LAB Assignment No 4

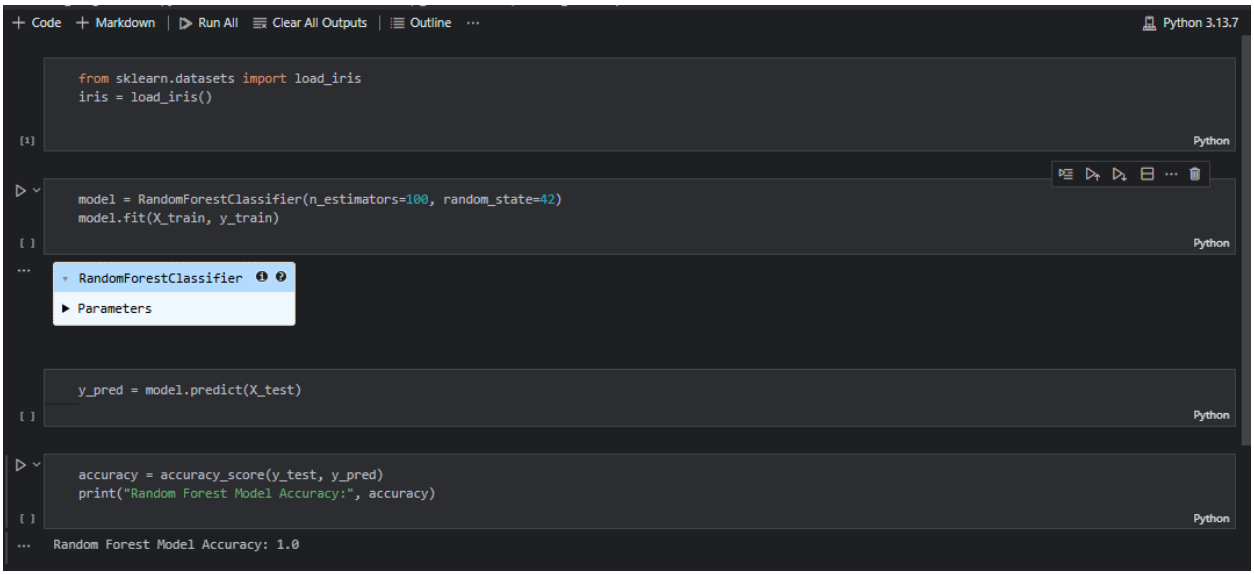
Topic: Random Forest and Support Vector Machine Classifier

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**.

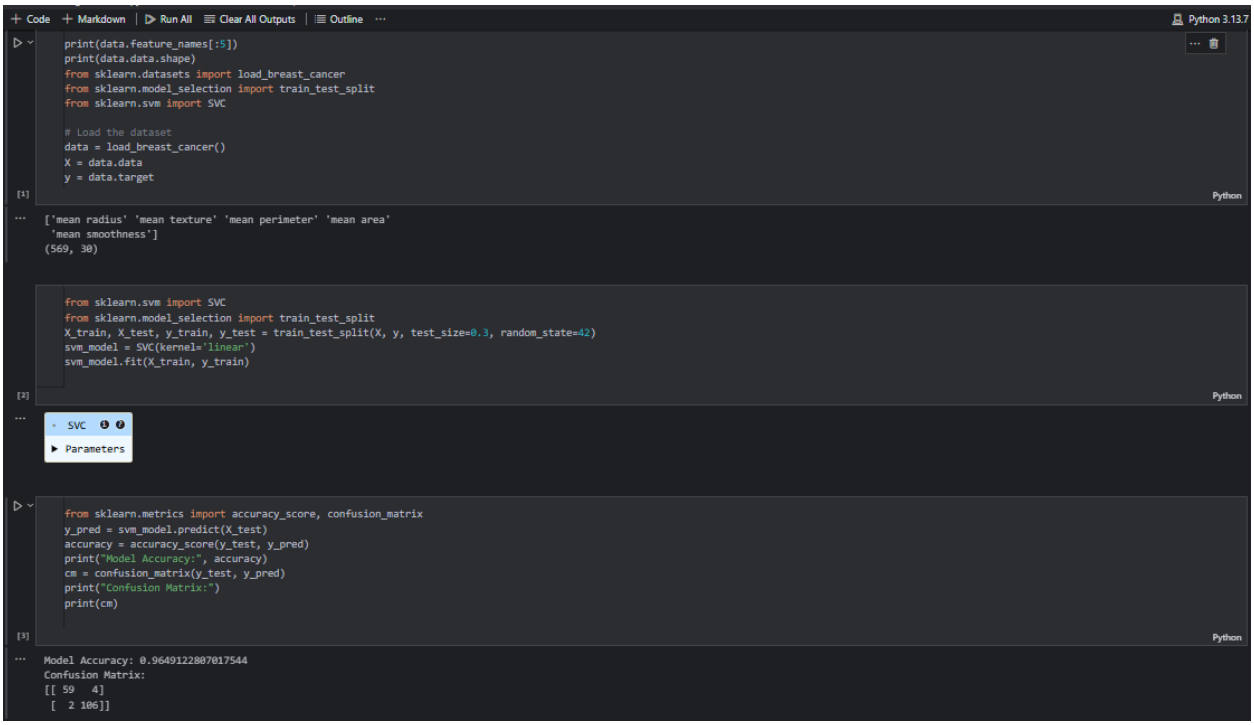


Question 2

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**.

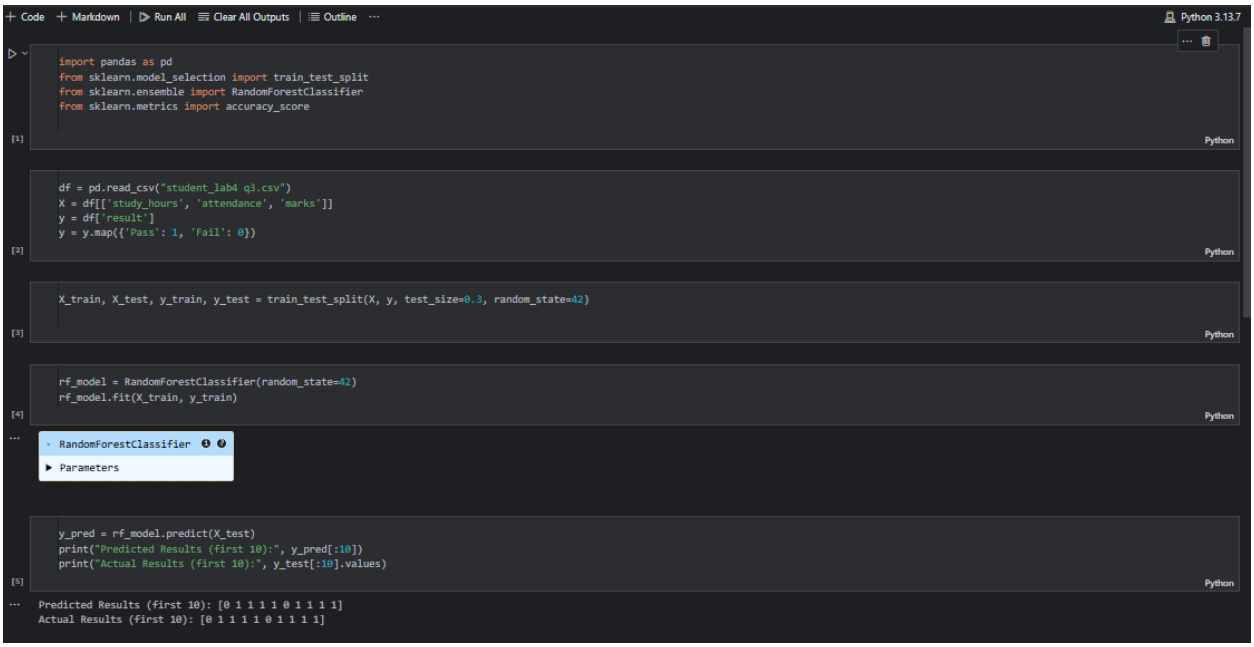


Question 3

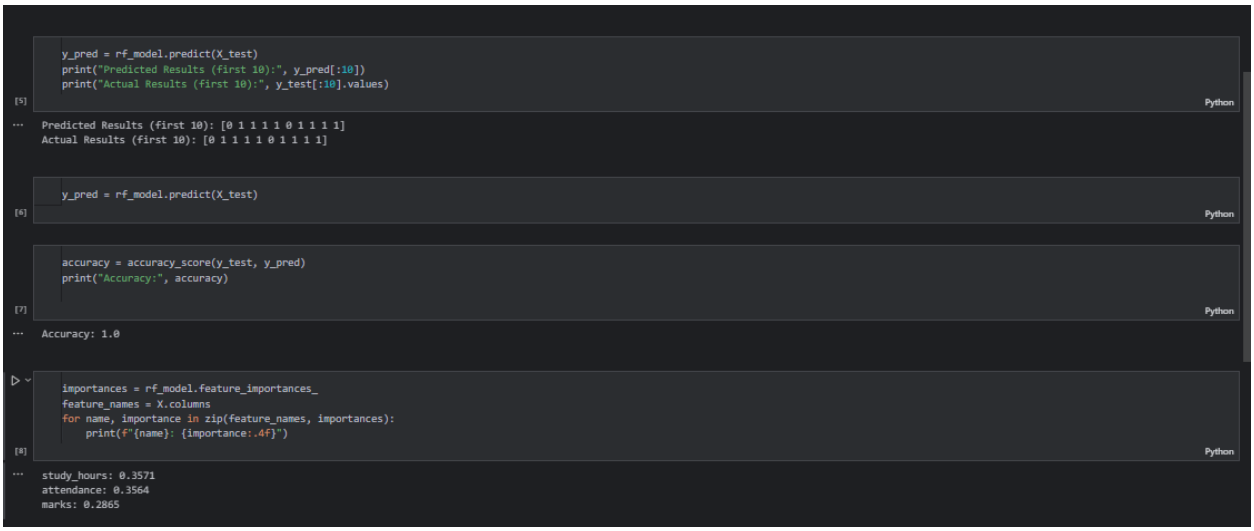
Use Random Forest on CSV Dataset (Custom) : Predict student pass/fail based on study hours and scores.

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**.



```
+ Code + Markdown | ▷ Run All | Clear All Outputs | Outline ···                                Python 3.13.7

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

df = pd.read_csv("student_lab4 q3.csv")
X = df[['study_hours', 'attendance', 'marks']]
y = df['result']
y = y.map({'Pass': 1, 'Fail': 0})
[2]                                                                                          Python

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

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

··· RandomForestClassifier
    ▶ Parameters

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

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



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

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

y_pred = rf_model.predict(X_test)
[6]                                                                                          Python

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

··· Accuracy: 1.0

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

··· 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

from sklearn.datasets import load_digits

digits = load_digits()
X = digits.data      # features (image pixels)
y = digits.target    # labels (0-9 digits)

[1]                                                                              Python
```

```
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]                                                                              Python
```

SVC
▶ Parameters

```
y_pred = svm_model.predict(X_test)

[3]                                                                              Python
```

```
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()
```

Question 5:

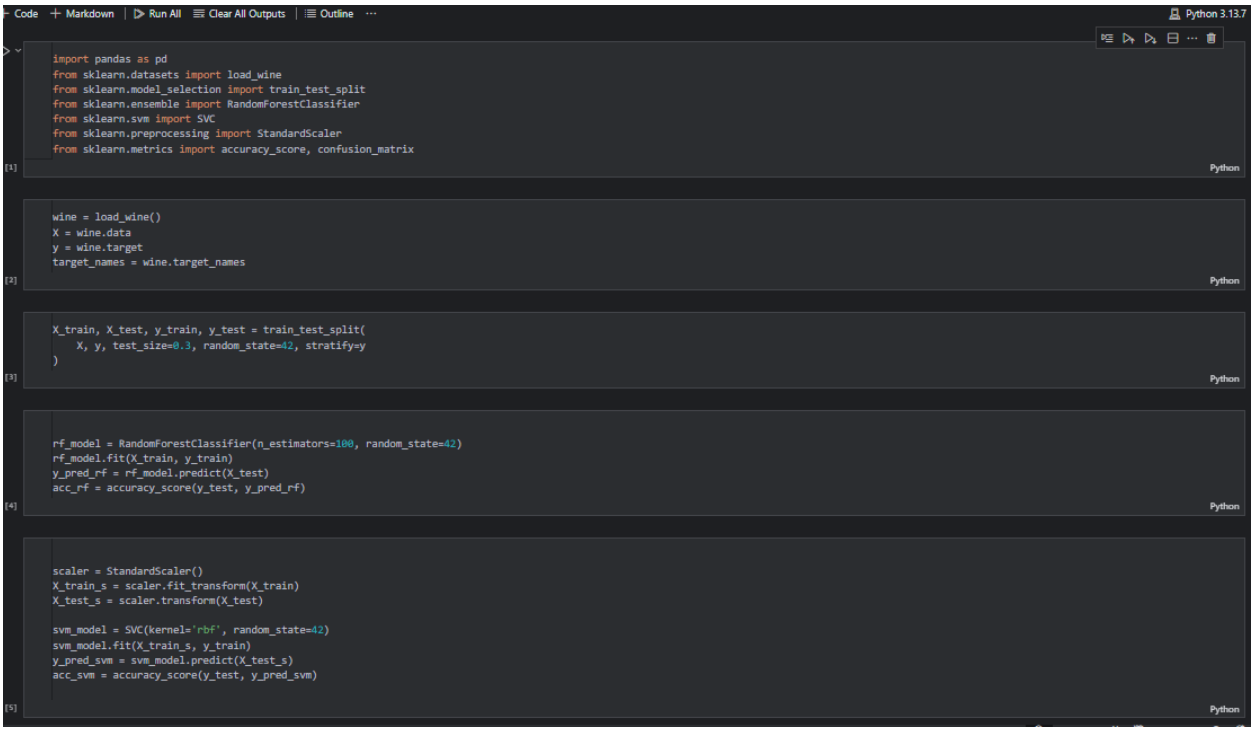
Compare Random Forest vs SVM on Same Dataset (you can choose any dataset): Compare two models on the same data.

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.



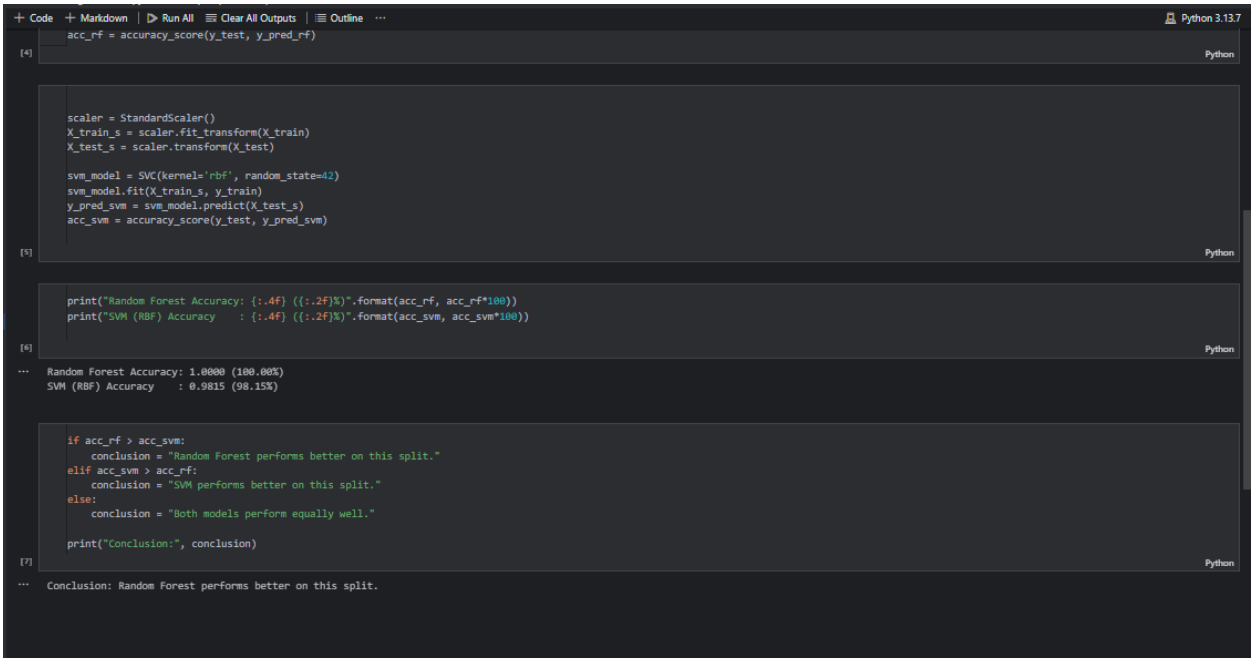
```
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
wine = load_wine()
X = wine.data
y = wine.target
target_names = wine.target_names
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

```
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)
```

```
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)
```



```
acc_rf = accuracy_score(y_test, y_pred_rf)
```

```
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)
```

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

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

```
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)
```

```
Conclusion: Random Forest performs better on this split.
```