Student Name: ShivaShankar Bathuka
Student ID: 24071919
GitHub: https://github.com/bathukashivashankar/svm-kernels-make-moons-tutorial

Visualising SVM Kernels on the make_moons Dataset

Introduction

Support Vector Machines (SVMs) are powerful supervised learning models that construct decision boundaries with maximum margin between classes. They are widely used for classification problems where robustness and good generalisation are important. A key design choice in SVMs is the **kernel function**, which controls the shape of the decision boundary the model can learn.

This tutorial investigates how three common kernels—**linear**, **radial basis function (RBF)** and **polynomial**—behave on the synthetic make_moons dataset. The aim is to give an intuitive understanding of the **kernel trick** and to show, through code and visualisations, why non-linear kernels are often necessary when the classes cannot be separated by a straight line. The focus is on clear explanation and visual insight rather than on mathematical derivations.

Dataset and Experimental Setup

The make_moons function from scikit-learn generates two interlocking crescent-shaped clusters in two dimensions. In this tutorial, 1000 samples are created with moderate Gaussian noise (noise=0.2), giving a realistically messy but clearly non-linear pattern. Each point is described by two numeric features and a binary label indicating its moon.

The dataset is split into **training (80%)** and **test (20%)** sets using a stratified split to preserve the 50/50 class balance in both subsets. All models in this tutorial are trained using the same train–test split to ensure a fair comparison. No feature scaling is strictly necessary for make_moons, but using SVMs with RBF and polynomial kernels generally benefits from normalisation on real datasets.

To support accessibility, all plots use a **colorblind-friendly palette** (sns.set_palette("colorblind")) and enlarged axis labels and titles. In the final PDF version, each figure can be accompanied by alt-text that verbally describes the shapes and boundaries in the plot so that screen-reader users can follow the tutorial.

Intuitive View of SVMs and the Kernel Trick

An SVM classifier searches for the decision boundary that maximises the margin between the boundary and the closest data points, called **support vectors**. In two dimensions this boundary is a line; in higher dimensions, it is a hyperplane. Maximising the margin tends to improve generalisation because the model is not forced to pass arbitrarily close to any training point.

However, a **linear SVM** can only draw straight boundaries in the original feature space. When the classes are curved or intertwined, no straight line can separate them well. The **kernel trick** addresses this by implicitly mapping the data into a higher-dimensional space where a straight boundary *is* possible, while computing everything in terms of kernel functions that measure similarity between points. Different kernels correspond to different ways of bending the boundary when projected back to the original two-dimensional space.

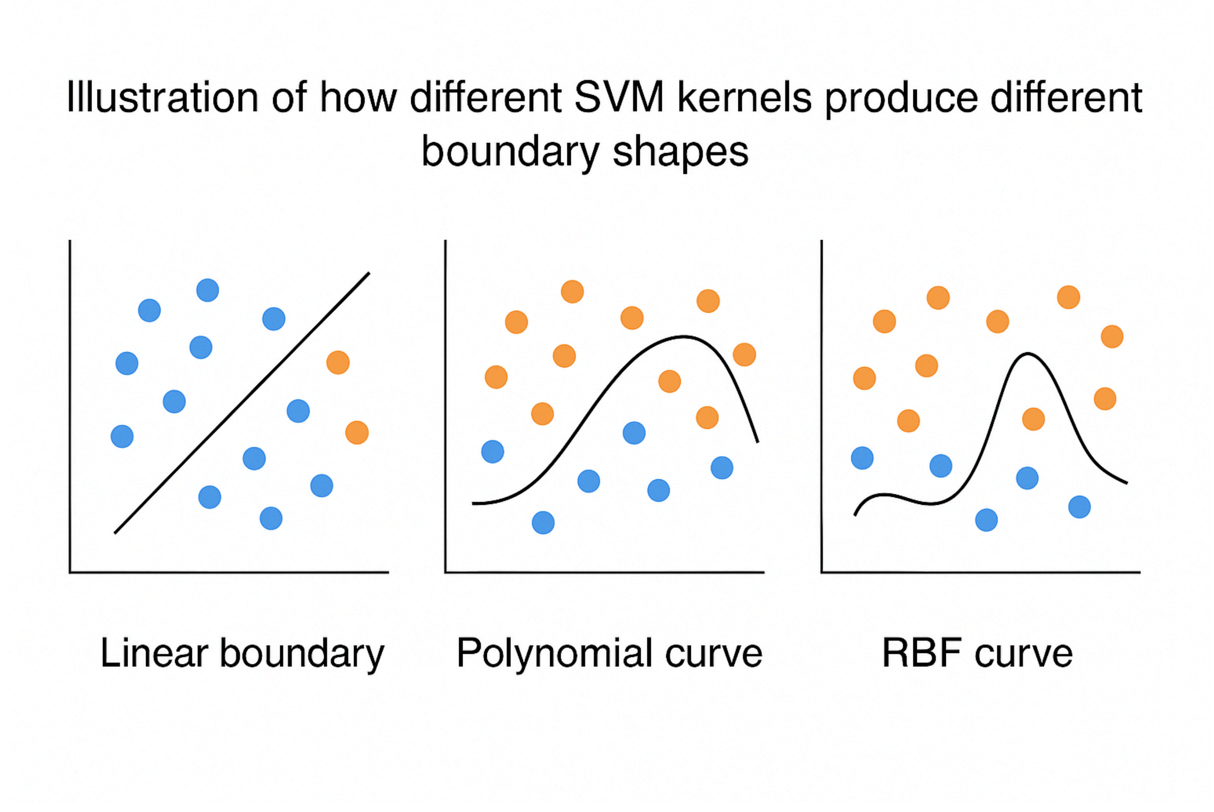


Illustration of how different SVM kernels produce different boundary shapes
Linear boundary
Polynomial curve
RBF curve

Figure 1: Conceptual illustration of how different SVM kernels produce different boundary shapes: a straight linear boundary, a smooth polynomial curve, and a more flexible RBF curve.

Why make_moons is a Good Kernel Example

The make_moons dataset is deliberately chosen because it is **not linearly separable**. The two moons curve around each other, and any straight line that separates one part of a moon will inevitably cut through the other. This makes it ideal for demonstrating the limitations of linear models and the benefits of non-linear kernels.

Another advantage is that the data is **two-dimensional**, so decision boundaries can be visualised directly. For each SVM model, the notebook creates a fine grid over the feature space, uses the trained classifier to predict the class at each grid point, and then plots the resulting decision regions along with the original data points. This style of plot is common in SVM tutorials because it clearly shows how boundary shape changes with the choice of kernel.

Linear Kernel: A Straight Boundary on Curved Data

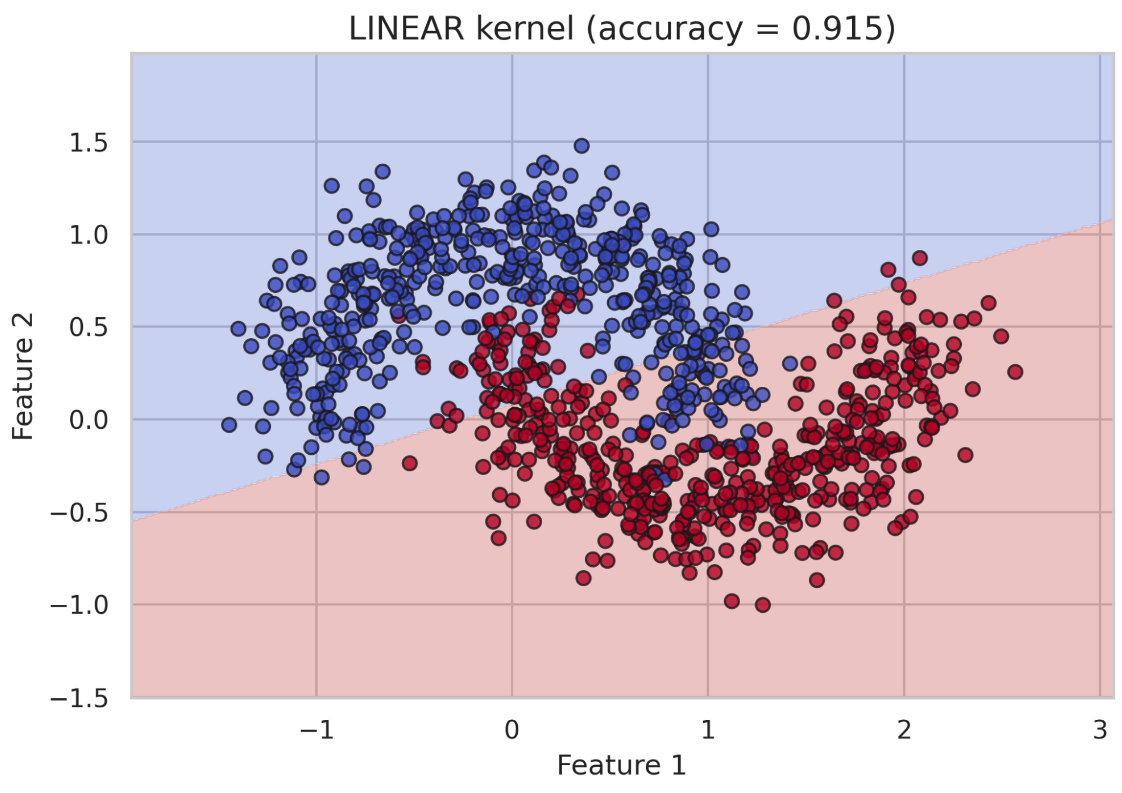


Figure 2 – Linear kernel
Linear SVM decision boundary on the make_moons dataset, showing a straight separating line that cuts through both crescent-shaped clusters.

The first experiment trains an SVM with a **linear kernel** and regularisation parameter C=1.0. On the test set, this model achieves an accuracy of approximately **0.915**. Many points are classified correctly, but the confusion matrix and classification report show that both classes still suffer from misclassifications, especially around the overlapping central region.

In the linear decision-boundary plot, the feature space is split into two large half-planes by an almost straight line. Points from the upper moon occupy most of the blue region, and points from the lower moon occupy most of the red region, but several red points lie above the line and several blue points lie below it. This visualises the core issue: the linear kernel cannot bend enough to follow the curvature of the moons, so it has to compromise with a boundary that cuts through both shapes.

This result is important for students: it shows that even a high-margin linear SVM is not magic. When the underlying geometry of the data is fundamentally non-linear, linear separation is simply the wrong hypothesis class.

RBF Kernel: Flexible Local Boundaries

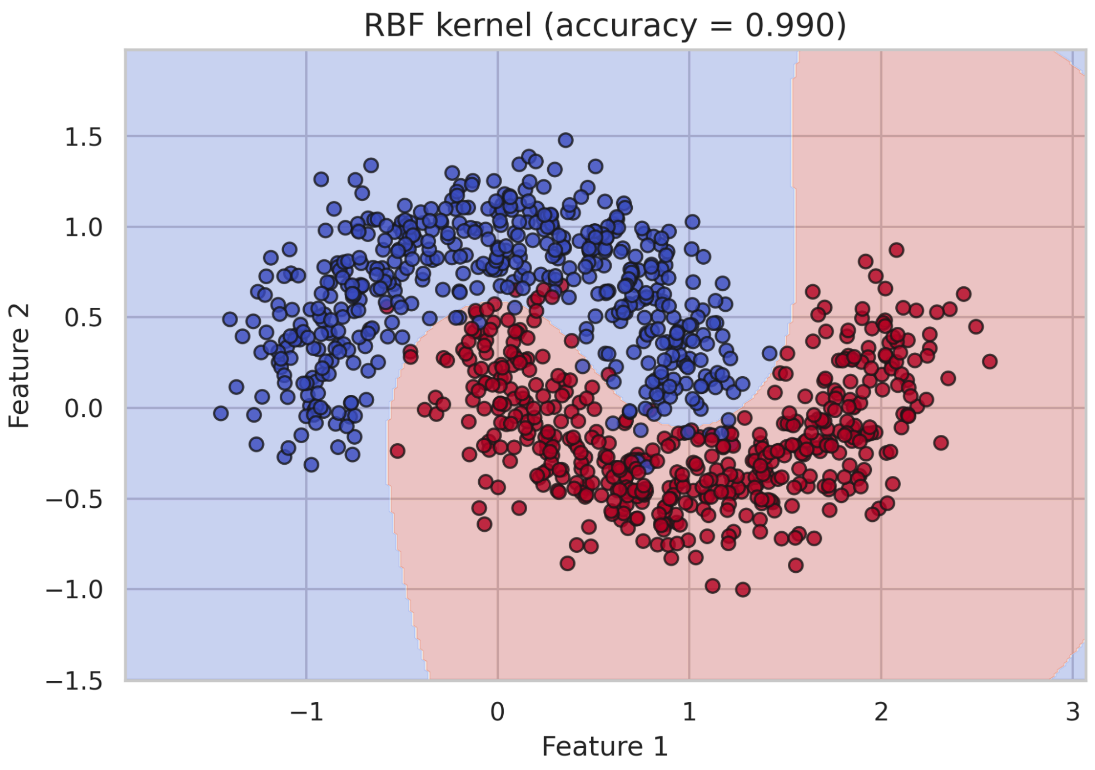


Figure 3 – RBF kernel
RBF-kernel SVM decision boundary, with curved regions that closely wrap around each moon and achieve much higher classification accuracy.

The second experiment uses the **RBF kernel** (also known as the Gaussian kernel) with C=1.0 and gamma=1.0. The RBF kernel measures similarity based on the distance between points; nearby points in the original space have high similarity, and distant points have low similarity. This leads to decision boundaries that can bend and adapt locally.

On the same train–test split, the RBF SVM reaches a test accuracy of about **0.99**, a substantial improvement over the linear model. In the corresponding decision-boundary plot, the red and blue regions wrap tightly around each moon, leaving only a small number of misclassified points in the overlapping noisy area. The boundary curves smoothly between the moons and around their tips, closely matching the true structure.

This behaviour illustrates how the kernel trick enables **non‑linear separation** without explicitly constructing high‑dimensional features. The model is still a “linear” SVM in the kernel‑induced feature space, but when projected back to two dimensions the boundary is highly curved. The RBF kernel often serves as a strong default choice for non‑linear datasets because of its flexibility.

Polynomial Kernel: Smooth Global Curvature

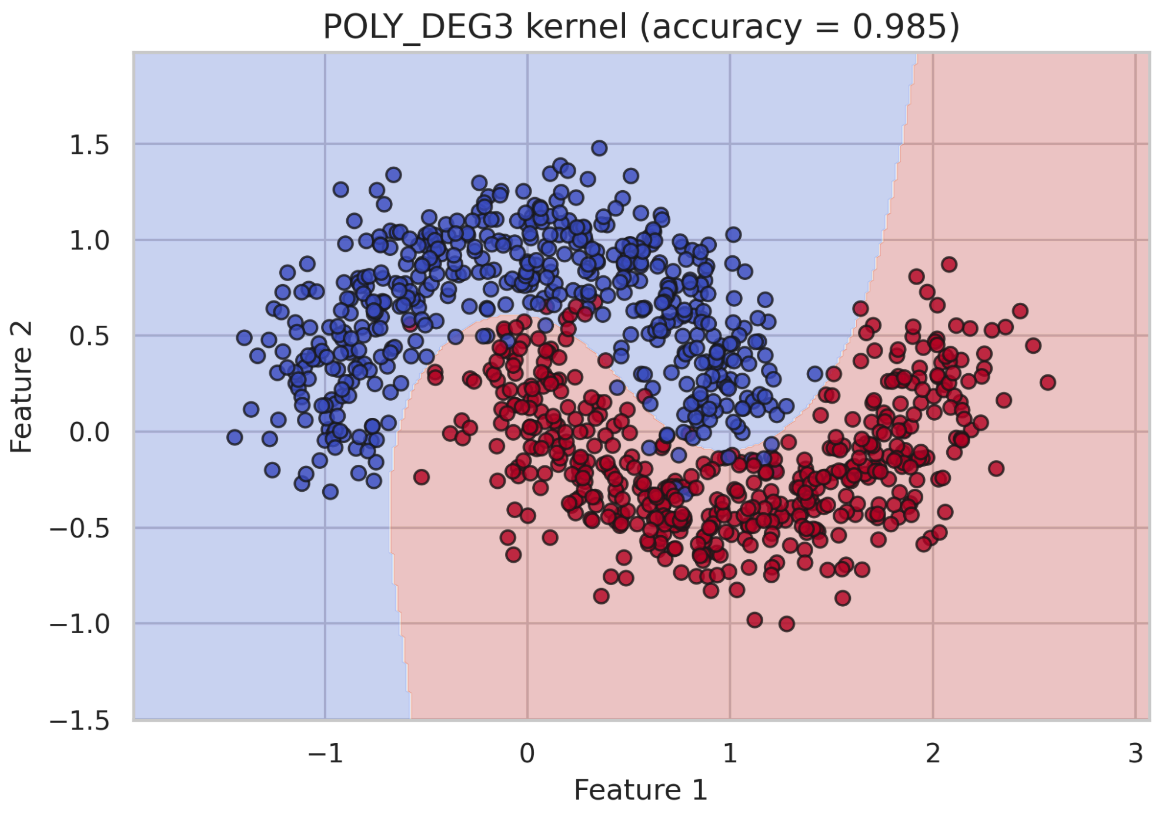


Figure 4 – Polynomial kernel

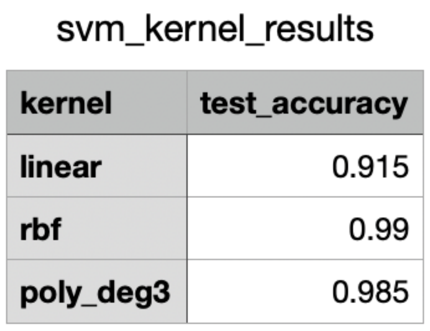
Degree‑3 polynomial‑kernel SVM decision boundary, producing a smooth curved separator that follows the moons well but slightly less tightly than the RBF boundary.

The third experiment considers a **polynomial kernel** of degree 3 with C=1.0, gamma=1.0 and coef0=1.0. Polynomial kernels allow the decision function to depend on polynomial combinations of the original features, giving non‑linear boundaries that are typically smoother and more global than those produced by RBF.

The polynomial SVM attains a test accuracy of roughly **0.985**, slightly below the RBF model but still clearly above the linear baseline. The decision‑boundary plot shows a curved boundary that successfully separates most of the upper and lower moons. Compared to RBF, the boundary appears a bit less tightly wrapped around every local bump, reflecting the more global nature of low‑degree polynomials.

This model demonstrates that there is not a single “best” kernel: both RBF and polynomial kernels are capable of non-linear separation, but they encode different assumptions about how decision boundaries should behave. If the true boundary is smooth and can be approximated by a polynomial of low degree, the polynomial kernel can perform very well.

Quantitative Comparison of Kernels



svm_kernel_results

kernel	test_accuracy
linear	0.915
rbf	0.99
poly_deg3	0.985

Table 1

Test accuracy of SVM classifiers with linear, RBF, and degree-3 polynomial kernels on the make_moons dataset.

To compare the kernels side by side, the notebook summarises the results in a small table:

- Linear kernel: test accuracy ≈ **0.915**
- RBF kernel: test accuracy ≈ **0.990**
- Polynomial (degree 3): test accuracy ≈ **0.985**

All three models are trained with the same C value and evaluated on the same test set. The difference in performance therefore comes from the ability of each kernel to fit the non-linear structure of make_moons. Linear SVMs underfit the data, while RBF and polynomial kernels provide enough flexibility to capture the curved boundary.

On larger or noisier datasets, these numbers would depend strongly on hyperparameter choices such as C, gamma, and degree. In practice, these parameters should be chosen using cross-validation rather than fixed manually. The tutorial keeps them simple to focus on the visual intuition rather than model selection.

Discussion: When to Use Which Kernel?

The experiments lead to several practical guidelines for using SVM kernels:

- **Linear kernel**
  Best suited for data that is approximately linearly separable or for very high-dimensional feature spaces where a linear boundary is often sufficient. It is computationally cheaper than non-linear kernels and easier to interpret, but will underfit strongly non-linear patterns like make_moons.

- **RBF kernel**
  A strong default for many non-linear problems. It can model highly flexible, local decision boundaries and often achieves excellent accuracy when hyperparameters are tuned. The downside is that it may overfit if gamma is too large, and the resulting boundary is harder to interpret geometrically.

- **Polynomial kernel**
  Useful when interactions between features can reasonably be captured by low-degree polynomials, giving smooth global curvature. It can be a good compromise between the simplicity of the linear kernel and the flexibility of RBF, but performance is sensitive to degree, gamma, and coef0.

Conclusion

This tutorial showed how changing the SVM kernel dramatically alters the decision boundary learned on the make_moons dataset. The linear kernel produced a straight separator that could not fully follow the curved moons, leading to noticeably lower accuracy. In contrast, both RBF and degree-3 polynomial kernels created non-linear boundaries that wrapped around the moons and achieved much higher test accuracy, illustrating the practical power of the kernel trick.

More broadly, the experiments highlight that kernel choice encodes assumptions about boundary shape: linear for roughly straight separations, RBF for highly flexible local patterns, and polynomial for smooth global curvature. On real problems, a good workflow is to start with a linear SVM as a baseline and then move to RBF or polynomial kernels when the data is clearly non-linear, tuning hyperparameters such as C, γ, and degree with cross-validation. The make_moons example provides a compact visual demonstration of why such non-linear kernels are often essential. Understanding these trade-offs helps practitioners select SVM kernels that balance model flexibility, generalisation, and interpretability.

Ethical and Real-World Considerations

Although this tutorial uses a synthetic dataset, similar SVM models are often applied to high-stakes domains such as credit scoring or medical diagnosis. In these settings, kernel choice and model complexity affect not only accuracy but also fairness and interpretability: highly flexible kernels like RBF can overfit historical biases or spurious correlations, while their complex non-linear boundaries make it harder to explain individual decisions to affected users or regulators. Simpler kernels (such as linear or low-degree polynomial) may be easier to audit and monitor, even if they sacrifice a small amount of performance, and model validation should therefore include checks for overfitting and disparate impact across demographic groups, not just overall accuracy.

References

- Scikit-learn developers. "1.4. Support Vector Machines." *scikit-learn User Guide*. https://scikit-learn.org/stable/modules/svm.htmlscikit-learn

- Scikit-learn developers. "make_moons — scikit-learn 1.7.2 documentation." https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.htmlscikit-learn

- GeeksforGeeks. "Kernel Trick in Support Vector Classification." 2024. https://www.geeksforgeeks.org/machine-learning/kernel-trick-in-support-vector-classification/geeksforgeeks

- GeeksforGeeks. "How to Choose the Best Kernel Function for SVMs." 2024. https://www.geeksforgeeks.org/machine-learning/how-to-choose-the-best-kernel-function-for-svms/geeksforgeeks

- freeCodeCamp. "SVM Kernels Explained: How to Tackle Nonlinear Data in Machine Learning." 2025. https://www.freecodecamp.org/news/svm-kernels-how-to-tackle-nonlinear-data-in-machine-learning/freecodecamp