```
import pandas as pd
from sklearn.impute import SimpleImputer
from                                    _split
from
from                                    er, MinMaxScaler
from                                    ort, accuracy_score, confusion_matrix
from                                    r
from
impor
impor
impor
```



```
# Loa
data =

data.h
```

	LengthCm	PetalWidthCm	anomaly	Species_1	Species_2
	0.067797	0.041667	1	False	False
	0.067797	0.041667	1	False	False
	0.050847	0.041667	1	False	False
	0.084746	0.041667	1	False	False
	0.067797	0.041667	1	False	False

```
# 2. Split the dataset into features (X) and target (y)
X = data.iloc[:, :2].values
y = data.iloc[:, -1].values
```

```
# Encode target labels (species) to numeric values
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
```

```
# 3. Split the dataset into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
```

```
# 4. Initialize the SVM model with a non-linear kernel (RBF)
svm_model = SVC(kernel='rbf', gamma='auto')
```

```
# 5. Train the SVM model
svm_model.fit(X_train, y_train)
```

```
SVC
SVC(gamma='auto')
```

```
y_pred = svm_model.predict(X_test)
```

```
# Accuracy
accuracy = accuracy_score(y_test, y_pred) * 100
print(f"Accuracy: {accuracy:.4f}\n")
```

```
Accuracy: 100.0000
```

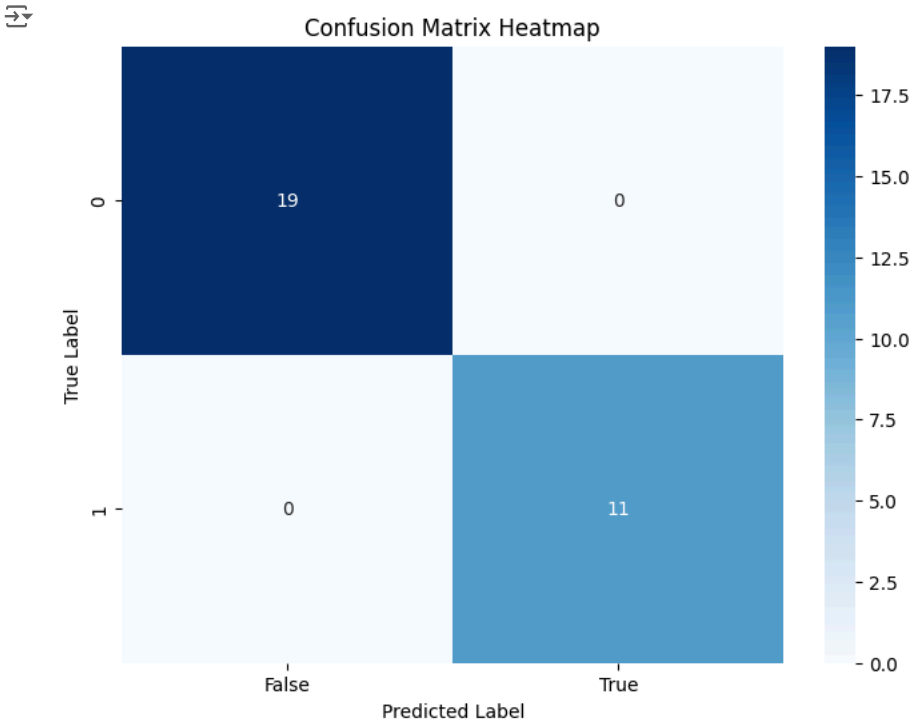
```
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)
print()
```

```
Confusion Matrix:
[[19  0]
 [ 0 11]]
```

```
# Classification report (formatted as a DataFrame)
report_dict = classification_report(y_test, y_pred, target_names=label_encoder.classes_, output_dict=True)
report_df = pd.DataFrame(report_dict).transpose()
print("Classification Report:\n", report_df)
```

```
Classification Report:
                precision  recall  f1-score  support
False               1.0     1.0       1.0     19.0
True                1.0     1.0       1.0     11.0
accuracy            1.0     1.0       1.0      1.0
macro avg           1.0     1.0       1.0     30.0
weighted avg        1.0     1.0       1.0     30.0
```

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=label_encoder.classes_, yticklabels=1)
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```

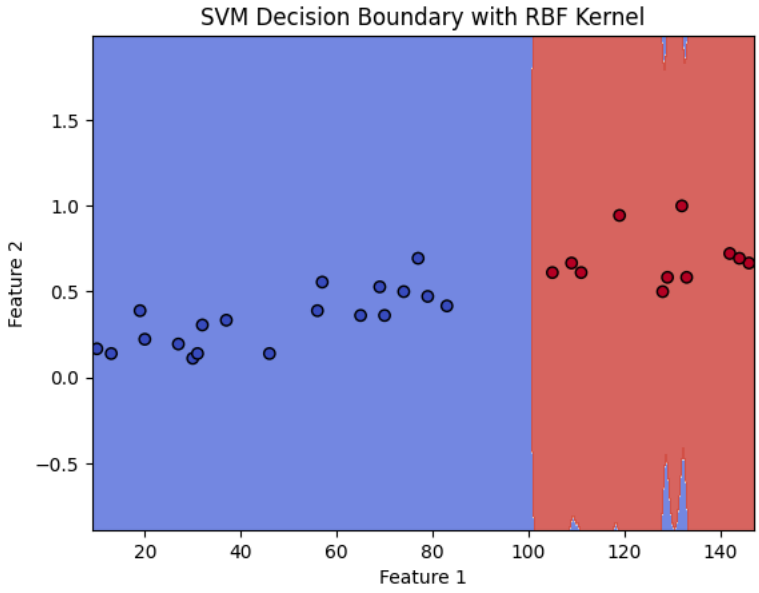



```
def plot_decision_boundary(X, y, model):
    h = 0.02  # step size in the mesh
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

    # Predict the class label for each point in the mesh
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Create the contour plot
    plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.coolwarm)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o', cmap=plt.cm.coolwarm)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')
    plt.title('SVM Decision Boundary with RBF Kernel')
    plt.show()
```

```
plot_decision_boundary(X_test, y_test, svm_model)
```


SVM Decision Boundary with RBF Kernel
Feature 2
Feature 1
1.5
1.0
0.5
0.0
−0.5
20
40
60
80
100
120
140