LAB No. 4

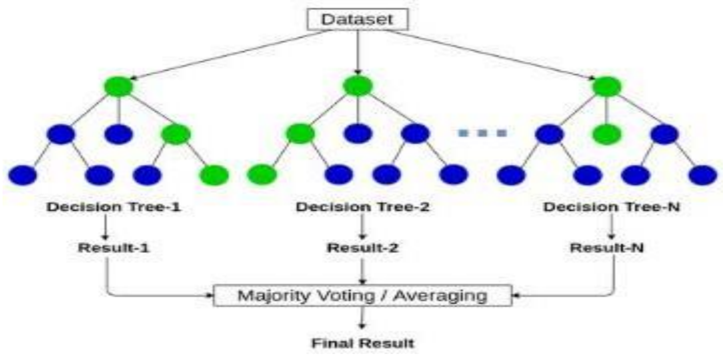
Random Forest and Support Vector Machine Classifier

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 s 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:

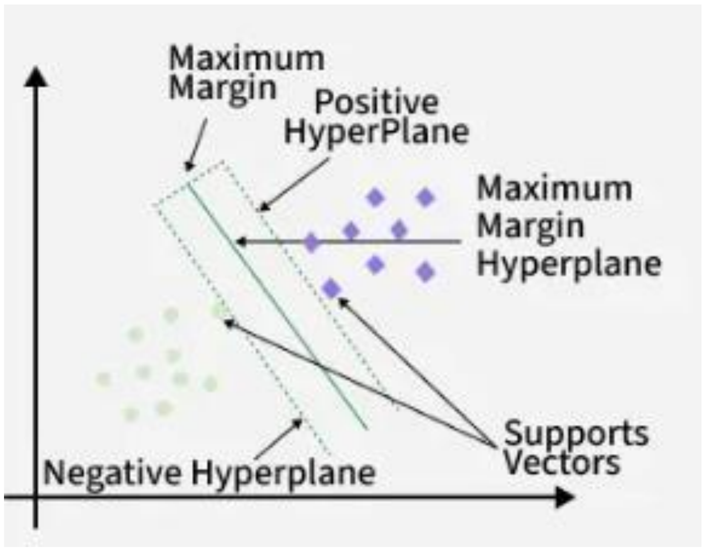
- High accuracy
- Reduces overfitting
- Works well with large dataset

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']
}
```

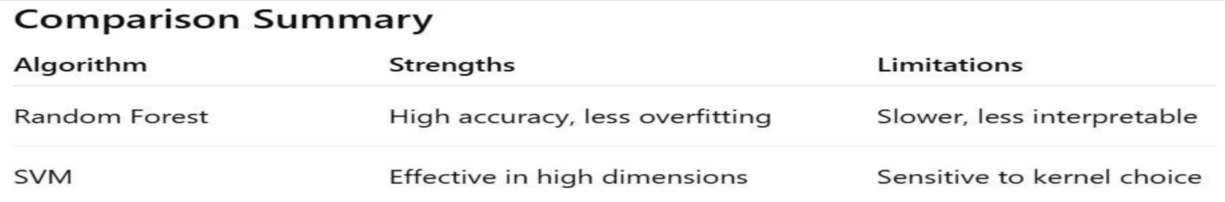
```
df = pd.DataFrame(data)

# 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.

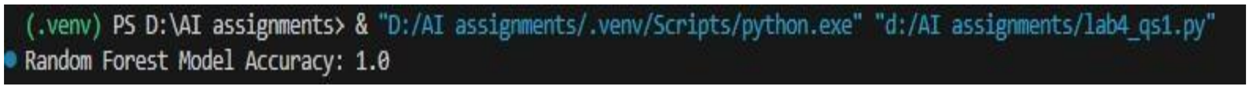
Task:

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**.

Code:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split from
sklearn.ensemble import RandomForestClassifier from
sklearn.metrics import accuracy_score
iris = load_iris()
X = iris.data y
= iris.target
# 3. Split into training (70%) and testing (30%) sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) #
4. Train a Random Forest Classifier
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)

# 5. Predict flower species on the test set y_pred
= rf.predict(X_test)

# 6. Calculate and print model accuracy
accuracy  =  accuracy_score(y_test,  y_pred)
print("Random Forest Model Accuracy:", accuracy)
```

Output:



```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs1.py"
Random Forest Model Accuracy: 1.0
```

Question 2

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

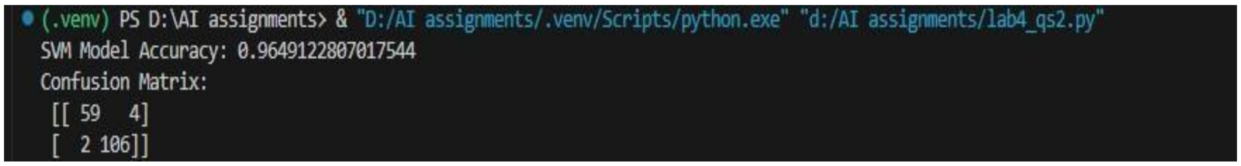
Task:

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**.

Code:

```
from sklearn.datasets import load_breast_cancer from
sklearn.model_selection import train_test_split from
sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix cancer
= load_breast_cancer()
X = cancer.data
y = cancer.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 4. Train SVM classifier with linear kernel
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)

# 5. Make predictions on the test set
y_pred = svm_model.predict(X_test)

# 6. Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# 7. Print results
print("SVM Model Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
```

output:



```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs2.py"
SVM Model Accuracy: 0.9649122807017544
Confusion Matrix:
 [[ 59   4]
 [  2 106]]
```

Question 3

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

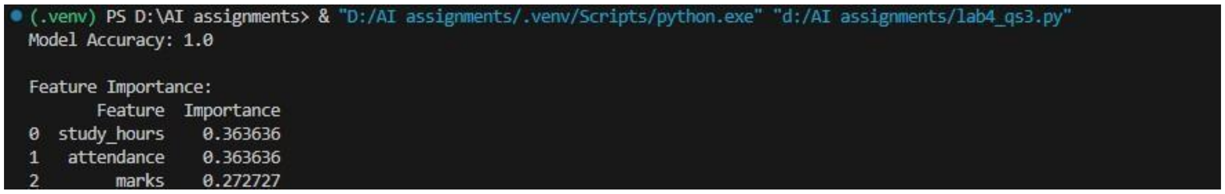
Task:

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**.

Code:

```
import pandas as pd
from sklearn.model_selection import train_test_split from
sklearn.ensemble import RandomForestClassifier from
sklearn.metrics import accuracy_score
data = pd.read_csv("students.csv")
X = data[['study_hours', 'attendance', 'marks']] y
= data['result']
y = y.map({'Fail': 0, 'Pass': 1})
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
y_pred = rf_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
feature_importance = pd.DataFrame({
   'Feature': X.columns,
   'Importance': rf_model.feature_importances_
}).sort_values(by='Importance', ascending=False)
print("\nFeature Importance:")
print(feature_importance)
```

output:



```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs3.py"
Model Accuracy: 1.0

Feature Importance:
       Feature  Importance
0  study_hours    0.363636
1   attendance    0.363636
2        marks    0.272727
```

Question 4

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

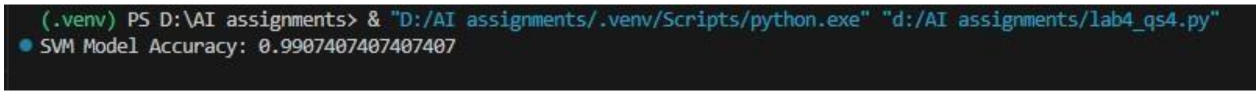
Task:

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**.

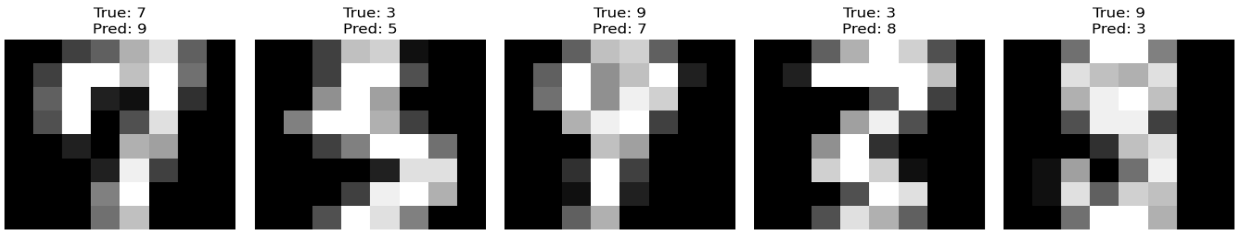
Code:

```
# Import required libraries
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score import
matplotlib.pyplot as plt
import numpy as np digits
= load_digits()
X = digits.data y
= digits.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42) svm_model
= SVC(kernel='rbf', gamma=0.001, C=10)  # RBF kernel
svm_model.fit(X_train, y_train) y_pred
= svm_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred) print("SVM
Model Accuracy:", accuracy)
misclassified_indices = np.where(y_test != y_pred)[0]
print(f"\nNumber of misclassified samples: {len(misclassified_indices)}")

plt.figure(figsize=(10,  4))
  for i, index in enumerate(misclassified_indices[:5]):
     plt.subplot(1, 5, i + 1)
     plt.imshow(X_test[index].reshape(8, 8), cmap='gray')
     plt.title(f"True:                    {y_test[index]}\nPred:
     {y_pred[index]}") plt.axis('off')

plt.tight_layout() plt.show()
```

output:



```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs4.py"
● SVM Model Accuracy: 0.9907407407407407
```



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)

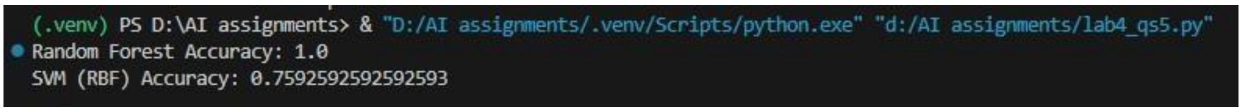
- SVC(kernel='rbf')
- Print accuracy of both models.
- Conclude which performs better.

Code:

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split from
sklearn.ensemble import RandomForestClassifier from
sklearn.svm import SVC
from sklearn.metrics import accuracy_score wine
= load_wine()
X = wine.data y
= wine.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 3. Train Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_pred)
```

```
# 4. Train SVM with RBF kernel
svm_model = SVC(kernel='rbf', gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
svm_pred = svm_model.predict(X_test) svm_accuracy
= accuracy_score(y_test, svm_pred)

# 5. Print accuracy of both models
print("Random Forest Accuracy:", rf_accuracy)
print("SVM (RBF) Accuracy:", svm_accuracy)

# 6. Conclude which model performs better if
rf_accuracy > svm_accuracy:
  print("\n✅ Random Forest performs better on the Wine dataset.") elif
svm_accuracy > rf_accuracy:
  print("\n✅ SVM (RBF) performs better on the Wine dataset.") else:
    print("\n     Both models perform equally well on the Wine dataset.")
```

output:



```
(.venv) PS D:\AI assignments> & "D:/AI assignments/.venv/Scripts/python.exe" "d:/AI assignments/lab4_qs5.py"
● Random Forest Accuracy: 1.0
SVM (RBF) Accuracy: 0.7592592592592593
```

LAB Assessment

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

Laboratory Work Assessment Rubrics

Sr. No.	Performance Indicator	Excellent (5)	Good (4)	Average (3)	Fair (2)	Poor (1)
1	**Theoretical knowledge 10%**	Student knows all the related concepts about the theoretical background of the experiment and rephrase those concepts in written and oral assessments	Student knows most of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows few of the related concepts about the theoretical background of the experiment and partially rephrase those concepts in written and oral assessments	Student knows very little about the related concepts about the theoretical background of the experiment and poorly rephrase those concepts in written and oral assessments	Student has poor understanding of the related concepts about the theoretical background of the experiment and unable to rephrase those concepts in written and oral assessments
2	**Application Functionality 10%**	Application runs smoothly and operation of the application runs efficiently	Application compiles with no warnings. Robust operation of the application, with good recovery.	Application compiles with few or no warnings. Consideration given to unusual conditions with reasonable	Application compiles and runs without crashing. Some attempt at detecting and correcting errors.	Application does not compile or compiles but crashes. Confusing. Little or no error detection or correction.
3	**Specifications 10%**	The program works very efficiently and meets all of the required specifications.	The program works and meets some of the specifications.	The program works and produces the correct results and displays them correctly. It also meets most of the other specifications.	The program produces correct results but does not display them correctly.	The program is producing incorrect results.
4	**Level of understanding of the learned skill 10%**	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner	Provide complete and logical answers based upon accurate technical content to the questions asked by examiner with few errors	Provide partially correct and logical answers based upon minimum technical content to the questions asked by examiner	Provide very few and illogical answers to the questions asked by examiner.	Provide no answer to the questions asked by examiner.
5	**Readability and Reusability 10%**	The code is exceptionally well organized and very easy to follow and reused	The code is fairly easy to read. The code could be reused as a whole or each class could be reused.	Most of the code could be reused in other programs.	Some parts of the code require change before they could be reused in other programs.	The code is poorly organized and very difficult to read and not organized for reusability.

6	**AI System Design 10%**	Well-designed AI models. Code is highly maintainable	Good designed AI models and Little code duplications	Some attempt to make AI models. Code can be maintained with significant effort	Little attempt to design AI models and less understanding of code	Very poor attempt to design AI models and its code
7	**Responsiveness to Questions/ Accuracy 10%**	1. Responds well, quick and very accurate all the time. 2. Effectively uses eye contact, speaks clearly, effectively and confidently using suitable volume	1. Generally Responsive and accurate most of the times. 2. Maintains eye contact, speaks clearly with suitable volume and pace.	1. Generally Responsive and accurate few times. 2. Some eye contact, speaks clearly and unclearly in different portions.	1. Not much Responsive and accurate most of the times. 2. Uses eye contact ineffectively and fails to speak clearly and audibly	. 1. Non Responsive and inaccurate all the times. 2. No eye contact and unable to speak 3. Dresses inappropriately
8	**Efficiency** **10%**	The code is extremely efficient without sacrificing readability and understanding	The code is fairly efficient without sacrificing readability and understanding	Some part of the code is efficient and other part of the code is not understandable and work properly	The code is brute force and unnecessarily long	The code is huge and appears to be patched together
9	**Delivery** **10%**	The program was delivered in time during lab.	The program was delivered in Lab before the end time.	The program was delivered within the due date.	The code was delivered within a day after the due date.	The code was delivered more than 2 days overdue.
10	**Awareness of Safety Guidelines 10%**	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is fully compliant to the guidelines	Student has sufficient knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is Partially compliant to the guidelines	Student has little knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines	Student has no knowledge of the laboratory safety SOPs and protocol and is non-compliant to the guidelines