# ML_LAB_SVM

DATE : 9/10/2025 SEM : 5

SECTION : F SRN : PES2UG23CS366

**Analysis Questions for Moons:**

1. Based on the metrics and the visualizations, what inferences about the performance of the Linear Kernel can you draw?

**Answer :-**

Linear kernel — inferences from metrics & visualization

- **Behavior:** Linear SVM produces a straight-line decision boundary.
- **Observed effect:** It **underfits** the Moons data (which is intrinsically non-linear).
- **Evidence:** Lower accuracy (~0.84) and lower F1-scores for both classes; many points along the curved moon arcs are misclassified.
- **Conclusion:** Linear kernel is too simple (high bias) for this dataset — it cannot capture the curved class boundary.

**2. Compare the decision boundaries of the RBF and Polynomial kernels. Which one seems to capture the shape of the data more naturally**?

- **RBF:** Produces *smooth*, *radial* decision regions that closely follow the curved moon shapes; tends to give the best accuracy (in your run ≈ 0.94–0.95).
- **Polynomial (degree 3):** Can fit curved boundaries but may introduce extra wiggles or irregularities (depending on degree and coef0); performance is usually slightly worse than RBF for this dataset.

- **Verdict: RBF** captures the moons more naturally — smoother boundary and better empirical performance.

## Analysis Questions for Banknote:

1. **In this case, which kernel appears to be the most effective?**

- **Typical result:** The **linear kernel** often performs very well on the Banknote Authentication dataset because its features (variance, skewness, curtosis, entropy) are often linearly separable after scaling.

- **If your experiments show otherwise:** If RBF slightly outperforms linear, it means a small nonlinearity exists; otherwise choose **linear** for simplicity and interpretability.

2. **The Polynomial kernel shows lower performance here compared to the Moons dataset. What might be the reason for this?**

- **Reasons:**
    - The Banknote dataset is **closer to linearly separable**; an unnecessarily complex polynomial boundary can *overfit* noise instead of improving class separation.
    - Polynomial kernels introduce **higher model complexity** (extra curvature / interactions) which is unnecessary when the classes are already separable in the original feature space.
    - Sensitive to **feature scaling** and hyperparameters (degree, coef0) — if not tuned, polynomial can underperform.

□ **Conclusion:** Polynomial underperforms because it adds complexity that isn't needed and can fit noise or cause unstable boundaries.

### General Margin / C Analysis (Soft vs Hard)

### 1) Which model produces a wider margin: C=0.1 (soft) or C=100 (hard)?

- **Wider margin: Soft margin (C = 0.1)** produces a wider margin.

    - Low **C** emphasizes maximizing margin over classifying every training point correctly → larger margin, more slack allowed.

### 2) Why does SVM allow points inside/on the wrong side of the margin in Soft Margin?

- **Reason:** Soft-margin SVM introduces **slack variables** ($\xi_i$) and a penalty controlled by **C**.
- **Trade-off:** The model trades some training errors (points inside or on the wrong side) for a **larger margin** that usually generalizes better.
- **Primary goal: Maximize the margin** while controlling misclassification via the penalty term — i.e., **good generalization**, not perfect training accuracy.

### 3) Which model is more likely to overfit?

- **More likely to overfit: Hard margin / high C (C = 100)**.
    - High C strongly penalizes misclassification, forcing the classifier to fit training points tightly; this can fit noise and reduce generalization.

## 4) Which model to trust on a new unseen point? Which C to prefer when data is noisy?

- **Trust more: Soft-margin model (low C)** — more robust to noise and less likely to be misled by outliers.
- **Real-world recommendation:** Start with a **lower C** (e.g., 0.01–1 depending on scale) when data is noisy, then tune C with cross-validation. Low C favors simpler decision boundaries and typically generalizes better; increase C only if validation shows underfitting.

### 4 . Imagine you receive a new, unseen data point. Which model do you trust more to classify it correctly? Why? In a real-world scenario where data is often noisy, which value of C (low or high) would you generally prefer to start with?

I would trust the **Soft Margin SVM (with low C)** more to classify a new, unseen data point correctly.

**Reasoning:**

- A **low C value** means the model allows some misclassifications on the training data in exchange for a **wider margin**.

- This wider margin makes the model **less sensitive to noise and outliers**, resulting in **better generalization** to unseen data.
- In contrast, a **high C (hard margin)** model tries to perfectly classify all training samples, which can lead to **overfitting** — it performs well on the training set but poorly on new data.

**In a real-world scenario:**

- Data is almost always **noisy or imperfect**.
- Therefore, it is better to **start with a lower C value** (e.g., 0.1 or 1) and later tune it using cross-validation if needed.
- A smaller C produces a **simpler, smoother decision boundary** that generalizes better.

---