

Import Required Libraries

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
```

Prepare Dataset

Feature Scaling

```
X = np.array([[1, 2], [2, 3], [2, 1], [2, 4], [5, 4], [6, 3]])
y = np.array([0, 0, 0, 0, 1, 1])

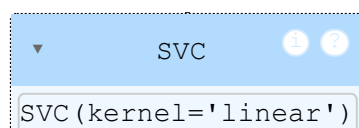
# Feature Scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Split Data

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

Initialize and Train SVM Model

```
svm_classifier = SVC(kernel='linear', C=1.0)
svm_classifier.fit(X_train, y_train)
```



Predictions and Evaluation

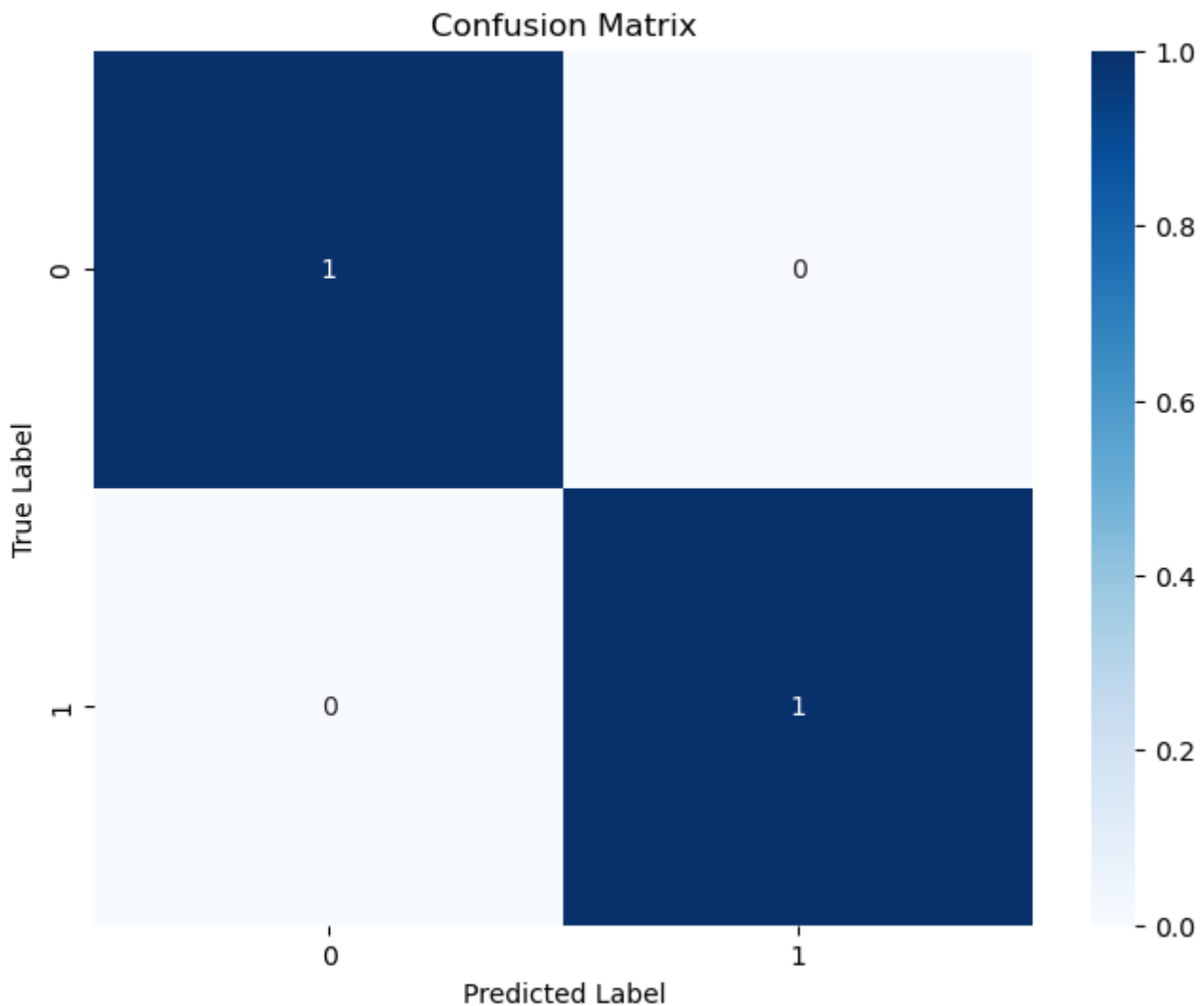
```
y_pred = svm_classifier.predict(X_test)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1
1	1.00	1.00	1.00	1
accuracy			1.00	2
macro avg	1.00	1.00	1.00	2
weighted avg	1.00	1.00	1.00	2

Confusion Matrix Visualization

```
plt.figure(figsize=(8, 6))
cm = confusion_matrix(y_test, y_pred, labels=[0, 1])
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()
```



Decision Boundary Visualization Function

```
def plot_decision_boundary(X, y, model, scaler):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                          np.arange(y_min, y_max, 0.1))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.figure(figsize=(10, 8))
    plt.contourf(xx, yy, Z, alpha=0.4)
    scatter = plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.RdBu,
                          edgecolor='black')
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary')
    plt.colorbar(scatter)
    plt.show()
```

Visualize Decision Boundary

```
plot_decision_boundary(X_scaled, y, svm_classifier, scaler)
```

