Support Vector Machine (SVM) Classifier using Different Kernels

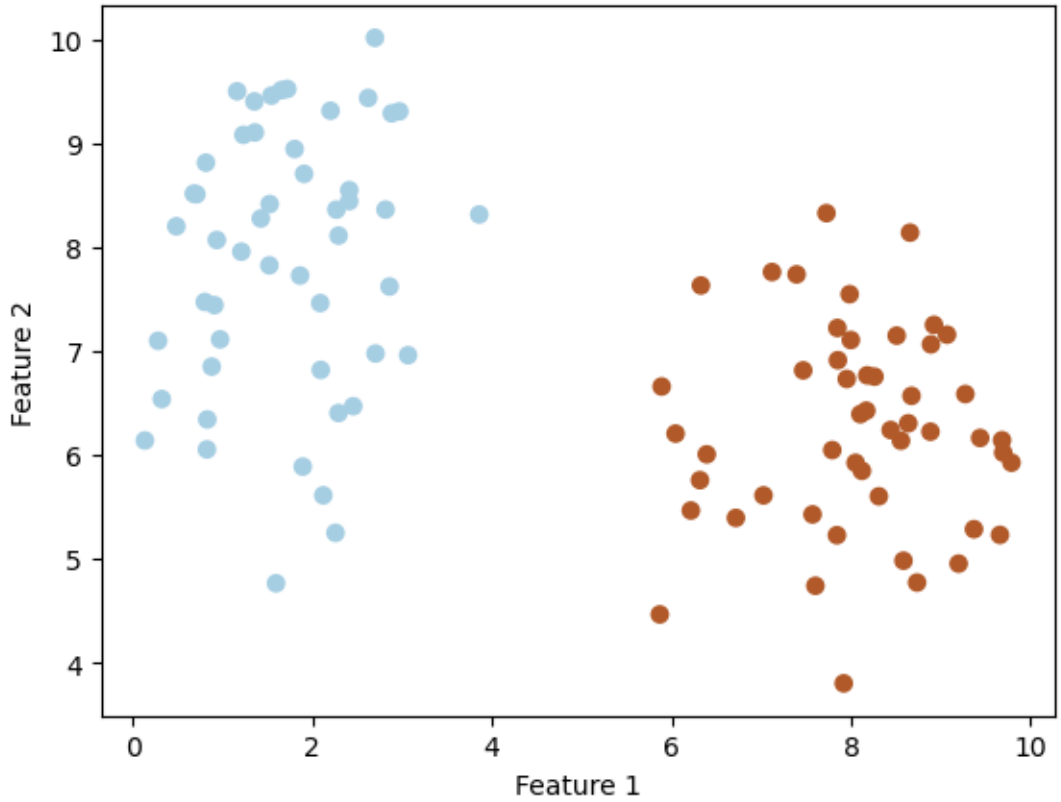
Overview

This notebook demonstrates the use of the Support Vector Machine (SVM) algorithm for binary classification on a synthetic 2D dataset. We explore how different kernels—**linear**, **polynomial**, and **RBF (Radial Basis Function)**—affect the decision boundary and performance of the model.

1. Dataset Generation

We use make_blobs from sklearn.datasets to generate a dataset with 2 features and 2 distinct centers for binary classification.

```
X, y = make_blobs(n_samples=100, n_features=2, centers=2, random_state=20)
```



🔍 **Plot Analysis:**
The data shows two clearly separated clusters, which makes it well-suited for classification using SVM. Since it's 2D, the decision boundaries can be visualized easily.

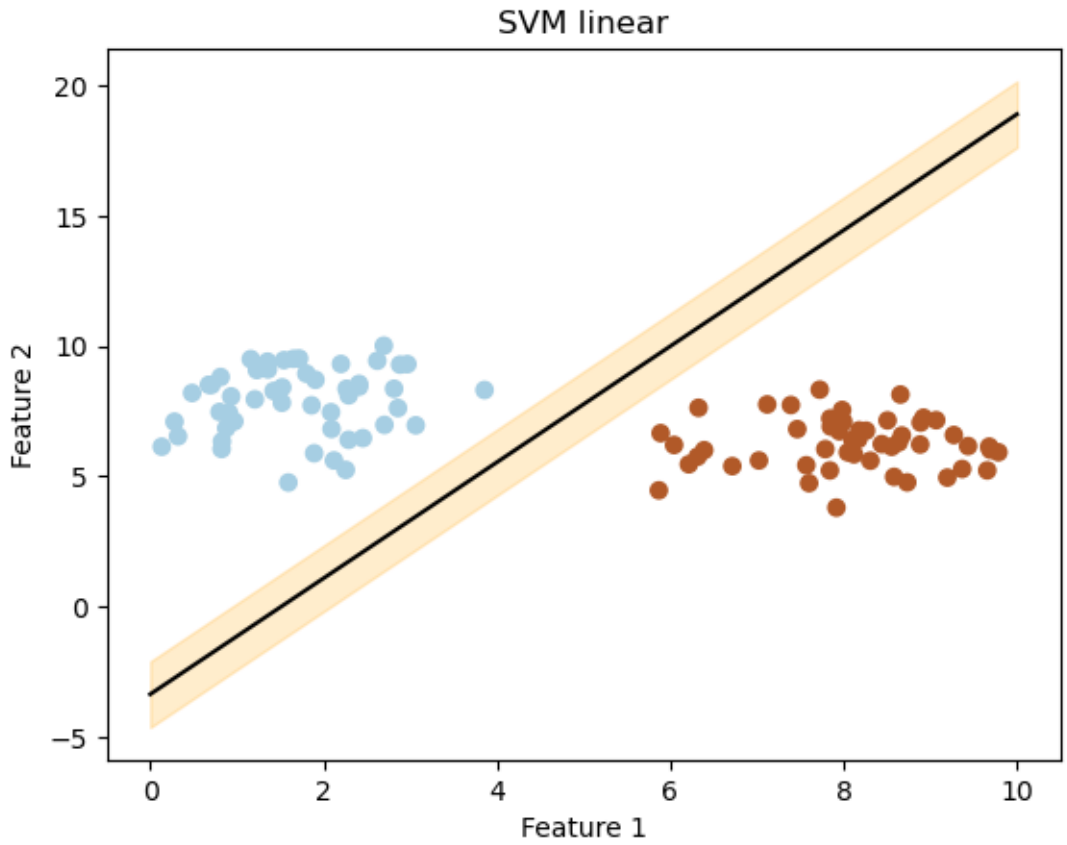
2. Train-Test Split

We split the dataset into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. SVM with Linear Kernel

```
svm_linear_model = SVC(kernel="linear", C=9)
svm_linear_model.fit(X_train, y_train)
```



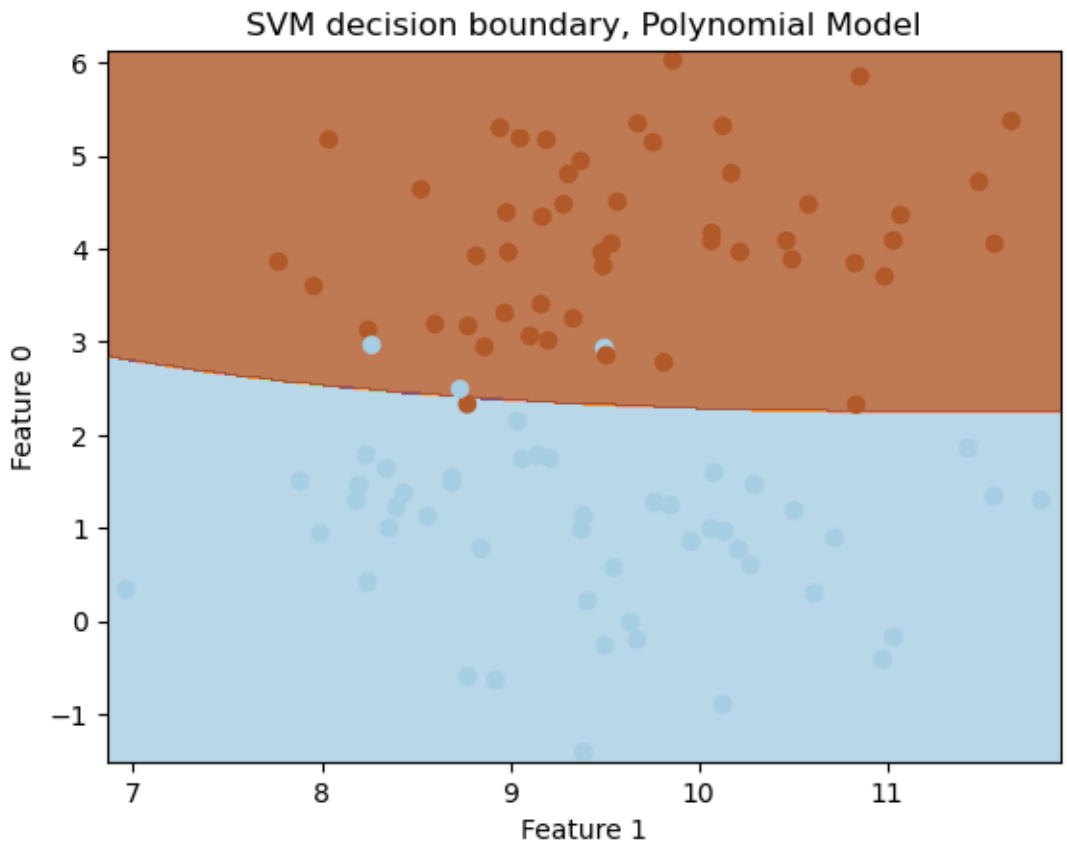
🔍 **Plot Analysis:**
The linear SVM draws a straight line between the two classes. Since the data is linearly separable, the classifier performs well.
The margin is visualized as a shaded region around the boundary. Its width is calculated using:

```
margin = 1 / np.sqrt(np.sum(svm_linear_model.coef_ ** 2))
```

This margin helps prevent overfitting by maximizing the distance between the support vectors and the boundary.

4. SVM with Polynomial Kernel

```
svm_poly_model = SVC(kernel="poly", degree=3)
svm_poly_model.fit(X_train, y_train)
```

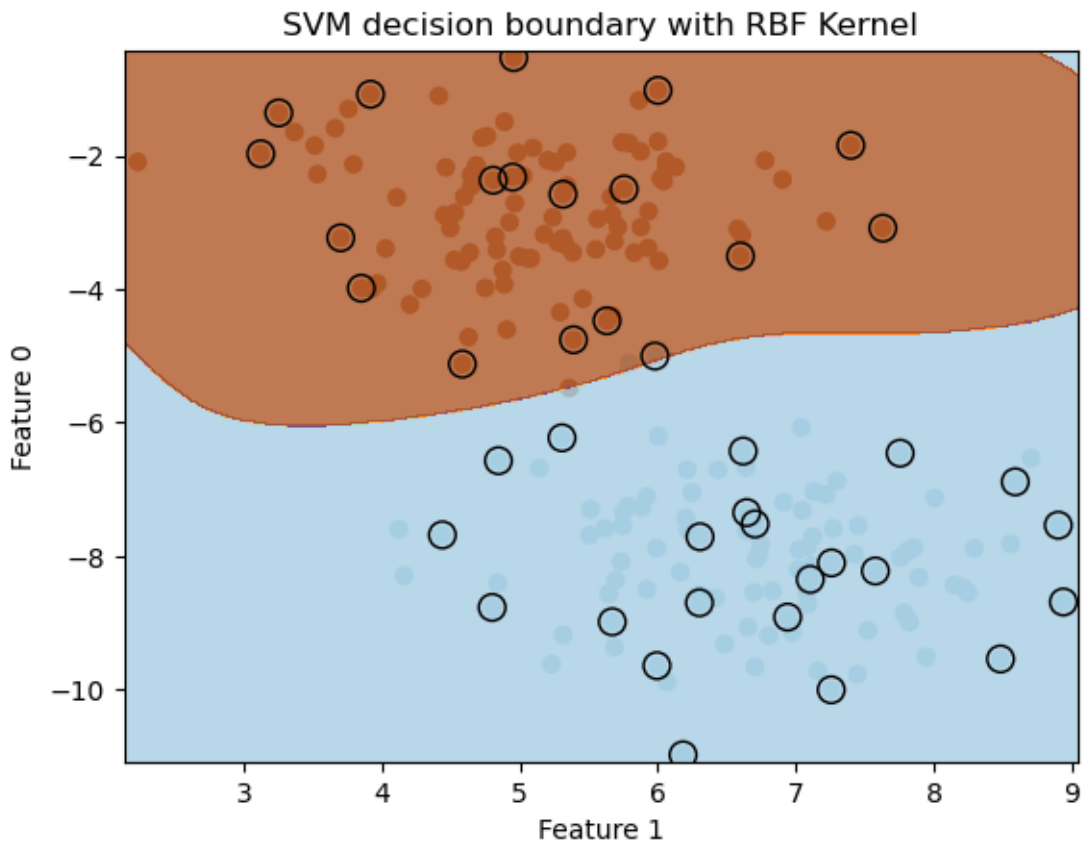


🔍 **Plot Analysis:**
The polynomial kernel introduces non-linearity to the boundary. For linearly separable data, this may add unnecessary complexity, but it's useful when the data has curved patterns. The classifier may still perform well but could show signs of **overfitting** if the degree is too high.

5. SVM with RBF Kernel

```
svm_rbf_model = SVC(kernel="rbf")

svm_rbf_model.fit(X_train, y_train)
```



🔍 **Plot Analysis:**
The RBF kernel draws non-linear boundaries and is very flexible. Even though the original data is linearly separable, RBF might still find a slightly different boundary due to its ability to model local variations. It's powerful but needs careful tuning of parameters like gamma to avoid overfitting.

6. Evaluation Metrics

For each model, evaluate performance on the test set:

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Classification Report:\n", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

Classification reports and confusion matrix for **Polynomial** model:

```
              precision    recall  f1-score   support

           0       1.00      0.85      0.92        13
           1       0.78      1.00      0.88         7

    accuracy                           0.90        20
   macro avg       0.89      0.92      0.90        20
weighted avg       0.92      0.90      0.90        20

----------------
[[11  2]
 [ 0  7]]
```

Classification reports and confusion matrix for **RBF** model:

```
0.95
--------------------
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        22
           1       0.94      0.94      0.94        18

    accuracy                           0.95        40
   macro avg       0.95      0.95      0.95        40
weighted avg       0.95      0.95      0.95        40

--------------------
[[21  1]
 [ 1 17]]
```