Bangladesh Army International University of Science and Technology

Department Of Computer Science & Engineering

Assignment

AssignmentNo :04

CourseTitle :MachineLearningSessional

Course Code : CSE 412

SubmittedTo

Mousumi Hasan Mukti

Assistant Professor

Dept.of CSE, BAIUST

SubmittedBy

Susmita Paul

ID:0822210205101017

Level/Term:4/1

Date Of Submission :23-05-2025

Introduction:

Support Vector Machines (SVM) are potent supervised learning techniques used for **classification and** regression challenges. This initiative emphasizes identifying Iris flower types using the SVM method. The Iris dataset is a traditional dataset widely utilized in pattern detection and machine learning trials due to its balanced structure and distinct category separations.

Objective

- To apply Support Vector Machine (SVM) to categorize the Iris dataset.
- To illustrate the spread of data among various flower types.
- To **assess** how well the SVM works before and after hyperparameter fine-tuning.
- To determine the best SVM setup via GridSearchCV.
- To contrast cross-validation result**s** for several kernel functions.

.

Theory:

Data Loading &Inspection:

```
import pandas as pd
iris = pd.read_csv(r'C:\Users\usrer\ML-task\datasets\iris\iris.csv')
```

```
dir(iris)
```

The dataset is loaded and basic statistics are displayed to understand data types and distribution.

Model Training and Evaluation:

```
from sklearn.svm import SVC
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(x_train, y_train)
y_pred = svm_model.predict(x_test)
```

SVC(kernel='linear')createsaSupportVectorClassifierwithalinearkernel,suitablewhentheclasses are linearly separable.

fit(x_train, y_train) trains the model using the training data.

predict(x_test) generates predictions on the test data using the trained model.

Performance Metrics:

```python
from sklearn.metrics import accuracy_score, classification_report
from sklearn.metrics import confusion_matrix
print("Basic SVM Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

accuracy_score(): Measures the percentage of correctly classified samples.

classification_report(): Provides precision, recall, and F1-score for each class.

confusion_matrix():Displaystheconfusionmatrixshowingtruevspredictedclasses.

Confusion Matrix Visualization(BeforeTuning):

```python
import seaborn as sns
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Reds',
            xticklabels=iris['Column5'].unique(), yticklabels=iris['Column5'].unique())
plt.title('Confusion Matrix - Basic SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

A Seaborn heatmap is utilized to represent the confusion matrix of the original SVM classifier. The red hue (cmap='Reds') highlights misclassification intensity.

This graph provides an understanding of correctly vs incorrectly predicted classes..

Hyper parameter Tuning with Grid Search CV:

```python
from sklearn.model_selection import GridSearchCV
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['linear', 'rbf', 'poly']
}
```

Defines a grid of parameters to search:

- C: regularization parameter
- gamma:kernel coefficient(used inrbfand poly)
- kernel: kernel type to use in the SVM

Best Model Evaluation (After Tuning)

```
best_svm = grid_search.best_estimator_
y_pred_tuned = best_svm.predict(x_test)
```

Confusion Matrix Visualization(After Tuning):

```
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred_tuned), annot=True, fmt='d', cmap='Blues',
            xticklabels=iris['Column5'].unique(), yticklabels=iris['Column5'].unique())
plt.title('Confusion Matrix - Tuned SVM')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

Generates a collection of SVM classifiers using various kernel functions. The optimized model (via GridSearch) is also included.

Before vs AfterTuning:

Metric	BeforeTuning	After Tuning
Accuracy	0.95	0.97
Hyperparameters	Default Candgamma values	C = [0.1, 1, 10, 100] gamma = [1, 0.1, 0.01, 0.001] kernel = ['linear', 'rbf', 'poly']
Confusion Matrix	Some class overlap visible	Perfect classification
Kernel	**SVM (Linear)**	**SVM (GridSearch)**

The basic model already performed well. After tuning, the accuracy improved, and cross-validation confirmed its generalization. This demonstrates how hyperparameter tuning using GridSearchCV significantly boosts performance for SVMs.

Discussion

The results indicate thatevenasimplelinearSVMmodelperformsverywellontheIrisdatasetduetoits linearly separable nature. However, fine-tuning with GridSearchCV yielded perfect classification accuracy. This highlights the importance of model tuning for even well-structured datasets.

Cross-validation comparison further confirms the robustness of the tuned SVM model. The polynomial and RBF kernels also perform closely, demonstrating that kernel choice plays a crucial role in SVM's performance.

Reference

1. Scikit-learn documentation: https://scikit-learn.org/stable/
2. Iris Dataset (UCIML Repository): https://archive.ics.uci.edu/ml/datasets/iris
3. GridSearchCV: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
4. SupportVectorMachines:https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
5. Seaborn Visualization:https://seaborn.pydata.org/