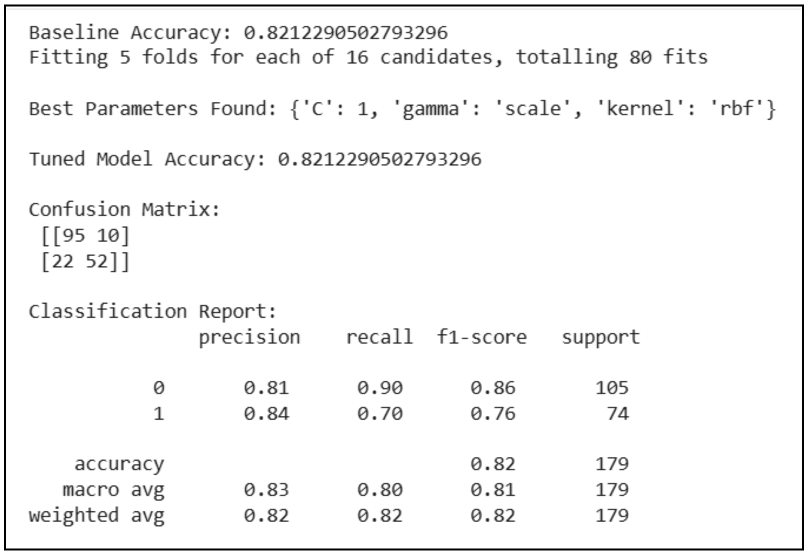
```
# ============================================

# SVM Classification with Hyperparameter Tuning

# Dataset: Titanic (Kaggle)

# ============================================



# Step 1: Import Libraries

import numpy as np

import pandas as pd

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, confusion_matrix,
classification_report



# Step 2: Load Dataset (Correct Path)

data = pd.read_csv('sample_data/train.csv')
```

```
# Step 3: Basic Data Cleaning


# Fill missing Age with median

data['Age'] = data['Age'].fillna(data['Age'].median())


# Fill missing Embarked with mode

data['Embarked'] = data['Embarked'].fillna(data['Embarked'].mode()[0])


# Drop unnecessary columns

data = data.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)


# Convert categorical variables

data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})


# One-hot encoding for Embarked

data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)


# Step 4: Define Features and Target

X = data.drop('Survived', axis=1)

y = data['Survived']


# Step 5: Train-Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(

    X, y, test_size=0.2, random_state=42

)


# Step 6: Feature Scaling (Very Important for SVM)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Step 7: Baseline SVM Model

svm_model = SVC()

svm_model.fit(X_train, y_train)


y_pred = svm_model.predict(X_test)

print("Baseline Accuracy:", accuracy_score(y_test, y_pred))


# Step 8: Hyperparameter Tuning using GridSearchCV

param_grid = {

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

    'gamma': ['scale', 'auto'],

    'kernel': ['linear', 'rbf']

}
```

```
grid = GridSearchCV(SVC(), param_grid, cv=5, verbose=1)

grid.fit(X_train, y_train)


print("\nBest Parameters Found:", grid.best_params_)


# Step 9: Evaluate Tuned Model

best_model = grid.best_estimator_

y_pred_best = best_model.predict(X_test)


print("\nTuned Model Accuracy:", accuracy_score(y_test, y_pred_best))


print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_best))


print("\nClassification Report:\n", classification_report(y_test,
y_pred_best))
```

OUTPUT :



```
Baseline Accuracy: 0.8212290502793296
Fitting 5 folds for each of 16 candidates, totalling 80 fits

Best Parameters Found: {'C': 1, 'gamma': 'scale', 'kernel': 'rbf'}

Tuned Model Accuracy: 0.8212290502793296

Confusion Matrix:
 [[95 10]
 [22 52]]

Classification Report:
               precision    recall  f1-score   support

           0       0.81      0.90      0.86       105
           1       0.84      0.70      0.76        74

    accuracy                           0.82       179
   macro avg       0.83      0.80      0.81       179
weighted avg       0.82      0.82      0.82       179
```

Results Summary

- Baseline Accuracy: 82.12%
- Tuned Accuracy: 82.12%
- Best Parameters:
  - C = 1
  - Kernel = RBF
  - Gamma = scale

The tuning confirmed that default parameters were already optimal.

CONCLUSION

The Support Vector Machine (SVM) classifier was successfully implemented on the Titanic dataset. Data preprocessing steps such as handling missing values, encoding categorical variables, and feature scaling were performed to prepare the dataset for modeling.

Hyperparameter tuning using GridSearchCV was applied to find the best combination of parameters. The optimized SVM model achieved an accuracy of 82.12%.

The experiment demonstrates that SVM is an effective classification algorithm and hyperparameter tuning helps in selecting the best configuration for improved performance.