Assignment 1: Classifying Iris Flowers with a Support Vector Machine

Name: Pratik Amrit
Roll No.: 2201AI29

1. Objective

For this assignment, the objective was to apply a machine learning algorithm to a dataset of my choice. I selected the well-known Iris flower dataset and implemented a **Support Vector Machine (SVM)** classifier to predict the species of a flower based on its physical measurements. The entire project was developed in Python using the scikit-learn library.

2. The Iris Dataset

The dataset contains 150 samples from three different species of Iris flowers: *Iris Setosa*, *Iris Versicolor*, and *Iris Virginica*.

For each sample, four features are provided:

1. Sepal Length (cm)
2. Sepal Width (cm)
3. Petal Length (cm)
4. Petal Width (cm)

The task is to build a model that can take these four features as input and predict the correct species as the output.

3. Algorithm: Support Vector Machine (SVM)

I chose the Support Vector Machine algorithm because of its effectiveness in classification tasks. The fundamental goal of an SVM is to find the optimal **hyperplane**—a boundary that best separates the data points into their respective classes.

The "optimal" hyperplane is the one that has the largest possible **margin**, which is the distance between the hyperplane and the nearest data points from each class. These closest points are called "support vectors" because they are critical in defining the position of the hyperplane. For this project, I used a linear kernel, which means the model looks for a straight-line boundary to separate the classes.

4. Python Implementation

Here is the complete Python script used to load the data, train the SVM model, and evaluate its results.

```
# 1. Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC # Support Vector Classifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# 2. Load the dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target # Target labels (0, 1, 2)

# Get feature and target names for context
feature_names = iris.feature_names
target_names = iris.target_names
print(f"Feature Names: {feature_names}")
print(f"Target Names: {target_names}\n")

# 3. Split the data into training and testing sets
# We use 80% for training and 20% for testing.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}\n")

# 4. Initialize and train the Support Vector Machine (SVM) model
# We use a linear kernel, a common choice for SVM.
svm_model = SVC(kernel='linear', random_state=42)
svm_model.fit(X_train, y_train)
print("SVM Model trained successfully.\n")

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

# 6. Evaluate the model's performance
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}\n")

# Display the classification report
print("Classification Report:")
```

```
print(classification_report(y_test, y_pred, target_names=target_names))

# Generate and display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target_names,
yticklabels=target_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix for Iris Classification (SVM)')

# Save the plot as an image file
plt.savefig('confusion_matrix_svm.png')
print("Confusion matrix plot saved as 'confusion_matrix_svm.png'")
plt.show()
```

5. Results and Output

The model was trained on 80% of the dataset and then evaluated on the remaining 20% (30 samples).

A. Performance Metrics

The script produced the following output, showing the model's performance:

```
Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target Names: ['setosa' 'versicolor' 'virginica']

Training data shape: (120, 4)
Testing data shape: (30, 4)

SVM Model trained successfully.

Model Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
```

macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

The model achieved a **perfect accuracy of 1.00**, meaning it correctly classified all 30 flowers in the test set. The precision, recall, and f1-score were all 1.00 for each class, which confirms that there were no false positives or false negatives.

B. Confusion Matrix

The confusion matrix provides a visual breakdown of the model's predictions versus the actual labels.

(confusion_matrix_svm.png, is generated when you run the Python script)

As shown, the diagonal of the matrix contains all 30 test samples, and the off-diagonal cells are all zero. This visually confirms that every single prediction was correct.

6. Conclusion

This project successfully demonstrated the use of a Support Vector Machine for a multi-class classification problem. The SVM model, even with a simple linear kernel, was able to perfectly classify the Iris species in the test set, achieving 100% accuracy. This indicates that the classes in the Iris dataset are linearly separable and that SVM is a highly effective algorithm for this task.

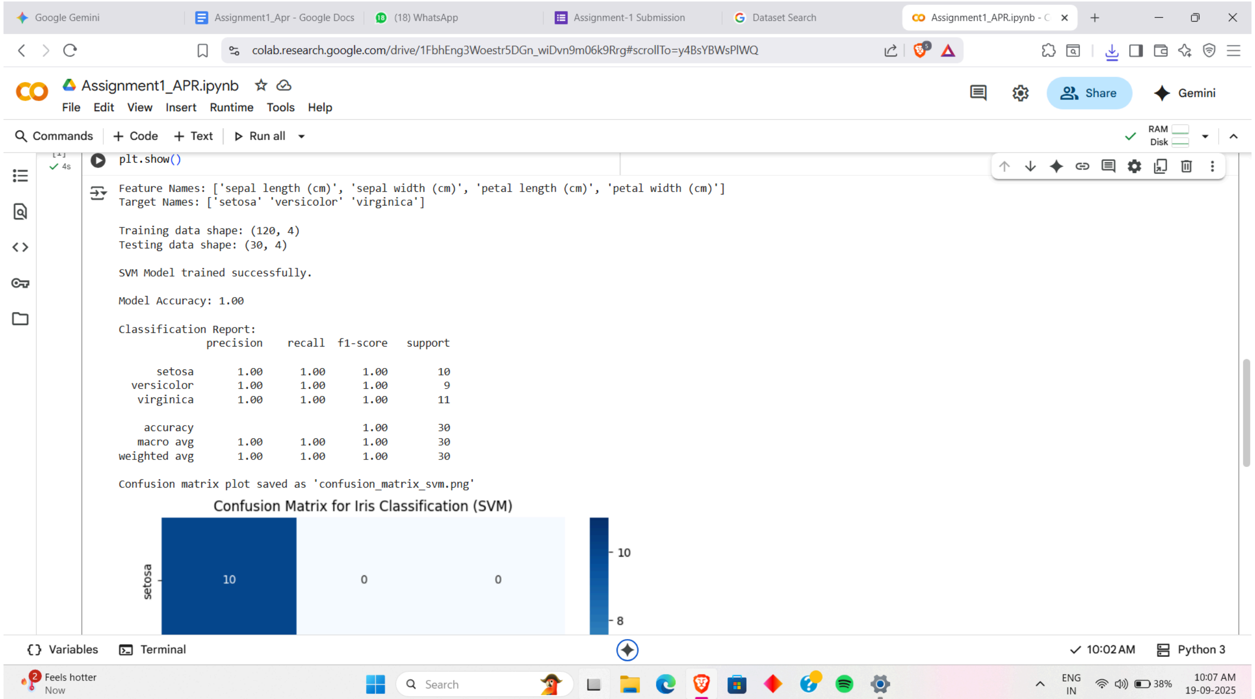
7. ScreenShot

Assignment1_APR.ipynb

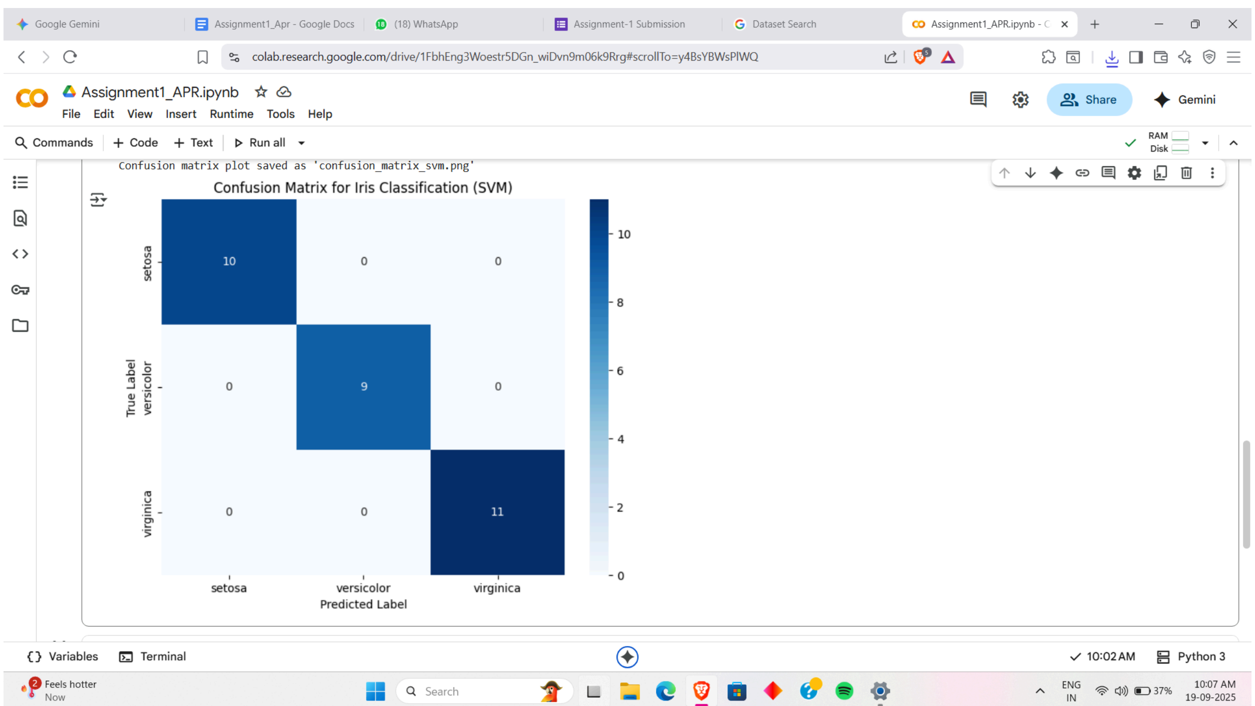
```python
# main.py

# 1. Import necessary libraries
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC # Support Vector Classifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# 2. Load the dataset
iris = load_iris()
X = iris.data  # Features
y = iris.target # Target labels (0, 1, 2)

# Get feature and target names for context
feature_names = iris.feature_names
target_names = iris.target_names
print(f"Feature Names: {feature_names}")
print(f"Target Names: {target_names}\n")

# 3. Split the data into training and testing sets
# We use 80% for training and 20% for testing.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}\n")

# 4. Initialize and train the Support Vector M
# We use a linear kernel, a common choice for S...
```

```python
svm_model.fit(X_train, y_train)
print("SVM Model trained successfully.\n")

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

# 6. Evaluate the model's performance
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}\n")

# Display the classification report
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=target_names))

# Generate and display the confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=target_names, yticklabels=target_names)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix for Iris Classification (SVM)')

# Save the plot as an image file
plt.savefig('confusion_matrix_svm.png')
print("Confusion matrix plot saved as 'confusion_matrix_svm.png'")
plt.show()
```

```
Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target Names: ['setosa' 'versicolor' 'virginica']
```



```
plt.show()

Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target Names: ['setosa' 'versicolor' 'virginica']

Training data shape: (120, 4)
Testing data shape: (30, 4)

SVM Model trained successfully.

Model Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion matrix plot saved as 'confusion_matrix_svm.png'
```

```
Confusion matrix plot saved as 'confusion_matrix_svm.png'
```