

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**.

The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, there are tabs for 'Code', 'Markdown', 'Run All', 'Clear All Outputs', 'Outline', and a Python version indicator (Python 3.13.7). Below the tabs, the code cell contains the following Python code:

```
from sklearn.datasets import load_iris
iris = load_iris()

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Random Forest Model Accuracy:", accuracy)
```

The code cell is followed by the output cell, which displays the result of the last print statement: "Random Forest Model Accuracy: 1.0". The output cell has a 'Python' label at the bottom right.

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 + Markdown | ▶ Run All ⌂ Clear All Outputs | ⌂ Outline ... Python 3.13.7
▶ ...
print(data.feature_names[:5])
print(data.data.shape)
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

# Load the dataset
data = load_breast_cancer()
X = data.data
y = data.target
[1]
...
['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness']
(569, 30)

[2]
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
...
SVC
Parameters

[3]
from sklearn.metrics import accuracy_score, confusion_matrix
y_pred = svm_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", accuracy)
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
...
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 + Markdown | ▶ Run All ⌂ Clear All Outputs | ⌂ Outline ...
Python 3.13.7
... ⌂

[1] import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

[2] df = pd.read_csv("student_lab4_q3.csv")
X = df[['study_hours', 'attendance', 'marks']]
y = df['result']
y = y.map({'Pass': 1, 'Fail': 0})

[3] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

[4] rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

... - RandomForestClassifier ⓘ ⓘ
  ▶ Parameters

[5] y_pred = rf_model.predict(X_test)
print("Predicted Results (first 10):", y_pred[:10])
print("Actual Results (first 10):", y_test[:10].values)

... Predicted Results (first 10): [0 1 1 1 0 1 1 1]
  Actual Results (first 10): [0 1 1 1 0 1 1 1]

[6] y_pred = rf_model.predict(X_test)

[7] accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

... Accuracy: 1.0

[8] importances = rf_model.feature_importances_
feature_names = X.columns
for name, importance in zip(feature_names, importances):
    print(f'{name}: {importance:.4f}')

... study_hours: 0.3571
  attendance: 0.3564
  marks: 0.2865

```

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 + Markdown | ▶ Run All ⌂ Clear All Outputs | ⌂ Outline ...
```

Python 3.13.7

```
[1] from sklearn.datasets import load_digits
    digits = load_digits()
    X = digits.data      # features (image pixels)
    y = digits.target    # labels (0-9 digits)
```

[2]

```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

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)  # RBF kernel use kiya
svm_model.fit(X_train, y_train)               # Model train hua
```

[2] SVC

Parameters

```
[3] y_pred = svm_model.predict(X_test)
```

[3]

```
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
print("Model Accuracy:", accuracy_score(y_test, y_pred))
misclassified = np.where(y_test != y_pred)[0]
plt.figure(figsize=(10, 4))
for i, index in enumerate(misclassified[: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.show()
```

[4]

```
Model Accuracy: 0.9987407407407407
```

True: 7
Pred: 9

True: 3
Pred: 5

True: 9
Pred: 7

True: 3
Pred: 8

True: 9
Pred: 3

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 + Markdown | ▶ Run All ⌂ Clear All Outputs | ⌂ Outline ...
[1] import pandas as pd
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.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
[1] Python

[2] wine = load_wine()
X = wine.data
y = wine.target
target_names = wine.target_names
[2] Python

[3] X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
[3] Python

[4] rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
acc_rf = accuracy_score(y_test, y_pred_rf)
[4] Python

[5] scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train_s, y_train)
y_pred_svm = svm_model.predict(X_test_s)
acc_svm = accuracy_score(y_test, y_pred_svm)
[5] Python

+ Code + Markdown | ▶ Run All ⌂ Clear All Outputs | ⌂ Outline ...
[4] acc_rf = accuracy_score(y_test, y_pred_rf)
[4] Python

[5] scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

svm_model = SVC(kernel='rbf', random_state=42)
svm_model.fit(X_train_s, y_train)
y_pred_svm = svm_model.predict(X_test_s)
acc_svm = accuracy_score(y_test, y_pred_svm)
[5] Python

[6] print("Random Forest Accuracy: {:.4f} ({:.2f}%)".format(acc_rf, acc_rf*100))
print("SVM (RBF) Accuracy : {:.4f} ({:.2f}%)".format(acc_svm, acc_svm*100))
[6] Python

... Random Forest Accuracy: 1.0000 (100.00%)
SVM (RBF) Accuracy : 0.9815 (98.15%)

[7] if acc_rf > acc_svm:
    conclusion = "Random Forest performs better on this split."
elif acc_svm > acc_rf:
    conclusion = "SVM performs better on this split."
else:
    conclusion = "Both models perform equally well."
print("Conclusion:", conclusion)
[7] Conclusion: Random Forest performs better on this split.
[7] Python
```