CLASSMATE
Date :
Page :

Assignment 5

Name - Gaurang Patil
PRN - 1032221535
RollNo - 20
T.Y B.Tech CSF

Title → W.A.P to implement SVM classifier, compare with tree algorithm.

Aim → W.A.P to implement SVM classifier, compare with decision tree algorithm

objective → To study the SVM classifier and decision tree algo.

Theory →

1) Overview of (SVM) and Decision Trees as classifier algorithm.

→ (SVM) Support Vector Machines
Goal - Find the optimal hyperplane to separate classes
strengths - Handles High dimensional data effective with non-linear kernels
Weakness - Slow on large datasets, Requires careful tuning
Best for complex boundaries with Small to Medium dataset.

Decision Trees
Goal - Creates a tree structure to classify data feature split
strength - Easy to interpret, works with both Categorical and numerical data.
Weakness - Prone to overfitting, Sensitive to data change.

Scanned with OKEN Scanner

2) What is confusion matrix evaluation metrics used to compare the performance of the classifiers (e.g accuracy, f-score, recall);

Ans - A table that summarizes the performance of a classifier model by comparing predicted and actual labels.

	Predicted Positive	Predicted Negative.
Actual Positive	True Positive (TP	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Accuracy → Measures overall correctness, but may be misleading for imbalanced datasets.

Accuracy = (TP + TN) / (Total Samples)

Precision → Fraction of Relevant instances among the predicted positive

Precision = TP / (TP + FP)

Recall → Fraction of actual positive correctly identified

Recall = TP / (TP + FN)

F1-Score → Harmonic mean of Precision and Recall, useful for imbalanced dataset.

F1-Score = 2 x (Precision x Recall) / (Precision + Recall)

3) Explain Iris Dataset Explain how the dataset was divided into training and testing sets. Mention the train-test split ratio

→ The Iris dataset is a classic dataset used for classification tasks. It contains 150 samples of iris flowers with four

features namely Speal length, Spepal width, Petal length and Petal width Each Sample belong to one of three classes: Iris Setosa, Iris Versicolor, Iris Virginica.

Train-test split → To evaluate the model, the dataset is typically divided into training and testing sets. The train-test split ratio is 80% (Training) and 20% (Testing)

FAQs →

Q-1) state the objectives of the lab experiment

→ i) Implement SVM and Decision Tree classifiers.

ii) Compare their performance using metrics like accuracy, precision, recall, and F1-score.

iii) Evaluate which algorithm works better for the given dataset.

Q2) Specify the parameters and hyperparameters used for both the SVM and decision Tree classifier.

Ans-2) SVM classifier

- Kernel: Defines the type of hyperplane for classification. 'linear'.
- Random state: 42 (Ensures Reproducibility)
- Other option: C (regularization), gamma (RBF Kernel coefficient)

Decision Tree classifier.

- Max Depth: None (Allows the tree to grow fully)
- Criteria: 'gini' (Used to measure the quality of a path)
- Random State: 42 (for Reproducibility).

(Q3) Explain the importance of comparing these two algorithms

Ans i) Performance Assessment - Identify which algorithm yields better accuracy precision recall and F1-score for the dataset

ii) Model selection - Help choose the right algo based on interpretability, training time and scalability

svm.py

```python
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

# Load a sample dataset (Iris dataset in this case)
iris = datasets.load_iris()
X = iris.data  # Features
y = iris.target  # Labels

# Split the dataset into training and testing sets (70% train, 30% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train an SVM classifier
svm_classifier = SVC(kernel='linear')  # Linear kernel SVM
svm_classifier.fit(X_train, y_train)

# Create and train a Decision Tree classifier
dt_classifier = DecisionTreeClassifier()  # Default parameters for Decision Tree
dt_classifier.fit(X_train, y_train)

# Make predictions with both classifiers
svm_predictions = svm_classifier.predict(X_test)
dt_predictions = dt_classifier.predict(X_test)

# Calculate accuracy for both classifiers
svm_accuracy = accuracy_score(y_test, svm_predictions)
dt_accuracy = accuracy_score(y_test, dt_predictions)

# Print out the accuracies
print("SVM Classifier Accuracy:", svm_accuracy)
print("Decision Tree Classifier Accuracy:", dt_accuracy)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
/Users/rushilpandhare/opt/anaconda3/bin/python "/Users/rushilpandhare/Desktop/AIMLT LABS/AIMLT_LAB5/svm.py"
(base) rushilpandhare@Rushils-Air AIMLT_LAB5 % /Users/rushilpandhare/opt/anaconda3/bin/python "/Users/rushilpandhare/Desktop/AIMLT LABS/AIMLT_LAB5/svm.py"
SVM Classifier Accuracy: 1.0
Decision Tree Classifier Accuracy: 1.0
(base) rushilpandhare@Rushils-Air AIMLT_LAB5 %
```