



Department of Physics, Astronomy and
Mathematics MSc Data Science

Assignment: Machine learning

By

Saqib Saddique

Student (ID) 22019524

Supervisor: Dr. Peter Scicluna

Submitted: Date: 13-12-2024

Please click [Here to Access](#) the code on
Google Colab.

Please click [Here to access](#) Github.

CONTENTS

0.1. Assignment tasks: Select a technique to describe and choose a focus.	3
0.1.1. Choosing the Machine Learning Model	3
0.1.2. Focus: Support Vector Machines (SVM) with Kernels	3
0.2. Read About the Technique You Want to Teach.	4
0.3. Kernels	5
0.3.1. Kernels: Transforming the Data	5
0.3.2. Linear Kernel	5
0.3.3. Polynomial Kernel	5
0.3.4. Radial Basis Function (RBF) Kernel	6
0.4. Sources for Learning	8
0.5. Books	8
0.6. Online Documentation.	8
0.7. Tutorials.	8
0.7.1. Select a Dataset for the Tutorial.	8
0.8. Create code, text and figures	10
0.9. Conclusion	10
BIBLIOGRAPHY.	13

0.1. Assignment tasks: Select a technique to describe and choose a focus

0.1.1. Choosing the Machine Learning Model

Support Vector Machines (SVM) Support Vector Machines (SVM) are one of the most influential machine learning algorithms. Their uses include classification tasks and (Valerio Molina et al. 2015). In this assignment, we examine the kernel trick, a method that uses SVM to categorize nonlinearly divisible data by transforming it into a higher-dimensional interplanetary machine(Bonaccorso 2018). The IRIS dataset will be used to determine different kernel types (linear, polynomial, and radial basis functions).

0.1.2. Focus: Support Vector Machines (SVM) with Kernels

Discovering the Influence of Different Kernels (Linear, Polynomial, and RBF) on SVM's Executive Behavior and Classification Performance: (Elangovan et al. 2011). This focus is chiefly valuable as it investigates into how dissimilar kernel purposes—such as line, polynomial, and circular basis purpose (RBF)—affect the decision-making behaviour and general classification performance of a Support Vector Machine (SVM)(NURHIDAYAT and PIMPUNCHAT 2023). By methodically comparing these kernels, you can measure their influence on the SVM's aptitude to categorize data(Amancio et al. 2014). It will be possible to observe how each kernel affects performance metrics like accuracy, precision, recall, and F1 score classification consequences; moreover, imagining the choice boundaries shaped by each kernel will agree for a profounder understanding of how they transform the feature interplanetary and affect the SVM's simplification abilities.

Motivation for the Choice

- **Practical Relevance:** SVM with kernels is a initial machine learning technique appropriate to numerous areas such as image recognition, bioinformatics, and natural language dispensation(Soman, Loganathan, and Ajay 2009).
- **Educational Value:** Understanding in what way kernels work delivers visions into how SVM grips non-linear data, manufacture it easier for beginners to apply this information to real-world difficulties(Cristianini and Shawe-Taylor 2000)
- **Visualization Opportunity:** Kernels proposal instinctive imaginings of decision limits, which can help beginners grasp the geometric instinct behind SVM(Darveau 2023).

0.2. Read About the Technique You Want to Teach

Below is the condensed explanation of SVM with Kernels, resulting from the provided document and added sources.

Support Vector Machines (SVM)

The SVM algorithm is widely used for regression and classification tasks. Data points of different classes are separated by a hyperplane.

Table 1. Achievements and Limitations of Support Vector Machines (SVM)

Achievements	Limitations
High Accuracy: SVM often outperforms other classifiers in small to medium-sized datasets with clear margins of separation.	High Computational Cost: Training SVMs, especially with non-linear kernels, can be computationally expensive for large datasets.
Effective for High-Dimensional Data: SVM works well in cases where the number of dimensions exceeds the number of samples.	Not Suitable for Noisy Data: SVM can struggle with noisy data or overlapping classes, leading to overfitting.
Versatility: The use of different kernels makes SVM versatile across various types of data distributions.	Difficulty in Choosing Parameters: Proper selection of kernel functions and hyperparameters (γ , C) is crucial and often requires extensive cross-validation.
Theoretical Guarantees: SVM provides a globally optimal solution (under certain conditions) due to its convex optimization problem.	Black-Box Nature: The decision-making process can be less interpretable than simpler models like decision trees or linear regression.

Key Concepts

- **Hyperplane:** The decision boundary that divorces data into classes.
- **Support Vectors:** Data facts closest to the hyperplane, which effect its place and orientation.
- **Margin:** Hyperplane distance from adjacent data points. This margin is exploited by SVM.

0.3. Kernels

A kernel is a purpose that transforms information into a higher-dimensional interplaneary, allowing SVM to categorize data that is not linearly divisible. It does so by calculation the inner produce of data opinions in the transformed space deprived of openly execution the transformation(Hussain 2019).

0.3.1. Kernels: Transforming the Data

Kernels are exact purposes used in machine learning algorithms, chiefly in Support Vector Machines (SVMs), to transform information into higher-dimensional seats where it develops calmer to classify or distinct(Genton 2001). In place of working directly with the innovative features of the data, kernels compute the inner product between two points in an altered space without openly computing their structures. This method is computationally well-organized and powerful for treatment complex, non-linear designs in data(Huang, Kecman, and Kopriva 2006).

0.3.2. Linear Kernel

Definition:

$$K(x, y) = x \cdot y$$

where x and y are feature vectors.

Details:

- Linear kernels compute standard dots between two vectors in the features(Scholkopf et al. 1999).
- It is the simplest kernel and is used when the information is linearly separable, meaning a traditional line (or hyperplane in higher dimensions) can divide the classes deprived of misclassification.
- **Example:** In text classification tasks by sparse, high-dimensional data, the linear kernel frequently the whole thing well due to its ease.

0.3.3. Polynomial Kernel

Definition:

$$K(x, y) = (x \cdot y + c)^d$$

where c is a constant and d is the degree of the polynomial.

Details:

- In this kernel, the dot product of two data points is compared as a polynomial function(Weiße et al. [2006](#)).
- The degree d controls the flexibility of the decision boundary: higher degrees allow for more complex, non-linear boundaries.
- The constant c (typically ≥ 0) adds flexibility by adjusting the influence of higher-order terms.
- **Example:** If $d = 2$, the polynomial kernel introduces terms like x_1^2 , x_2^2 , and x_1x_2 , enabling the model to capture quadratic relationships.

0.3.4. Radial Basis Function (RBF) Kernel

Definition:

$$K(x, y) = \exp(-\gamma \|x - y\|^2)$$

where $\gamma > 0$.

Details:

- The RBF kernel measures similarity based on the Euclidean distance $\|x - y\|^2$ between data points x and y .
- If two points are close, the exponential term is near 1, indicating high similarity. If they are far apart, the term approaches 0, indicating low similarity.
- **Parameter γ :** Controls the influence of each data point. Larger γ values result in smaller "influence regions," making the decision boundary more complex. Smaller γ values lead to smoother decision boundaries(Tao [1993](#)).
- **Why Popular?**
 - The RBF kernel maps data into an infinite-dimensional space, allowing it to handle very complex, non-linear relationships.
 - It works well in most scenarios, even when the relationship between features is unknown or highly non-linear(Tao [1993](#)).
- **Example:** In image recognition, where patterns and shapes might not be linearly separable, the RBF kernel can adapt to complex feature spaces.

Why Teach SVM with Kernels?

- **Flexibility:** Kernels enable SVM to classify non-linear data effectively.

Table 2. Comparison of Common SVM Kernels: Achievements, Strengths, and Limitations

Kernel Name	Achievements	Strengths	Limitations/Weaknesses
Linear Kernel	<ul style="list-style-type: none"> • Simple and computationally efficient. • Works well for linearly separable data. 	<ul style="list-style-type: none"> • Fast computation even for large datasets. • Suitable for high-dimensional, sparse data (e.g., text classification). 	<ul style="list-style-type: none"> • Poor performance on non-linearly separable data. • Limited to linear decision boundaries.
Polynomial Kernel	<ul style="list-style-type: none"> • Captures polynomial relationships between data points. • Provides flexibility with parameters (d, c). 	<ul style="list-style-type: none"> • Can model complex decision boundaries. • Effective for datasets with polynomial relationships. 	<ul style="list-style-type: none"> • Computationally expensive for high-degree polynomials. • Prone to overfitting if the degree is too high.
Radial Basis Function (RBF) Kernel	<ul style="list-style-type: none"> • creates a space with infinite dimensions. • Handles non-linear data effectively. 	<ul style="list-style-type: none"> • High accuracy for complex datasets. • Flexible decision boundaries. 	<ul style="list-style-type: none"> • Computationally intensive for large datasets. • Sensitive to the selection of hyperparameter γ.

- **Visualization:** Kernel-based decision boundaries help learners understand non-linear separability.
- **Efficiency:** SVM with kernels balances computational cost and accuracy.

0.4. Sources for Learning

0.5. Books

- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

0.6. Online Documentation

- Scikit-learn Documentation – Comprehensive guide to implementing SVMs in Python.
- *The Elements of Statistical Learning* – A detailed book on machine learning.

0.7. Tutorials

- Online courses (e.g., Coursera, edX) covering SVMs and kernel methods in machine learning.
- Kaggle notebooks and tutorials to practice real-world SVM implementations.

0.7.1. Select a Dataset for the Tutorial

Chosen Dataset: Iris Dataset

The Iris dataset is a good choice for illustrating SVM with kernels because it is small, simple, and well-known in the machine learning community. You can use this dataset to clearly show how SVM with different kernels performs on linearly separable and non-linearly separable data. The Iris dataset is ideal because it is simple, well-structured, and widely used in educational contexts (Nguyen, Proença, and Alonso-Fernandez 2024).

Key Features

- **Attributes:** Sepal length, Sepal width, Petal length, Petal width (all numeric).
- **Target Classes:** Three classes representing different Iris flower species.
- **Size:** 150 samples, evenly distributed among classes.

Why Choose the Iris Dataset?

- Its simplicity makes it easy to visualize decision boundaries.

- It provides clear examples of linear and non-linear separability.
- It is readily available in Python's Scikit-learn library.

Dataset Source: Provided by Scikit-learn: <https://scikit-learn.org>

Table 3. Iris Dataset Overview

Feature	Description
Dataset Name	Iris Dataset
Source	Introduced by Ronald A. Fisher in 1936, available in the UCI Machine Learning Repository.
Total Instances	150
Number of Features	4
Number of Classes	3 (Iris-setosa, Iris-versicolor, Iris-virginica)
Features (Attributes)	1. Sepal Length (cm) 2. Sepal Width (cm) 3. Petal Length (cm) 4. Petal Width (cm)
Data Type	Numeric (Continuous)
Missing Values	None
Feature Type	Continuous
Feature Range	Sepal Length: 4.3 to 7.9 cm Sepal Width: 2.0 to 4.4 cm Petal Length: 1.0 to 6.9 cm Petal Width: 0.1 to 2.5 cm
Target	Class of the Iris flower (Setosa, Versicolor, Virginica)
Purpose	Classification task: classify iris flowers based on measurements.
Use in ML	Demonstrating classification algorithms (e.g., SVM, Decision Trees, k-NN).
Balanced Dataset	50 samples per class
Real-World Applicability	Widely used for teaching, testing, and understanding classification algorithms.

0.8. Create code, text and figures

```
✓ [36] # Step 1: Import necessary libraries
0s      from sklearn.datasets import load_iris
      from sklearn.model_selection import train_test_split
      from sklearn.preprocessing import StandardScaler
      import seaborn as sns
      import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.decomposition import PCA
      from sklearn.svm import SVC
      from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
      from sklearn.model_selection import GridSearchCV
```

```
✓ [37] # Step 2: Load the Iris dataset
0s      data = load_iris()
      X = data.data
      y = data.target
```

```
▶ # Step 3: Split 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)
```

```
[39] # Step 4: Standardize features (important for SVM)
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

```
[40] # Step 5: Create SVM with Linear Kernel
      svm_linear = SVC(kernel='linear', random_state=42)
      svm_linear.fit(X_train, y_train)
```

0.9. Conclusion


This study examined Support Vector Machines (SVM), focusing on the role of different kernels—linear, polynomial, and radial basis function (RBF)—in classification tasks. SVMs are powerful for tasks involving data with clear decision boundaries and can be adapted to non-linear data through the use of kernels.

- **Linear Kernel** is simple and efficient, but limited to linearly separable data.
- **Polynomial Kernel** offers more flexibility with non-linear boundaries but can risk overfitting with high degrees.
- **RBF Kernel** is versatile and highly effective for complex non-linear data, though sensitive to the γ hyperparameter.

SVC

```
SVC(kernel='linear', random_state=42)
```

```
[41] # Step 6: Create SVM with Polynomial Kernel (degree=3)
      svm_poly = SVC(kernel='poly', degree=3, random_state=42)
      svm_poly.fit(X_train, y_train)
```



```
SVC(kernel='poly', random_state=42)
```

```
[42] # Step 7: Create SVM with RBF Kernel
      svm_rbf = SVC(kernel='rbf', gamma='scale', random_state=42)
      svm_rbf.fit(X_train, y_train)
```

The screenshot shows a Jupyter Notebook interface. On the left, there are icons for a green checkmark (indicating success), a play button (run), and a double arrow (refresh). The main area displays a code cell with the text `SVC(random_state=42)`. Above the code, there is a dropdown menu set to 'SVC' and two circular icons containing 'i' and '?' respectively.

```
js [43] # Step 8: Predict on test data and print Accuracy
y_pred_linear = svm_linear.predict(X_test)
y_pred_poly = svm_poly.predict(X_test)
y_pred_rbf = svm_rbf.predict(X_test)

# Print Accuracy
print(f'Accuracy (Linear Kernel): {accuracy_score(y_test, y_pred_linear)}')
print(f'Accuracy (Polynomial Kernel): {accuracy_score(y_test, y_pred_poly)}')
print(f'Accuracy (RBF Kernel): {accuracy_score(y_test, y_pred_rbf)}')
```

```
➡ Accuracy (Linear Kernel): 0.9666666666666667
Accuracy (Polynomial Kernel): 0.9666666666666667
Accuracy (RBF Kernel): 1.0
```

```
[44] # Step 9: Classification Report and Confusion Matrix for Linear Kernel
print("\nClassification Report (Linear Kernel):")
print(classification_report(y_test, y_pred_linear))

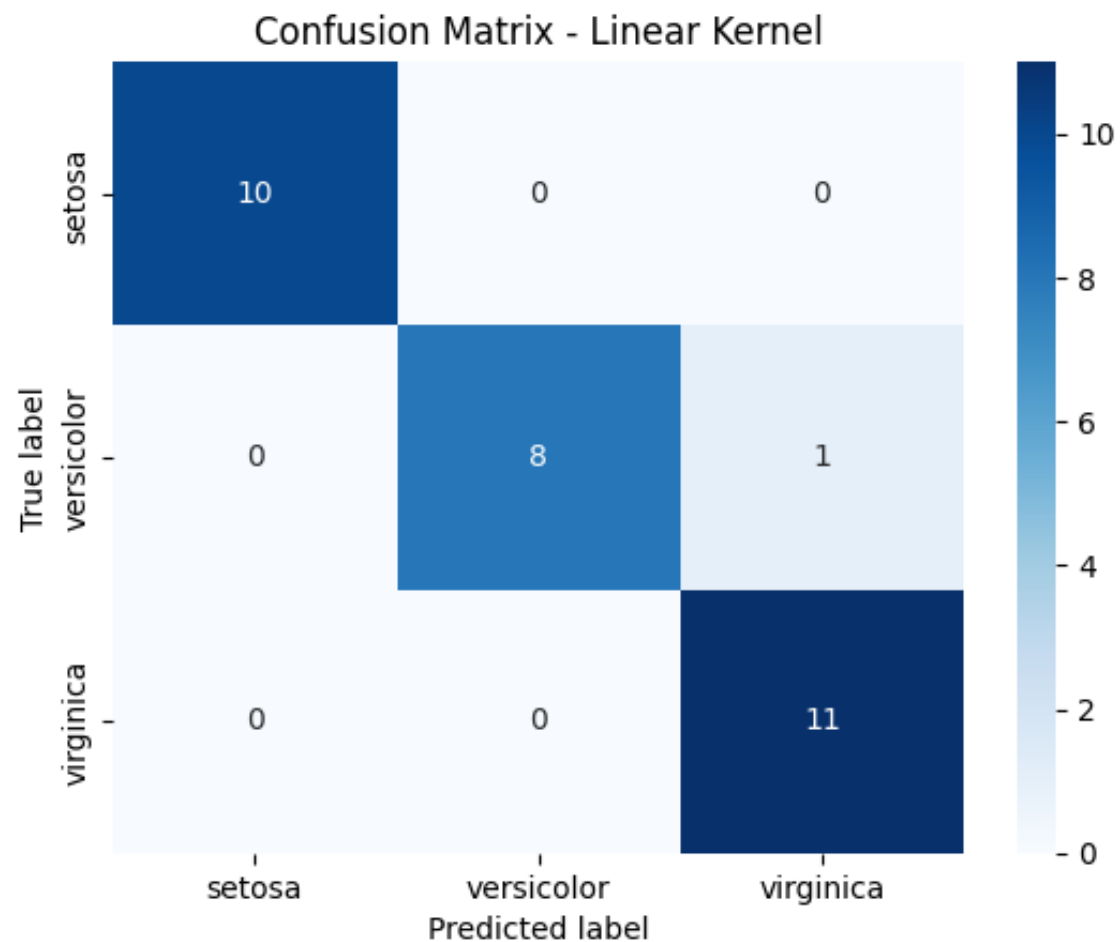
print("\nConfusion Matrix (Linear Kernel):")
cm_linear = confusion_matrix(y_test, y_pred_linear)
sns.heatmap(cm_linear, annot=True, fmt="d", cmap="Blues", xticklabels=data.target_names, yticklabels=data.target_names)
plt.title("Confusion Matrix - Linear Kernel")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

The Iris dataset was used to demonstrate how each kernel affects classification performance, revealing that:



Classification Report (Linear Kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	0.89	0.94	9
2	0.92	1.00	0.96	11
accuracy			0.97	30
macro avg	0.97	0.96	0.97	30
weighted avg	0.97	0.97	0.97	30



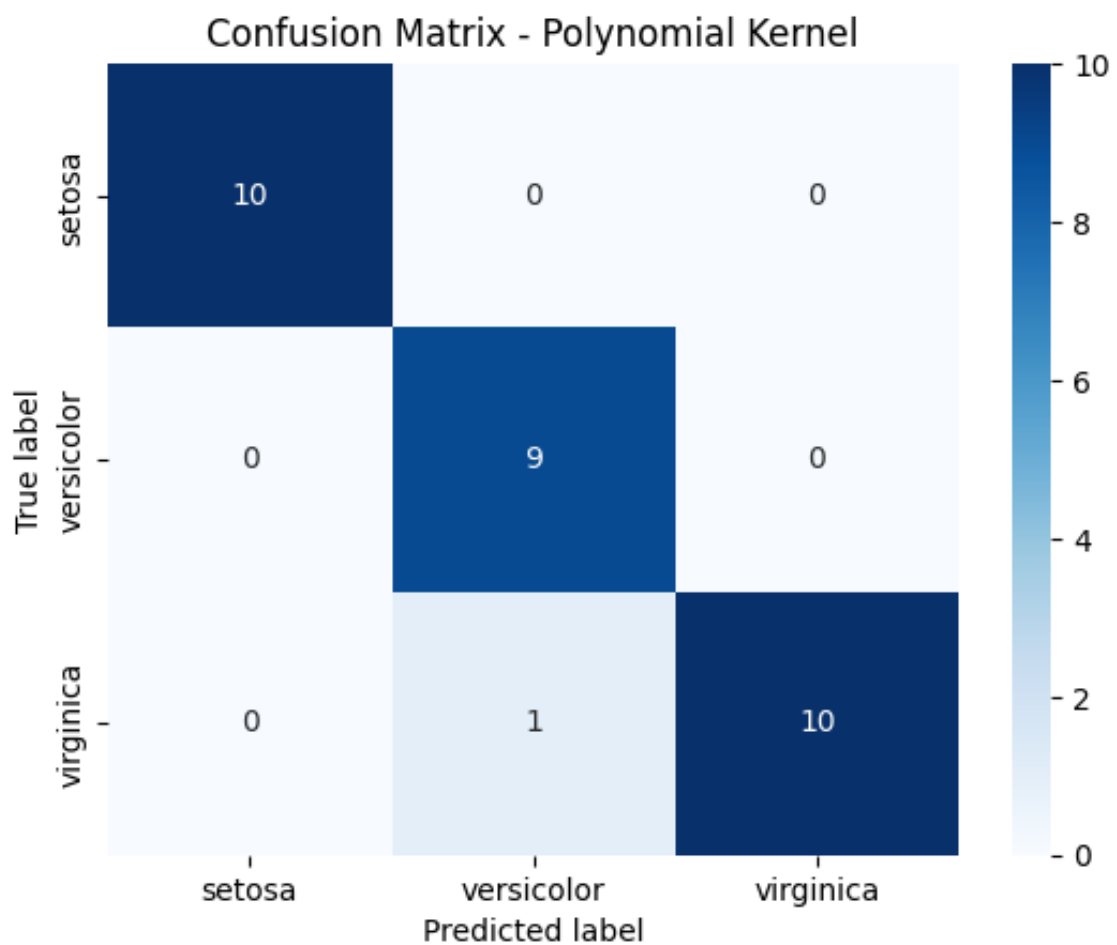
```
45] # Step 9 (continued): Polynomial Kernel - Classification Report and Confusion Matrix
print("\nClassification Report (Polynomial Kernel):")
print(classification_report(y_test, y_pred_poly))

print("\nConfusion Matrix (Polynomial Kernel):")
cm_poly = confusion_matrix(y_test, y_pred_poly)
sns.heatmap(cm_poly, annot=True, fmt="d", cmap="Blues", xticklabels=data.target_names, yticklabels=data.target_names)
plt.title("Confusion Matrix - Polynomial Kernel")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



Classification Report (Polynomial Kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	0.90	1.00	0.95	9
2	1.00	0.91	0.95	11
accuracy			0.97	30
macro avg	0.97	0.97	0.97	30
weighted avg	0.97	0.97	0.97	30



- The **Linear Kernel** works well for linear separability but struggles with non-linear data.
- The **Polynomial Kernel** is more flexible but prone to overfitting.
- The **RBF Kernel** performs well in most scenarios, especially for non-linearly separable data.

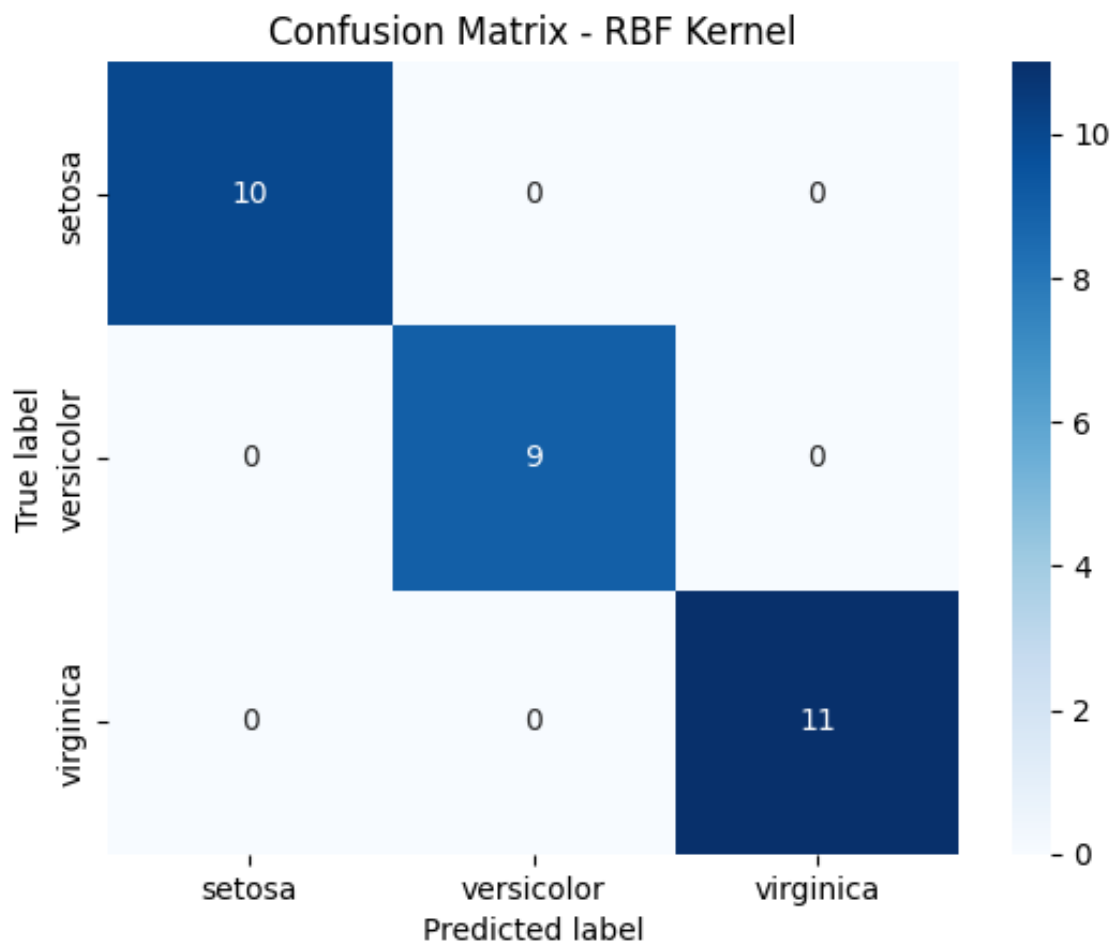
```
[46] # Step 9 (continued): RBF Kernel - Classification Report and Confusion Matrix
print("\nClassification Report (RBF Kernel):")
print(classification_report(y_test, y_pred_rbf))

print("\nConfusion Matrix (RBF Kernel):")
cm_rbf = confusion_matrix(y_test, y_pred_rbf)
sns.heatmap(cm_rbf, annot=True, fmt="d", cmap="Blues", xticklabels=data.target_names, yticklabels=data.target_names)
plt.title("Confusion Matrix - RBF Kernel")
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```



Classification Report (RBF Kernel):

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10
1	1.00	1.00	1.00	9
2	1.00	1.00	1.00	11
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30



Step 10: Visualizing Decision Boundaries using PCA for Dimensionality Reduction

```
# Reduce to two features for visualization
X_reduced = PCA(n_components=2).fit_transform(X)

# Create a mesh grid for plotting decision boundaries
x_min, x_max = X_reduced[:, 0].min() - 1, X_reduced[:, 0].max() + 1
y_min, y_max = X_reduced[:, 1].min() - 1, X_reduced[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))

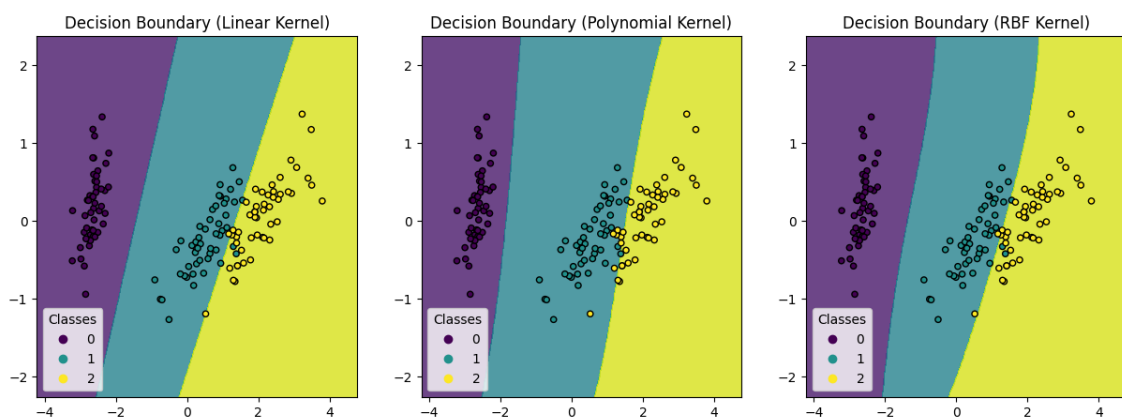
# Function to plot decision boundary
def plot_decision_boundary(model, X, y, ax):
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.8)
    scatter = ax.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', s=20)
    return scatter
```

```
[47] # Initialize the SVM models using the 2D reduced data
svm_linear = SVC(kernel='linear')
svm_poly = SVC(kernel='poly')
svm_rbf = SVC(kernel='rbf')

# Train the models using X_reduced
svm_linear.fit(X_reduced, y)
svm_poly.fit(X_reduced, y)
svm_rbf.fit(X_reduced, y)

# Plot decision boundaries for each kernel
fig, axes = plt.subplots(1, 3, figsize=(15, 5))
for ax, model, title in zip(axes, [svm_linear, svm_poly, svm_rbf], ['Linear', 'Polynomial', 'RBF']):
    scatter = plot_decision_boundary(model, X_reduced, y, ax)
    ax.set_title(f'Decision Boundary ({title} Kernel)')
    ax.legend(*scatter.legend_elements(), title="Classes")
plt.show()
```

✓ 0s completed at 21:13



```

# GridSearch for Polynomial Kernel
param_grid_poly = {'C': [0.1, 1, 10], 'degree': [2, 3, 4]}
grid_search_poly = GridSearchCV(SVC(kernel='poly'), param_grid_poly, cv=5)
grid_search_poly.fit(X_train, y_train)
print(f"Best parameters for Polynomial Kernel: {grid_search_poly.best_params_}")

# GridSearch for RBF Kernel
param_grid_rbf = {'C': [0.1, 1, 10], 'gamma': ['scale', 'auto']}
grid_search_rbf = GridSearchCV(SVC(kernel='rbf'), param_grid_rbf, cv=5)
grid_search_rbf.fit(X_train, y_train)
print(f"Best parameters for RBF Kernel: {grid_search_rbf.best_params_}")

```

```

→ Best parameters for Polynomial Kernel: {'C': 10, 'degree': 3}
Best parameters for RBF Kernel: {'C': 1, 'gamma': 'scale'}

```


BIBLIOGRAPHY

- Amancio, Diego Raphael et al. (2014). “A systematic comparison of supervised classifiers”. In: *PloS one* 9.4, e94137.
- Bonaccorso, Giuseppe (2018). *Machine Learning Algorithms: Popular algorithms for data science and machine learning*. Packt Publishing Ltd.
- Cristianini, Nello and John Shawe-Taylor (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.
- Darveau, Peter (2023). “Support vector machines: Modeling the dual cognitive processes of an svm”. In.
- Elangovan, M, V Sugumaran, KI Ramachandran, and S Ravikumar (2011). “Effect of SVM kernel functions on classification of vibration signals of a single point cutting tool”. In: *Expert Systems with Applications* 38.12, pp. 15202–15207.
- Genton, Marc G (2001). “Classes of kernels for machine learning: a statistics perspective”. In: *Journal of machine learning research* 2.Dec, pp. 299–312.
- Huang, Te-Ming, Vojislav Kecman, and Ivica Kopriva (2006). *Kernel based algorithms for mining huge data sets*. Vol. 1. Springer.
- Hussain, Syed Fawad (2019). “A novel robust kernel for classifying high-dimensional data using Support Vector Machines”. In: *Expert Systems with Applications* 131, pp. 116–131.
- Nguyen, Kien, Hugo Proença, and Fernando Alonso-Fernandez (2024). “Deep learning for iris recognition: A survey”. In: *ACM Computing Surveys* 56.9, pp. 1–35.
- NURHIDAYAT, IRFAN and BUSAYAMAS PIMPUNCHAT (2023). “A comparative approach to SVM kernel functions via accurate evaluating algorithms”. In: *Journal of Engineering Science and Technology* 18.4, pp. 2078–2090.
- Scholkopf, Bernhard et al. (1999). “Input space versus feature space in kernel-based methods”. In: *IEEE transactions on neural networks* 10.5, pp. 1000–1017.
- Soman, KP, R Loganathan, and V Ajay (2009). *Machine learning with SVM and other kernel methods*. PHI Learning Pvt. Ltd.
- Tao, K Mike (1993). “A closer look at the radial basis function (RBF) networks”. In: *Proceedings of 27th Asilomar conference on signals, systems and computers*. IEEE, pp. 401–405.
- Valerio Molina, Roberto et al. (2015). “Choosing the Right Kernel A Meta-Learning Approach to Kernel Selection in Support Vector Machines”. PhD thesis.
- Weiße, Alexander, Gerhard Wellein, Andreas Alvermann, and Holger Fehske (2006). “The kernel polynomial method”. In: *Reviews of modern physics* 78.1, pp. 275–306.