Applying Support Vector Machines (SVM) to the Iris Dataset

Name: Faisal Khalid
Student ID: 23100733
Github Link: https://github.com/Faisalch80/SVM-23100733-

Introduction

One of the best supervised learning methods for classification problems is Support Vector Machines (SVM). They function by determining which hyperplane in a datasets best divides several classes. SVM will be used in this tutorial on the Iris dataset, which is a popular dataset in the machine learning community.

After completing this lesson, you will comprehend:

- What SVM is and why classification works well with it.
- How to get data ready for SVM.
- How to train and assess the performance of an SVM model.
- How to use a confusion matrix to visualise the model's predictions.

1. An Interpretation of the iris dataset

There are 150 iris flower samples in the Iris dataset (UCI Machine Learning Repository - Iris Dataset – https://archive.ics.uci.edu/dataset/53/iris), which has four characteristics.

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

There are three classes in the dataset.
1. Iris-setosa
2. Iris-versicolor
3. Iris-virginica

This is a multiclass classification problem since each sample is a member of one of these three species.

2. Importing libraries

Importing libraries will be our first step.

[1]:
```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.datasets import load_iris
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

3. Exploring and loading the dataset

Second, we will load the dataset and examine its structure first.

```
[4]: from sklearn.datasets import load_iris
     iris = load_iris()

     import pandas as pd
     df = pd.DataFrame(iris.data, columns=iris.feature_names)
     df['species'] = iris.target_names[iris.target]
     print(df.head())
```

This will show the first few rows of the dataset, providing information about its structure.

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
```

4. Data Preprocessing and Data Splitting

We must divide the data into a training set (80%) and a testing set (20%) before we can begin training the SVM model.

```
[6]: from sklearn.model_selection import train_test_split

     X = df.drop('species', axis=1)
     y = df['species']

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
     print(f"Training set size: {X_train.shape[0]}")
     print(f"Testing set size: {X_test.shape[0]}")
```

The outcome is shown by this code.

```
Training set size: 120
Testing set size: 30
```

5. SVM Model Training

A linear kernel, which performs well for a variety of classification tasks, will now be used to train an SVM model. Using the training dataset, an SVM classifier is initialised and trained. We employ a linear kernel, which determines the most effective hyperplane for class separation. Because of its efficiency in processing linearly separable data, the linear kernel was selected.

```
[8]: svm_model = SVC(kernel='linear', random_state=42)
     svm_model.fit(X_train, y_train)
```

The outcome:

```
[8]: SVC
     SVC(kernel='linear', random_state=42)
```

A Linear Kernel: Why Use It?

- When the data is mostly linearly separable using a straight line, it functions effectively.
- It has a high computational efficiency.
- Overfitting is decreased by simpler models.

More complicated decision boundaries can be handled by using other kernel types, including polynomial or Radial Basis Function (RBF).

6. Trying out Various Kernels and Their Plots

The kernel selection has a big influence on SVM performance. Here's how several kernels operate:

(kernel='linear') Linear Kernel

- Ideal for data that can be separated linearly.
- Easy to use and computationally effective.
- Examples of Use Cases include text categorisation and spam detection.

Kernel for Radial Basis Function (RBF) (kernel='rbf').

- Ideal for data that is not linearly separable.
- Data is transformed into a higher-dimensional space using the Gaussian function.
- Gamma-controlled (higher values result in complicated decision limits).
- Example Use Case: Image classification, gene expression analysis.

Kernel = 'poly', a polynomial kernel

- Higher polynomial dimensions are used to map data.
- The degree parameter, such as degree=3 for cubic separation, regulates the complexity.
- Handwritten digit recognition (e.g., MNIST dataset) is an example of a use case.

(kernel='sigmoid') Sigmoid Kernel

- Comparable to the activation function of a neural network.
- Compared to polynomial or RBF kernels, they are rarely employed in practice.

We evaluate various kernels and contrast how well they perform:

- Linear Kernel
- RBF Kernel
- Polynomial Kernel (Degree = 3)
- Sigmoid Kernel

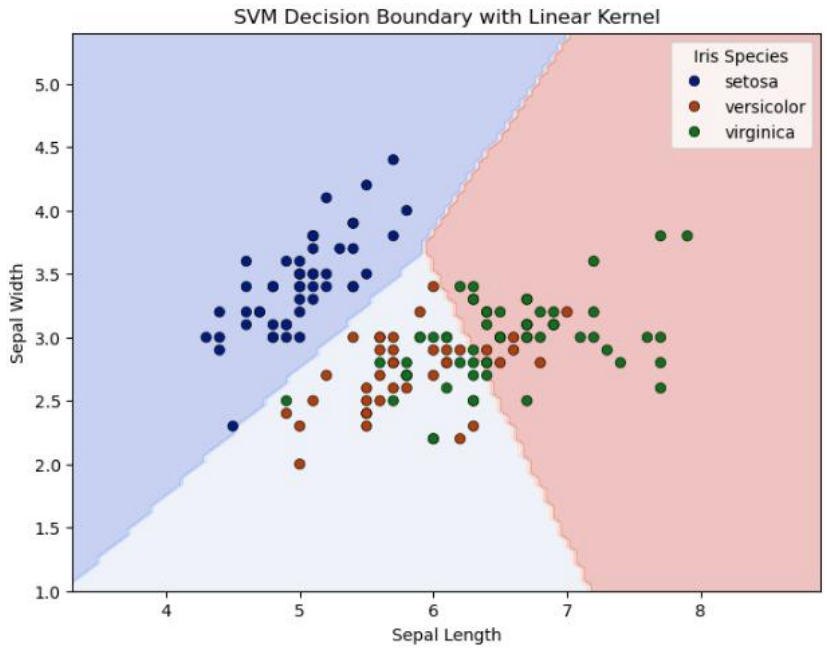
Examine Various Kernels on the Iris Dataset

[19]:
```
svm_rbf = SVC(kernel='rbf')
svm_poly = SVC(kernel='poly', degree=3)
svm_sigmoid = SVC(kernel='sigmoid')

svm_rbf.fit(X_train, y_train)
svm_poly.fit(X_train, y_train)
svm_sigmoid.fit(X_train, y_train)

print("RBF Kernel Accuracy:", svm_rbf.score(X_test, y_test))
print("Polynomial Kernel Accuracy:", svm_poly.score(X_test, y_test))
print("Sigmoid Kernel Accuracy:", svm_sigmoid.score(X_test, y_test))
```

```
RBF Kernel Accuracy: 1.0
Polynomial Kernel Accuracy: 1.0
Sigmoid Kernel Accuracy: 0.3
```

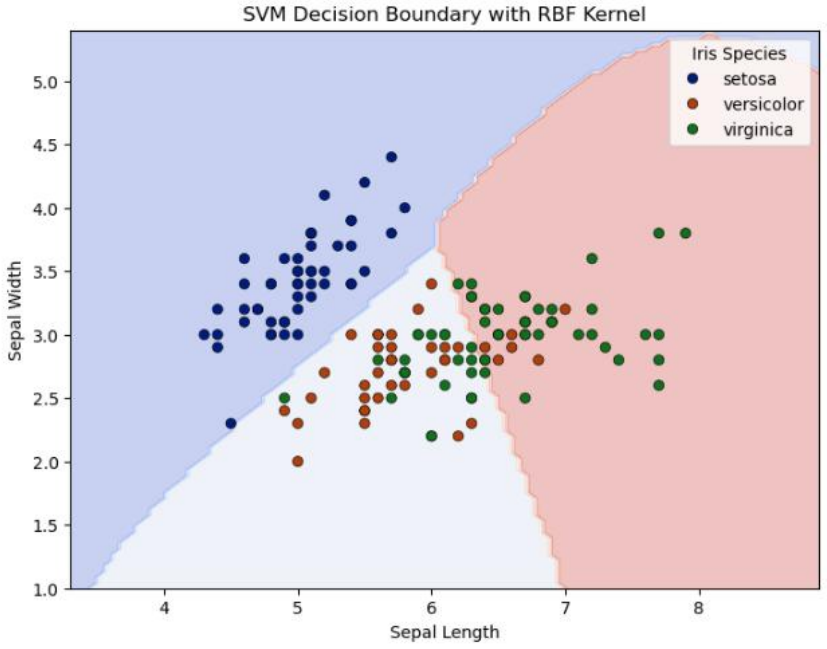
SVM Decision Boundary Plotting Using a Linear Kernel

The SVM decision boundary with a linear kernel offers a straight-line separation between classes, which makes it perfect for datasets where classes are linearly separable. The graph shows how an SVM model with a linear kernel is trained using the Sepal Length and Sepal Width features of the Iris dataset, and the resulting decision boundaries appear as straight lines dividing the feature space into distinct regions. This visualisation illustrates how a linear SVM efficiently finds the optimal hyperplane that maximises the margin between various iris species. Although the linear kernel performs well for simple patterns, it may not be able to handle more complex datasets that require non-linear decision boundaries. By balancing margin maximisation and misclassification tolerance, hyperparameters such as C (regularisation parameter), the boundary can be refined.



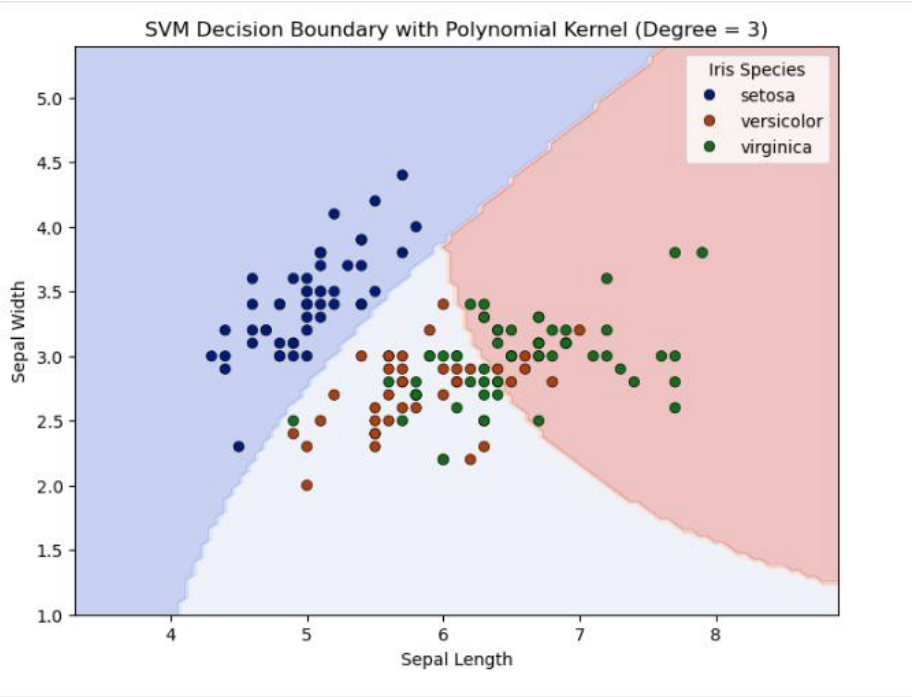
SVM Decision Boundary Plotting Using an RBF Kernel

For datasets with complicated feature interactions, the SVM decision boundary with an RBF (Radial Basis Function) kernel is quite successful since it produces a non-linear separation between classes. Curved decision boundaries that adjust to the distribution of the dataset are produced in the graph by training an SVM model with the RBF kernel using the Sepal Length and Sepal Width features from the Iris dataset. The RBF kernel can better identify data points and capture complex patterns than the linear kernel because it translates the input characteristics into a higher-dimensional space. The RBF kernel's adaptability aids in managing overlapping classes, however in order to prevent underfitting or overfitting, gamma—which regulates the impact of a single training example—and C—the regularisation parameter—must be carefully adjusted.



Plot the Polynomial Kernel SVM Decision Boundary (Degree = 3)

Compared to a linear SVM, the SVM decision boundary with a polynomial kernel (degree = 3) introduces non-linearity, enabling a more intricate division between classes. We can see how the polynomial kernel produces curved decision boundaries that more accurately depict the relationships within the data by using just two characteristics from the Iris dataset: Sepal Length and Sepal Width. The polynomial kernel is used to train the model, and decision zones are predicted and shown using a mesh grid. A potent tool for classification tasks with complex decision surfaces, the resulting figure demonstrates the versatility of polynomial SVMs in handling datasets when classes are not linearly separable.



7. Analysing the Model

Following training, we assess the model's accuracy using the test data.The accuracy of the model is determined by contrasting the anticipated and actual labels. An overall indicator of the model's ability to classify iris species is accuracy.

```
[10]: y_pred = svm_model.predict(X_test)
      accuracy = accuracy_score(y_test, y_pred)
      print(f"Accuracy: {accuracy * 100:.2f}%")

      Accuracy: 100.00%
```

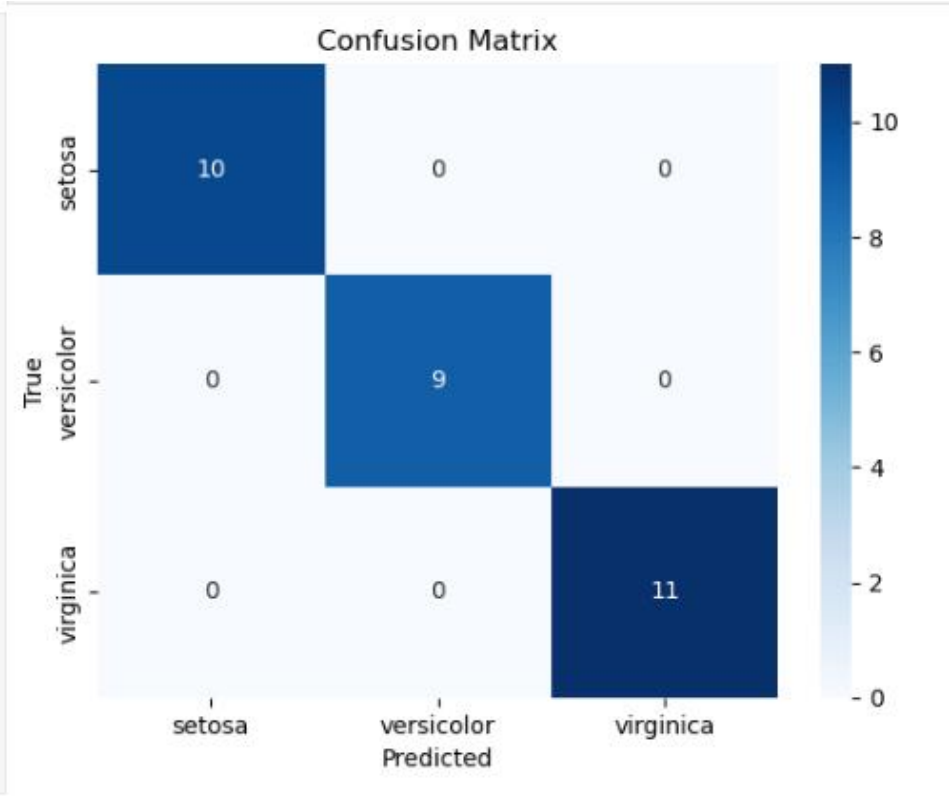
The model does well in identifying iris flowers when it has a high accuracy score.

8. Confusion Matrix Visualisation

Understanding the model's classification performance is aided by a confusion matrix.

```
[12]: cm = confusion_matrix(y_test, y_pred)
      sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=iris.target_names, yticklabels=iris.target_names)
      plt.xlabel('Predicted')
      plt.ylabel('True')
      plt.title('Confusion Matrix')
      plt.show()
```

The confusion matrix displays the proportion of accurate and inaccurate guesses for every class.



9. Findings and Understandings

- When it comes to iris species classification, the SVM model obtains great accuracy.
- The confusion matrix indicates potential misclassification regions for the model.
- Although the linear kernel successfully divides the classes, other kernels (polynomial, RBF) may be examined for enhancements.

10. Conclusion

- Using Support Vector Machines (SVM), we showed in this lesson how to categorize iris blossoms according to their characteristics. The main lessons are:
- SVM works well for classification tasks.
- When studying machine learning techniques, the Iris dataset is an excellent resource.
- Using an SVM with a linear kernel, we were able to achieve excellent accuracy.
- Model performance may be visualized with the use of a confusion matrix.

References:

1. Cortes, C., & Vapnik, V. (1995). "Support-vector networks." Machine Learning, 20(3), 273–297.
2. Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). "A training algorithm for optimal margin classifiers." Proceedings of the Fifth Annual Workshop on Computational Learning Theory, 144–152.
3. Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
4. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning. Springer.
5. Pedregosa, F., et al. (2011). "Scikit-learn: Machine Learning in Python." Journal of Machine Learning Research, 12, 2825–2830.
6. R. A. Fisher:“https://archive.ics.uci.edu/dataset/53/iris”

.