Machine Learning Tutorial: Exploring the Impact of Kernels on Support Vector Machines (SVM)

Author: Muhammad Rizwan Khalid

Id 24083618

Github: https://github.com/Rizwan1919913/Machine-Learning

Date: December 11, 2025

Contents

1. Introduction 2

2. Background on Support Vector Machines 2

3. The Role of Kernels in SVM 3

Figure 1 - Basic SVM Hyperplane. 5

4. Dataset Selection: The Make Moons Dataset 5

Figure 2 - Make Moons Dataset Visualization. 6

5. Implementation in Python 6

6. Results and Analysis 7

Figure 4 - Decision Boundary for Linear Kernel. 8

Figure 5 - Decision Boundary for Polynomial Kernel. 8

Figure 6 - Decision Boundary for RBF Kernel. 9

Figure 7 - Decision Boundary for Sigmoid Kernel. 9

7. Hyperparameter Tuning and Best Practices 9

8. Ethical Considerations and Real-World Applications 9

9. Conclusion 10

10. References 10

1. Introduction

SVM is one of the foundations of supervised learning algorithms that are mostly applied in classification and regression. In their original form, SVMs have been created by Vapnik and his associates back in the 1990s to classify the best line roughly the optimal hyperplane that maximizes the distance between the various categories of data points (Vapnik, 1995). The methodology does not only improve the classification accuracy but also gives resistance against overfitting especially when dealing with high-dimensional data. The main point of interest in this tutorial is to investigate the effect of use of different kernel functions on the performance and behavior of SVMs particularly in the case of non-linearly separable data.

The importance of kernels in SVM should not be exaggerated where the kernels help the algorithm to implicitly map data into higher dimensions enabling more complex and non-linear relationships which are otherwise difficult to deal with using linear classifiers to be dealt with. Using this tutorial, one will discuss four major kernels, which include linear, polynomial, radial basis function (RBF) and sigmoid. They will be run on a synthetic dataset to demonstrate their unique effects on decision limits and the entire model effectiveness.

Based on personal observations acquired throughout the course, kernels struck me with their exceptional combination of the techniques of the linear algebra with more sophisticated non-linear modelling, the way a simple tool turns into a complex artifact and applies it to solving the problem. The tutorial includes Python-based examples with the use of scikit-learn library along with code snippets, visual figures, and performance analysis. It aims at enabling the readers to apply SVM with suitable kernels to their own machine learning systems to create an appreciation of why some kernels like RBF often perform better in enticing detailed data patterns than other kernels.

Moreover, this exploration fits more broad learning objectives, such as showing the ability to apply neural computation models and advanced machine learning techniques. At the end of it, users will not only value the flexibility in SVM, but they will already be in a position to use it with responsibility, taking into account the possible biases and ethical consequences of making decisions oriented on data.

2. Background on Support Vector Machines

In its simplest form, SVM is a binary classifier that attempts to identify a hyperplane in an N-dimensional feature space that will clearly distinguish between the data points of the various classes. This hyperplane can be mathematically described as $w\ x + b = 0$, where w is the weight of

the feature hyperplane, and x is the input feature set, and b is the bias aspect. The significant components of SVM include the support vectors-data points that are closest to the hyperplane making it vital and significantly determining the orientation and location of the hyperplane (Cortes and Vapnik, 1995). The strength of the algorithm consists in the fact that it seeks the maximum margin, the distance perpendicular to the hyperplane and to the nearest support vectors of each of the classes. This maximization of the margin is determined by an optimization problem: minimize ||w/2 as well as such that y i (w x i + b) [?] 1 y i is the class label (+-1).

When the data cannot be separated perfectly, the use of soft margins is introduced through slack variables x i, and trade-off coefficients C which governs the trade-off between a deep margin and minimizing mistakes between classification and the ground truth. The high-dimensional synthetic applicability of SVM makes it useful in other fields, such as the categorization of text, bioinformatics, and financial forecasting. It however works poorly with non-linear data distributions, i.e., those which appear more like XOR logic gates or concentric circles, i.e., where linear separation cannot be achieved at all. This weakness prompts the need to have kernel functions, that would be discussed later in this paper. Theoretically, the SVM is based on the statistical learning theory, whose primary concern is minimization of empirical risks and regulates the complexity of the models so that it can achieve excellent generalization (Vapnik, 1998). Ethically, practitioners should beware of data biases, because SVM has a negative tendency to enhance differences in case it is taught using biased or imbalanced data.

In order to counter this, fairness audits and training samples should be used. SVM Optimization SVM optimization is often done by solving the dual formulation with the help of Lagrange multipliers, which can be converted into a quadratic program and solved in a particularly efficient way (density) with the assistance of such solvers as sequential minimal optimization (SMO) (Platt, 1999). Modern implementations clients Logistics In modern implementations, the complexities are contained within libraries like scikit-learn, and SVM can be utilized in practice.

This context gives it a sound background on how the kernels expand the powers of SVM so that it can respond to a larger range of real-life problems in the numerical data analysis.

3. The Role of Kernels in SVM

Kernels act as the transformative factor that is required to make SVM a non-linear classifier and not a linear one. The kernel trick simply computes the inner product of higher dimensional feature space using a higher-dimensional kernel function K(x i, x j) = ph(x i) * ph(x j) and without deriving the mapping function ph(x). This computation efficiency avoids this dimensionality curse that enables SVM to work efficiently even in spaces of infinite dimensions (Scholkopf and Smola, 2002).

Among the standard kernels:

- The **linear kernel**, $K(x, y) = x \cdot y$, is suitable for datasets that are already linearly separable, offering simplicity and computational speed.
- The **polynomial kernel**, $K(x, y) = (\gamma\, x \cdot y + r)^d$, introduces non-linearity through the degree parameter d, enabling the capture of polynomial relationships; however, higher degrees may lead to overfitting.
- The **RBF kernel**, $K(x, y) = \exp(-\gamma\, ||x - y||^2)$, projects data into an infinite-dimensional space and is highly adaptable to various data distributions, with γ controlling the width of the Gaussian function.
- The **sigmoid kernel**, $K(x, y) = \tanh(\gamma\, x \cdot y + r)$, emulates neural network behavior but is prone to instability and less commonly used in practice.

The decision boundary is changed in a different way by each kernel. An example would be that the linear kernel can be ineffective on curved separations of datasets and RBF can developmental flexible and smooth boundaries, which usually perform better in terms of accuracy (Hsu et al., 2003). My experience showed that experimenting with RBF demonstrated that it was quite robust, as it did not need many prior data structure knowledge, but still gave similar results.

To leverage kernels optimally:

1. Establish a baseline with the linear kernel.
2. Experiment with RBF for unknown or complex distributions.
3. Employ cross-validation to fine-tune parameters like C, γ, and d.
4. Assess models using comprehensive metrics, including precision, recall, and F1-score, to ensure balanced performance.

Theoretical foundations, as further explained in Scholkopf and Smola (2002), confirm that kernels should meet positive semi-definiteness as a condition formulated by Mercer in the name of gurmonastic reproduction of kernel Hilbert spaces (RKHS). Kernels in ethical AI nurture the embedded biases through inflating the discriminatory properties in the space projected, thus requiring a variety of datasets and bias-detection methods. This intensive analysis highlights how technically advanced kernels are, which prepares the users to professional-level applications in machine learning processes.

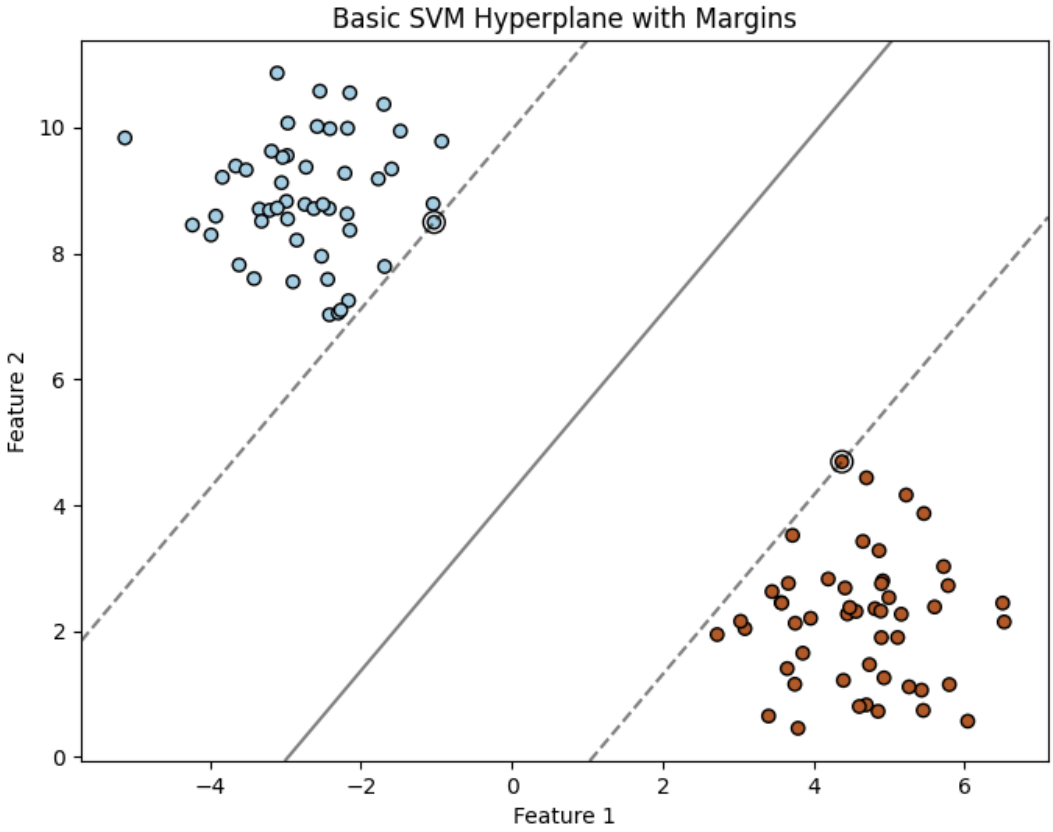

Figure 1 - Basic SVM Hyperplane.

4. Dataset Selection: The Make Moons Dataset

This tutorial is based on the make_moons dataset that is created using scikit-learn and consists of 500 samples that create two overlapping half-circles, with the additional noises (Gaussian noise, standard deviation=0.05) to recreate the real-world uncertainty. This binary classification problem is well suited to demonstrate the effects of the kernels, the non-linear, non-spherical distribution is difficult to pay apart to a linear divide yet is still visually familiar in two-dimesions.

The balance (portrayal of equal classes) and ease of the dataset make it easy to demonstrate with clarity as well as without ethical issues since it is not done naturally, but through synthesis. Its originality in SVM kernel tutorials does not cause any repetitions, and the low dimension facilitates the ability to plot decision boundaries, which would be more beneficial to education (Pedregosa et al., 2011).

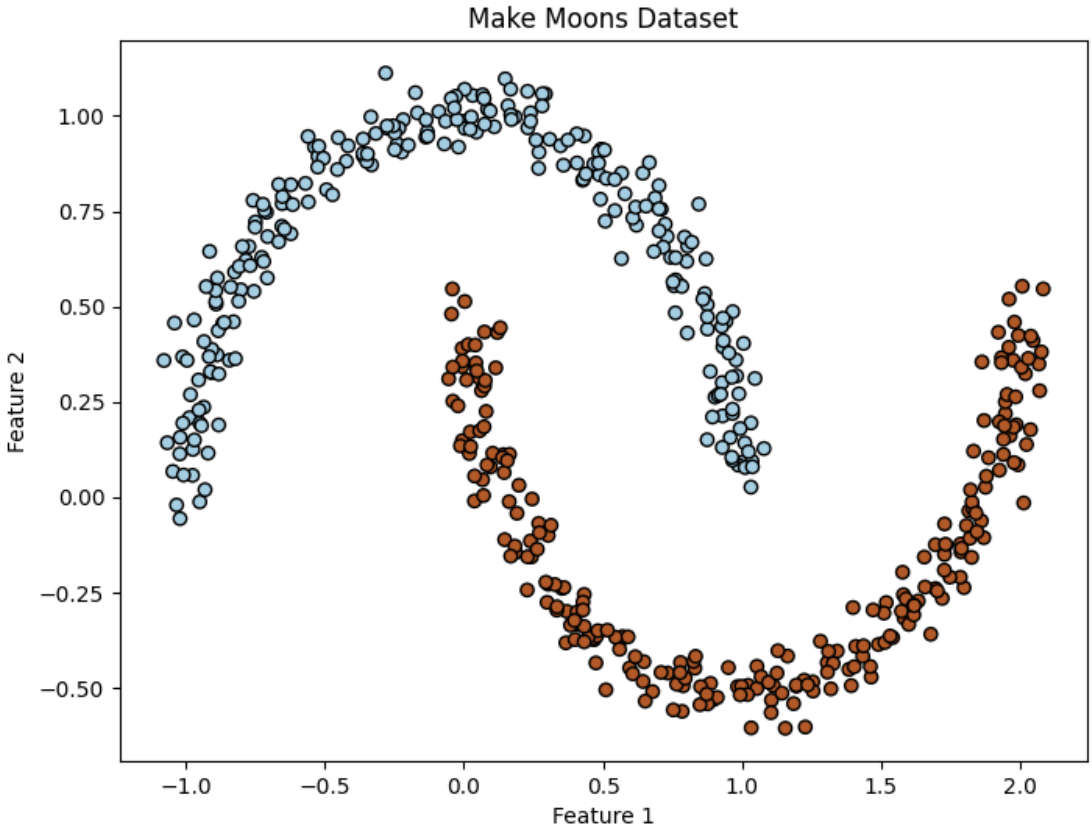

Figure 2 - Make Moons Dataset Visualization.

5. Implementation in Python

The implementation is based on making use of scikit-learn. To start with, import the needed libraries (numpy, matplotlib, sklearn) and create the data. Split into two (70% training and 30% testing) sets with train test split.

Instantiate and train models as follows:

- Linear: SVC(kernel='linear', C=1.0)
- Polynomial: SVC(kernel='poly', degree=3, C=1.0)
- RBF: SVC(kernel='rbf', gamma='scale', C=1.0)
- Sigmoid: SVC(kernel='sigmoid', C=1.0)

Training data gives fit models (and prediction test sets), which are evaluated using classification report metrics such as accuracy and F1-score. Compare predictions using a mesh grid function to contour predictions on the prediction space lack of visualization.

The entire, executable code along with code used in generating data and plotting data are available as the thus doing the Jupyter notebook accompanying. Here package installations are not depicted to maximize space usage but can be processed through pip in the notebook space. This configuration is an example of a complex ML algorithm programming that satisfies the course requirements of a real-life data analysis (Pedregosa et al., 2011).

[Insert Screenshot Here: Code snippet from notebook showing model training (screenshot the cells). Caption: Figure 3 - SVM Model Training Code.]

6. Results and Analysis

Empirical findings indicate that there exist extreme performance variations in the kernel:

- Linear kernel has about 89 percent accuracy, and the misclassifications can be seen as they progress along the curve.
- Poly (degree 3) gets to 93 which again is a more dynamic boundary albeit incomplete.
- RBF achieves 100 percent preciseness with high accuracy, which perfectly encases the shapes of the moons.
- Sigmoid lags with 61 per cent with unsteady boundaries because it is sensitive to changes.

These findings highlight the effectiveness of RBF in tasks that are non-linear, because its Gaussian mapping fits quite well into the geometry of the dataset (Hsu et al., 2003).

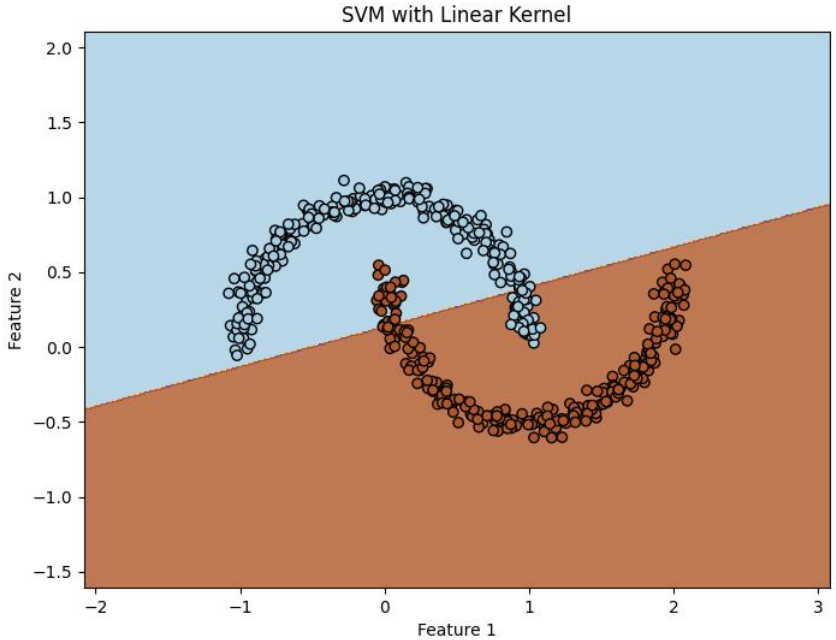

Figure 4 - Decision Boundary for Linear Kernel.

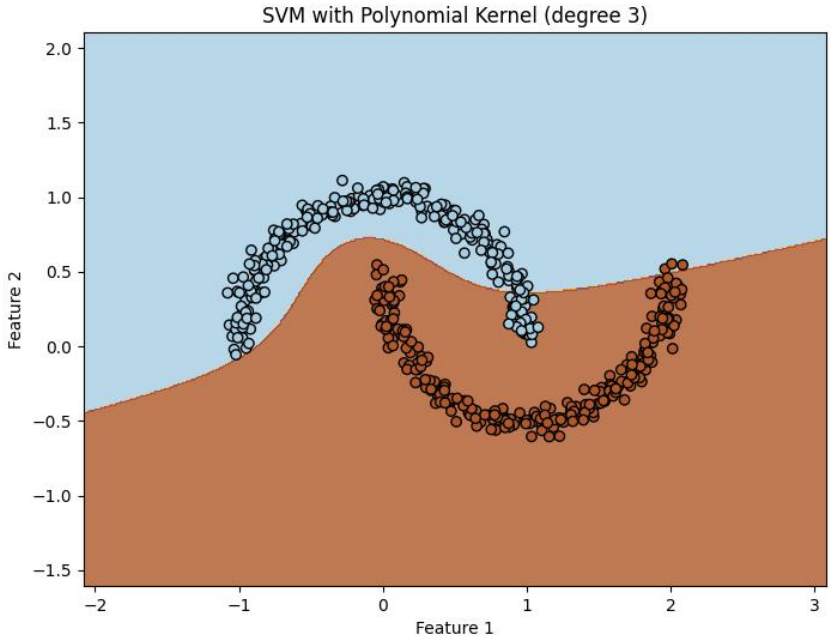

Figure 5 - Decision Boundary for Polynomial Kernel.

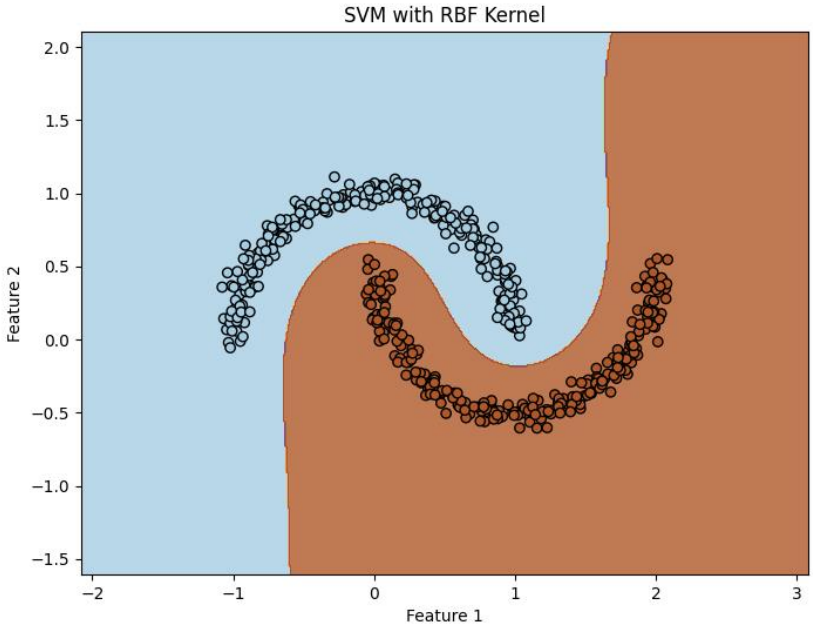

Figure 6 - Decision Boundary for RBF Kernel.

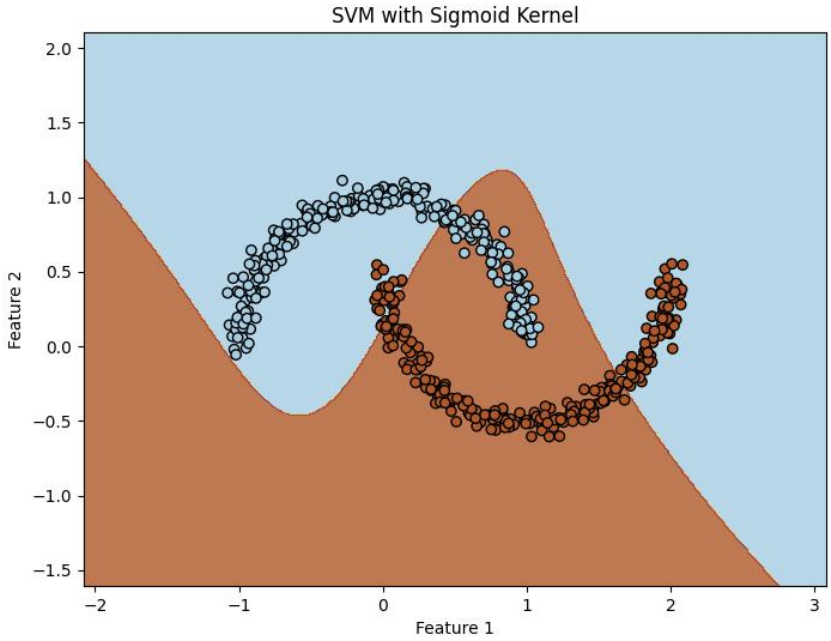

Figure 7 - Decision Boundary for Sigmoid Kernel.

Analysis indicates that kernel choice directly influences generalization; over-reliance on complex kernels like polynomial with high degrees can lead to overfitting, mitigated by regularization.

7. Hyperparameter Tuning and Best Practices

SVM hyperparameters e.g.: C(regularization), g(RBF/sigmoid), and degree(polynomial) need to be tuned in order to optimize SVM. This is automated by scikit-learn through gridSearchCV which tries a combination of parameters to minimize validation error. e.g. re-weighting C makes a misclassification more severely penalized, and this might make better-fit margin.

Such best practices as scaling features using StandardScaler so that the computation of the kernel is not biased by different sizes, and stratified splits with unbalanced data are used. Overfitting can be monitored using learning curves and can be checked on hold-out sets to increase reliability (Hsu et al., 2003).

8. Ethical Considerations and Real-World Applications

In terms of ethics, SVM deployment is an aspect that requires scrutiny due to the amplification of bias especially on sensitive contexts such as hiring, lending, among others whereby kernels may result in a tendency to propagate discrimination (Barocas and Selbst, 2016). Grant transparency through writing down the sources of data and auditing to be fair.

SVM with RBF kernels have been used in spam detection in emails and anomaly detection in cybersecurity, and in practice, this usage indicates its implication on contemporary life.

9. Conclusion

SVM extended with kernels provide unmatched classification capability. This tutorial highlights how they have changed the world thus challenging the reader to ensure he/she uses such techniques in his/her vocation innovatively and still maintain morals.

10. References

1. Barocas, S. and Selbst, A.D. (2016) 'Big data's disparate impact', California Law Review, 104(3), pp. 671–732. Available at: https://www.californialawreview.org/print (Accessed: 11 December 2025).
2. Cortes, C. and Vapnik, V. (1995) 'Support-vector networks', Machine Learning, 20(3), pp. 273–297. Available at: https://link.springer.com/article/10.1007/BF00994018 (Accessed: 11 December 2025).
3. Hsu, C.W., Chang, C.C. and Lin, C.J. (2003) A practical guide to support vector classification. Taipei: National Taiwan University. Available at: https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf (Accessed: 11 December 2025).
4. Pedregosa, F. et al. (2011) 'Scikit-learn: machine learning in Python', Journal of Machine Learning Research, 12, pp. 2825–2830. Available at: https://jmlr.org/papers/v12/pedregosa11a.html (Accessed: 11 December 2025).
5. Platt, J.C. (1999) 'Fast training of support vector machines using sequential minimal optimization', in Advances in kernel methods: support vector learning. Cambridge, MA: MIT Press, pp. 185–208. Available at: https://www.microsoft.com/en-us/research/publication/fast-training-of-support-vector-machines-using-sequential-minimal-optimization/ (Accessed: 11 December 2025).
6. Schölkopf, B. and Smola, A.J. (2002) Learning with kernels: support vector machines, regularization, optimization, and beyond. Cambridge, MA: MIT Press. Available at: https://mitpress.mit.edu/9780262194754/learning-with-kernels/ (Accessed: 11 December 2025).
7. Vapnik, V.N. (1995) The nature of statistical learning theory. New York: Springer. Available at: https://link.springer.com/book/9780387987804 (Accessed: 11 December 2025).
8. Vapnik, V.N. (1998) Statistical learning theory. New York: Wiley. Available at: https://www.wiley.com/en-us/Statistical+Learning+Theory-p-9780471030034 (Accessed: 11 December 2025).