LAB No. 4

Random Forest and Support Vector Machine Classifier

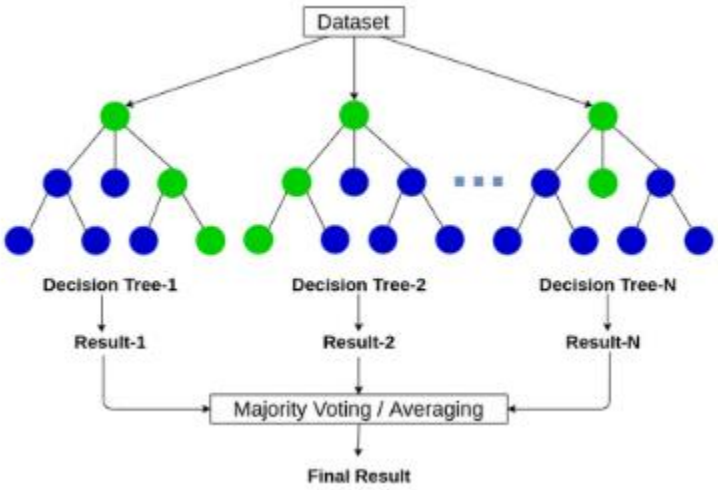
In this lab, students will learn and implement two powerful supervised machine learning algorithms: Random Forest Classifier and Support Vector Machine (SVM) Classifier. The lab focuses on understanding how ensemble learning improves classification accuracy using Random Forests and how SVM constructs optimal decision boundaries for classification problems.

Students will begin with a small dataset to understand model behavior and then apply both classifiers to real-world datasets using Python and Scikit-learn. Model performance will be evaluated using accuracy metrics, and comparisons will be made between Random Forest and SVM classifiers.

Introduction & Theory

Random Forest Classifier

Random Forest is an **ensemble learning method** that builds multiple decision trees and combines their outputs to make a final prediction. Each tree is trained on a random subset of data and features, which improves accuracy and reduces overfitting.



Advantages:

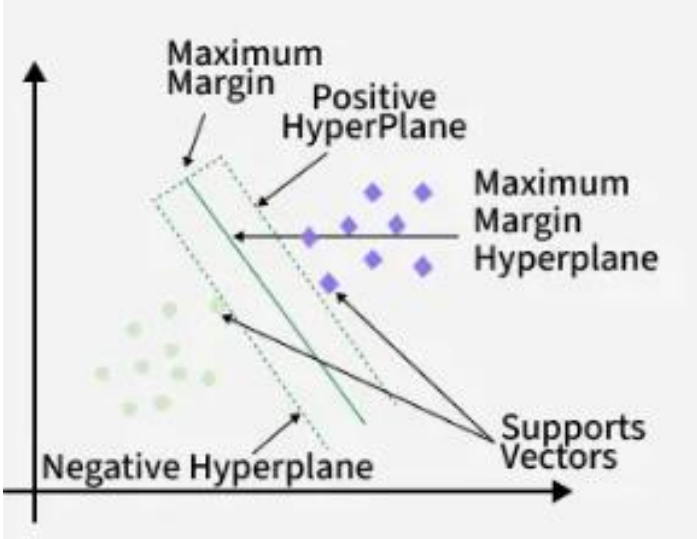
- High accuracy
- Reduces overfitting
- Works well with large datasets

Support Vector Machine (SVM) Classifier

Support Vector Machine is a supervised learning algorithm that finds the **optimal hyperplane** that best separates data points of different classes. SVM can perform both linear and non-linear classification using **kernel functions**.

Advantages:

- Effective in high-dimensional spaces
- Works well with small datasets
- Strong theoretical foundation



Solved Examples

Example 1

Build a Random Forest classifier to predict whether a student passes or fails based on study hours and attendance.

Solution:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Prediction
prediction = model.predict([[1, 1]])
print("Predicted Result (1=Pass, 0=Fail):", prediction[0])
```

Example 2:

Use an SVM classifier to predict whether a student passes or fails using the same dataset.

Solution:

```
from sklearn.svm import SVC

# Create dataset
data = {
    'StudyHours': ['Low', 'High', 'High', 'Low', 'High'],
    'Attendance': ['Poor', 'Good', 'Poor', 'Good', 'Good'],
    'Result': ['Fail', 'Pass', 'Pass', 'Fail', 'Pass']
}

df = pd.DataFrame(data)

# Encode categorical variables
encoder = LabelEncoder()
for col in df.columns:
    df[col] = encoder.fit_transform(df[col])

# Features and target
X = df[['StudyHours', 'Attendance']]
y = df['Result']

# Train SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X, y)

# Prediction
svm_prediction = svm_model.predict([[1, 1]])
print("Predicted Result (1=Pass, 0=Fail):", svm_prediction[0])
```

Example 3

Compare Random Forest and SVM classifiers on IRIS dataset.

Solution

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.svm import SVC

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# SVM model
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test)

# Accuracy
print("Random Forest Accuracy:", accuracy_score(y_test, rf_pred))
print("SVM Accuracy:", accuracy_score(y_test, svm_pred))
```

Comparison Summary

Algorithm	Strengths	Limitations
Random Forest	High accuracy, less overfitting	Slower, less interpretable
SVM	Effective in high dimensions	Sensitive to kernel choice

LAB Assignment No 4

Topic: Random Forest and Support Vector Machine Classifier

Question 1

Classify flower species using Random Forest.

Task:

1. Load the *Iris dataset* from sklearn.datasets.
2. Split into training (70%) and testing (30%) sets.

3. Train a **Random Forest Classifier**.
4. Predict flower species on the test set.
5. Calculate and print **model accuracy**.

Question 2

Use SVM on Breast Cancer Dataset and Classify tumors as malignant or benign.

Task:

1. Load the *Breast Cancer* dataset using sklearn.datasets.load_breast_cancer.
2. Train an **SVM classifier** (use SVC(kernel='linear')).
3. Evaluate the model using **accuracy** and **confusion matrix**.

Question 3

Use Random Forest on CSV Dataset (Custom) : Predict student pass/fail based on study hours and scores.

Task:

1. Load a CSV file (e.g., students.csv) with columns: study_hours, attendance, marks, result.
2. Train a **Random Forest Classifier** to predict result (Pass/Fail).
3. Display **accuracy score** and **feature importance**.

Question 4

Use SVM on Digits Dataset and to identify the: Handwritten digit recognition.

Task:

1. Load the *Digits dataset* from sklearn.datasets.load_digits.
2. Train an **SVM classifier** with an RBF kernel.
3. Test on unseen data.
4. Print **accuracy** and visualize some **misclassified samples**.

Question 5:

Compare Random Forest vs SVM on Same Dataset (you can choose any dataset):
Compare two models on the same data.

Task:

- Use the *Wine dataset* from sklearn.datasets.load_wine.
- Train both:

  RandomForestClassifier(n_estimators=100)

- SVC(kernel='rbf')
- Print accuracy of both models.
- Conclude which performs better.

LAB Assessment

Student Name		**LAB Rubrics**	CLO3 , P5, PLO5
		Total Marks	10
Registration No		**Obtained Marks**	
		Teacher Name	Dr. Syed M Hamedoon
Date		**Signature**	