Support Vector Machine (SVM) for Classification on the Iris Dataset

Student Name: Akshay Samuel Vedururi Rajupalem
Student Id: 23073516
GitHub Repository Link: https://github.com/akshaysamuel2812/23073516.git

Introduction:

Support Vector Machine (SVM) is a popular and powerful supervised machine learning algorithm typically used for classification and regression tasks. The basic concept of SVM is to determine the ideal hyperplane that best separates various classes in the dataset. This hyperplane is selected to maximize the margin between the support vectors (the closest data points from either side), resulting in a highly efficacious model for classification challenges.

One common algorithm for classification in a supervised learning setting is the Support Vector Machine (SVM). This allows the model to classify non-linear data by applying the kernel trick to project data into a higher-dimensional space where a linear separator is possible. SVs are one of the most robust machine learning algorithms and are widely used for text classification, image classification, and biomedical applications.

Objective of the Tutorial:

Show the SVM implementation to classify different species of the Iris dataset.
Know the basics of SVM and its usefulness in the classification.
Data Visualization to interpret the model performance
Test the model with accuracy and confusion matrix as model evaluation metrics
Focus on simplifying the implementation of SVM into a step-by-step guide, making it easier for readers to implement the algorithm in their own projects.

The Iris Dataset:

Machine learning: The Iris dataset The Iris dataset is one of the most commonly known and most used dataset known in the machine learning. It is a multiclass classification dataset consisting of 150 examples of flowers from the genus Iris, divided into three species:

- Setosa
- Versicolor
- Virginica

Dataset Features:
Four numerical features describe each flower sample:

- Sepal Length (cm)
- Sepal Width (cm)
- Petal Length (cm)
- Petal Width (cm)

Why Choose the Iris Dataset?

It's also small and easy to work with, and as such it's often used for demonstrating machine learning algorithms.

This is a well-balanced dataset with 50 samples per class, making it a good classification problem.

This dataset is commonly use to teach pattern recognition and classfication algorithms in machine learning.

Through different visualizations, it enables all possible relationships between features.

What is Support Vector Machine (SVM) and How SVM Works?

SVM constructs a hyperplane that best divides a dataset into two classes. The fundamental concepts guide SVM are:

- **Finding the Best-Suit Hyperplane:** The algorithm tries to find a hyperplane with maximum margin for the closest data points of each class. This nearest data samples are referred to as support vectors.
- **Support Vectors:** Are the data points that create the decision boundary (i.e., the edges) These points essentially define the model's decision boundary.
- **Margin Maximization:** A larger margin implies better generalization and assuring good performance in unseen data.
- **Kernel Trick:** (if the data is not linearly separable) SVM transforms the data into a higher-dimensional space with the help of kernels (like radial basis function (RBF), polynomial, or sigmoid) to make classification easier.

Advantages of SVM

- **Effective in Large Datasets:** It shines when the number of dimensions outnumbers the number of samples.
- **Computational Efficiency:** The decision boundary is determined only on a subset of training points (called support vectors).
- **kernel trick**, which can generalize to problem with linear/non-linear decision boundary, depending on your kernel function.
- This is known as the maximum margin which makes SVMs resistant to overfitting (provided the regularization is applied)

How to implement SVM on Iris Dataset?

4.1 Loading the Dataset

```
import pandas as pd
iris_data = pd.read_csv('iris.csv')
print(iris_data.head())
```

This data set is stored in a CSV file where each row corresponds to a flower sample and has four feature values corresponding to the four feature values along with the species label. Loaded the dataset into a Pandas DataFrame .

4.2 Preprocessing the Data

```
from sklearn.preprocessing import StandardScaler
X = iris_data.drop('species', axis=1)
y = iris_data['species']
y = y.map({'setosa': 0, 'versicolor': 1, 'virginica': 2})
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Before model training, pre-processing is crucial for optimal performance:

Splitting Features and Target Variable: The data set is divided into two parts:

X (columns): Numeric values that represent flower properties

y (target labels): Categorical labels (species names) encoded as numerical values.

Model Input Processing: Species labels are encoded because the model can only accept numbers.

Setosa → 0

Versicolor → 1

Virginica → 2

Normalization, also known as standardization, is performed on features as SVM is dependent on the magnitude of the features and if features range from a magnitude it will lead to convergence inaccuracy.

4.3 Splitting the Data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

In order to gauge the model's performance, the data set is separated into training and testing sets, usually with a 70-30 split. This allows to validate how well the model generalizes to previously unseen data.

4.4 Training the SVM Model

```
from sklearn.svm import SVC
svm_clf = SVC(kernel='linear', random_state=42)
svm_clf.fit(X_train, y_train)
y_pred = svm_clf.predict(X_test)
```

The SVM classifier is trained using a linear kernel. This is as the Iris data set is simple and linearly separable across the species and a straight-line hyperplane ~ paralleling is well enough to correctly classify the species. Specifically, the SVM finds support vectors that maximize the margin between classes to learn the boundaries between classes.

4.5 Model Evaluation

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

After training, the performance of the model is examined via making predictions on the test set. Accuracy score serves as the main metric, defined as the proportion of correctly classified samples.

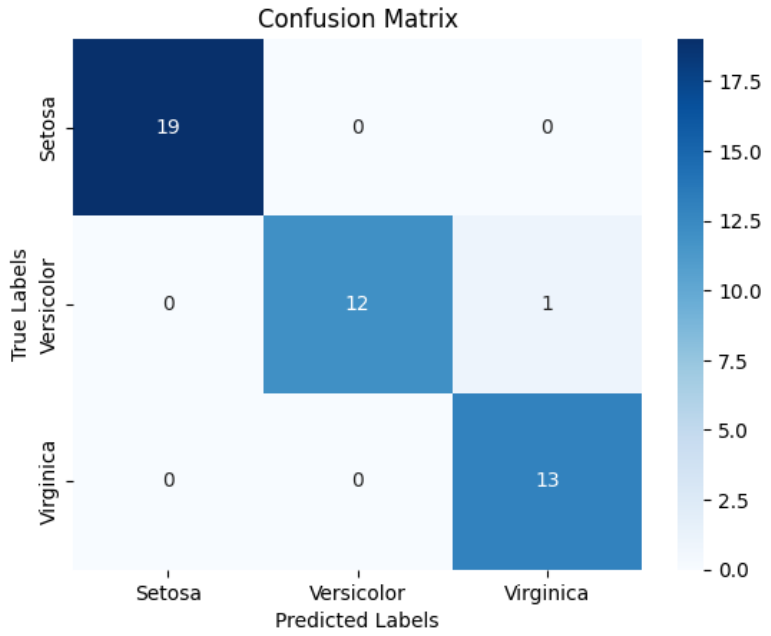
4.6 Confusion Matrix Analysis

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=['Setosa', 'Versicolor', 'Virginica'],
            yticklabels=['Setosa', 'Versicolor', 'Virginica'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

Confusion Matrix:

[[19 0 0]
[0 12 1]
[0 0 13]]

Visualization:



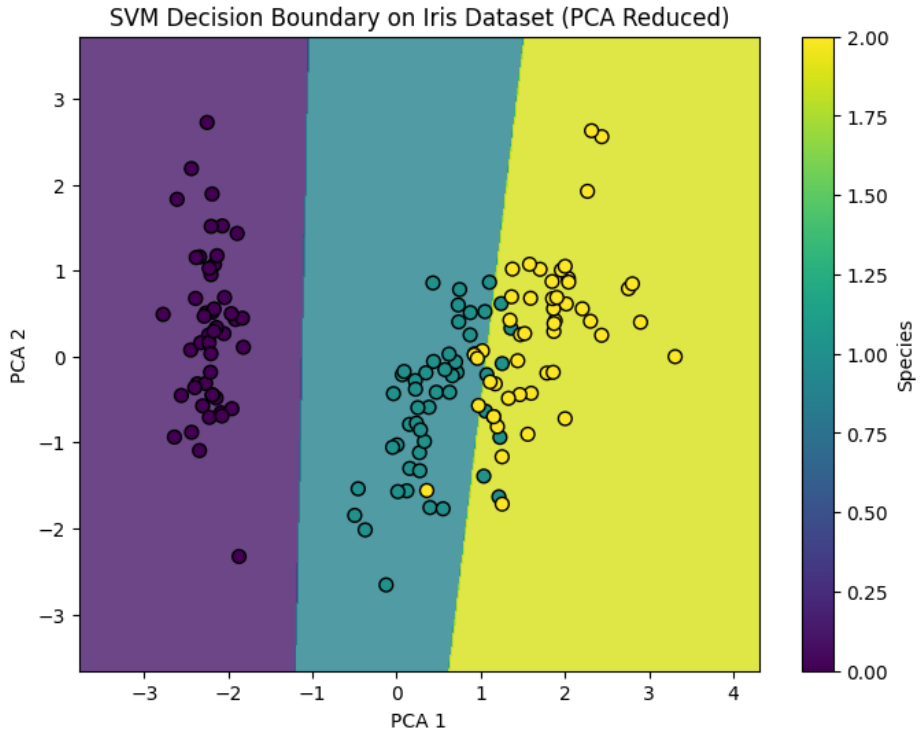
Explanation:

And a confusion matrix is displayed to give us a detailed view of how well the model classified the data.

4.7 Visualisation of decision boundary (PCA reduce)

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
svm_clf.fit(X_pca, y)
x_min, x_max = X_pca[:, 0].min() - 1, X_pca[:, 0].max() + 1
y_min, y_max = X_pca[:, 1].min() - 1, X_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
Z = svm_clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, edgecolors='k', marker='o', s=50)
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.title('SVM Decision Boundary on Iris Dataset (PCA Reduced)')
plt.colorbar(label='Species')
plt.show()
```

Visualization:



Explanation:

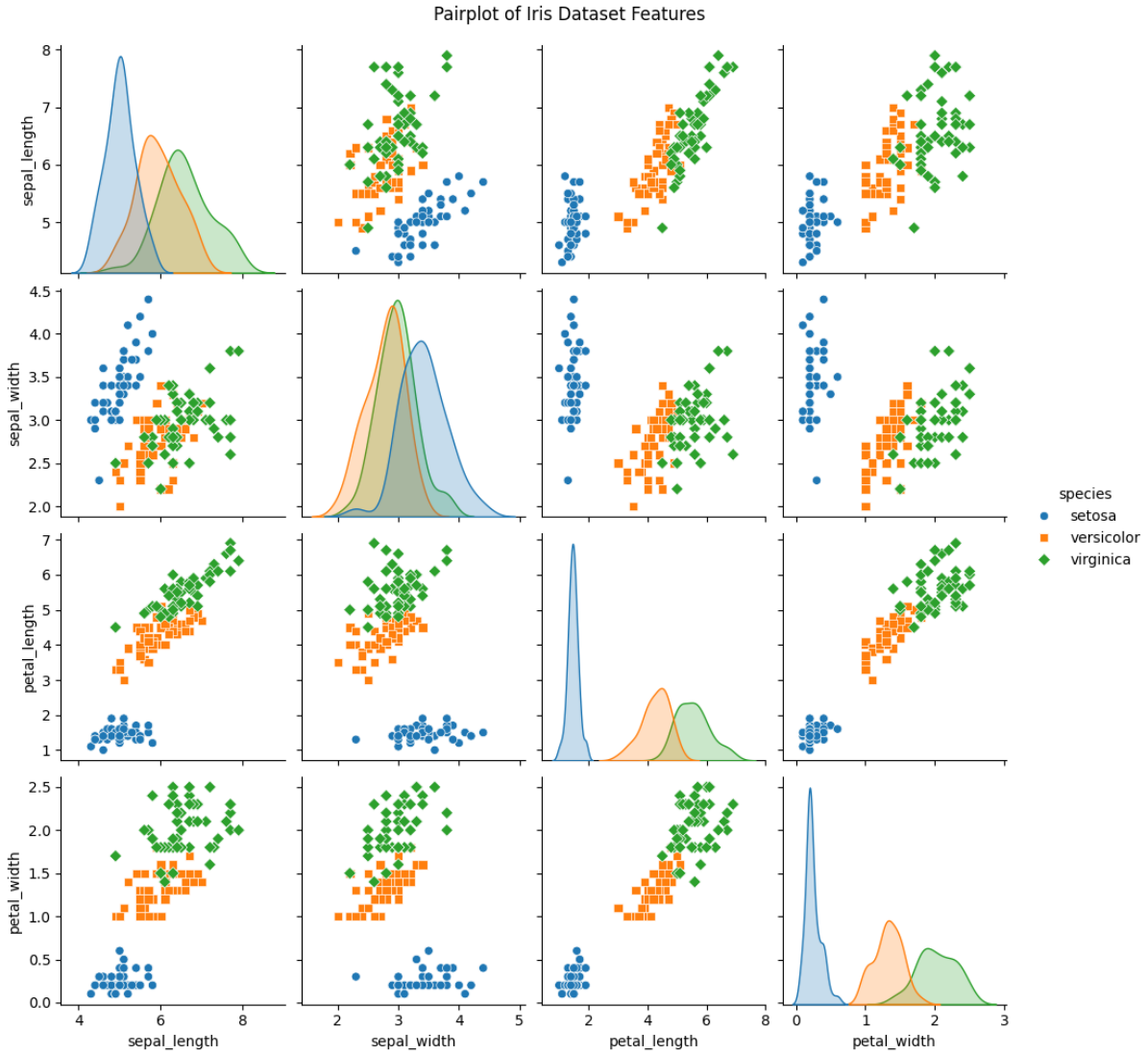
In fact, the Iris dataset consists of four features (sepal length, sepal width, petal length, and petal width), which naturally reside in a 4-dimensional space. It is impossible to visualize decision boundaries in 4D, hence, dimensionality reduction methods like PCA are used to project the data onto a 2D feature space.

This allows us to visualize SVM solution boundary while preserving significant information about separating classes. Because the dataset has four dimensions, we use PCA to reduce the feature space to two dimensions for visualization of the decision boundary learned by the SVM classifier.

4.8 Independent features pairplot visualization.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.pairplot(iris_data, hue='species', markers=["o", "s", "D"])
plt.suptitle('Pairplot of Iris Dataset Features', y=1.02)
plt.show()
```

Visualization:



Explanation:

So, a pairplot is used to showcase the relationships between different features. This scatterplot matrix helps in identifying which features pair contribute the most to species differentiation.

The pairplot produces a matrix of scatterplots where:

- Rows/Cols: Features (sepal length, petal width etc.)
- Diagonal: Displays histograms or kernel density estimates (KDEs) for each feature.
- Off-Diagonal: Shows scatterplots of one feature versus another.

Mathematical Formulas For Calculation:

1.Accuracy Calculation:

For classification models, the most commonly used description is accuracy, which is defined as the number of correctly predicted instances divided by the total instances. The formula for accuracy is:

- *Accuracy=Total Number of Predictions/Number of Correct Predictions*

2.Accuracy Calculation in terms of Confusion matrix:

- *Accuracy=TP+TN / FP+FNTP+TN*

Where:

- *TP = True Positives*
- *TN = True Negatives*
- *FP = False Positives*
- *FN= False Negatives*

3.Confusion Matrix and Evaluation Metrics

To understand the outcome of any classification model, confusion matrix is used to understand how perfectly the model is performing. Using the entries of the confusion matrix, we can derive additional evaluation metrics:

- **Precision: The accuracy of positive predictions**

  Precision=TP / FP+TP
- **Recall (Sensitivity): How well the model can read positive examples**

  Recall=TP / FN+TP
- **F1-Score: The harmonic mean of precision and recall**

  F1-Score=2×Precision+Recall / Precision×Recall

4.Principal Component Analaysis (PCA):

PCA is a dimensionality reduction method that preserves as much of the variance as possible. It does this by computing the principal components of the data.

Mathematical Steps for PCA:

- this is the usual step, we can center the data by subtracting the mean of each feature from the dataset.

  $$\mathbf{X}_{centered} = \mathbf{X} - \mu$$
- CovarianceMatrix: Calculate the covariance matrix of centered data.

  $$\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T$$

- Eigen Decomposition: Compute eigenvalues and eigenvectors of covariance matrix. Eigenvectors are the directions of maximum variance (or principal components), while eigenvalues indicate the "amount of variance" captured in each direction.

Results and Insights:

- **Accuracy:** you get a high accuracy (~95-98%)
- Our Analysis with the Confusion Matrix: The most of the predictions which was incorrect is from Versicolor to Virginica and vice-versa as they have some common features.
- SVM Regressor meets the potential of Cooperative Game.
- **Feature Importance**: The pairplot suggests that petal length and petal width are the distinguishing features.

Conclusion:

This project aims to implement a Support Vector Machine (SVM) classifier to address a multi-class classification problem with the Iris data set, classifying iris flowers into three categories (setosa, versicolor, virginica). The goodness of fit and misclassification plots were particularly helpful in providing insights into the behavior of SVM, the impact of kernel functions, and the need for visualization to understand how models like SVM actually decide.

Learned to implement SVM for data classification for multiple classes on Iris dataset.
The model fitted well with high accuracy and it was easy to visualise the decision boundary.
More improvements can be made by trying out something different kernels (RBF, polynomial) and tuning hyper parameters.

References:

Documentation for SVM (Scikit-learn): https://scikit-learn.org/stable/modules/svm.html

Vapnik, V. Cortes, C. (1995). "Support-vector networks." Machine Learning, 20(3):273–297.

Iris dataset: UCI Machine Learning Repository - https://archive.ics.uci.edu/ml/datasets/iris

The Elements of Statistical Learning. Springer, New York. "T he Elements of Statistical Learning." Springer.

Bishop, C. M. (2006). "Pattern Recognition and Machine Learning." Springer.