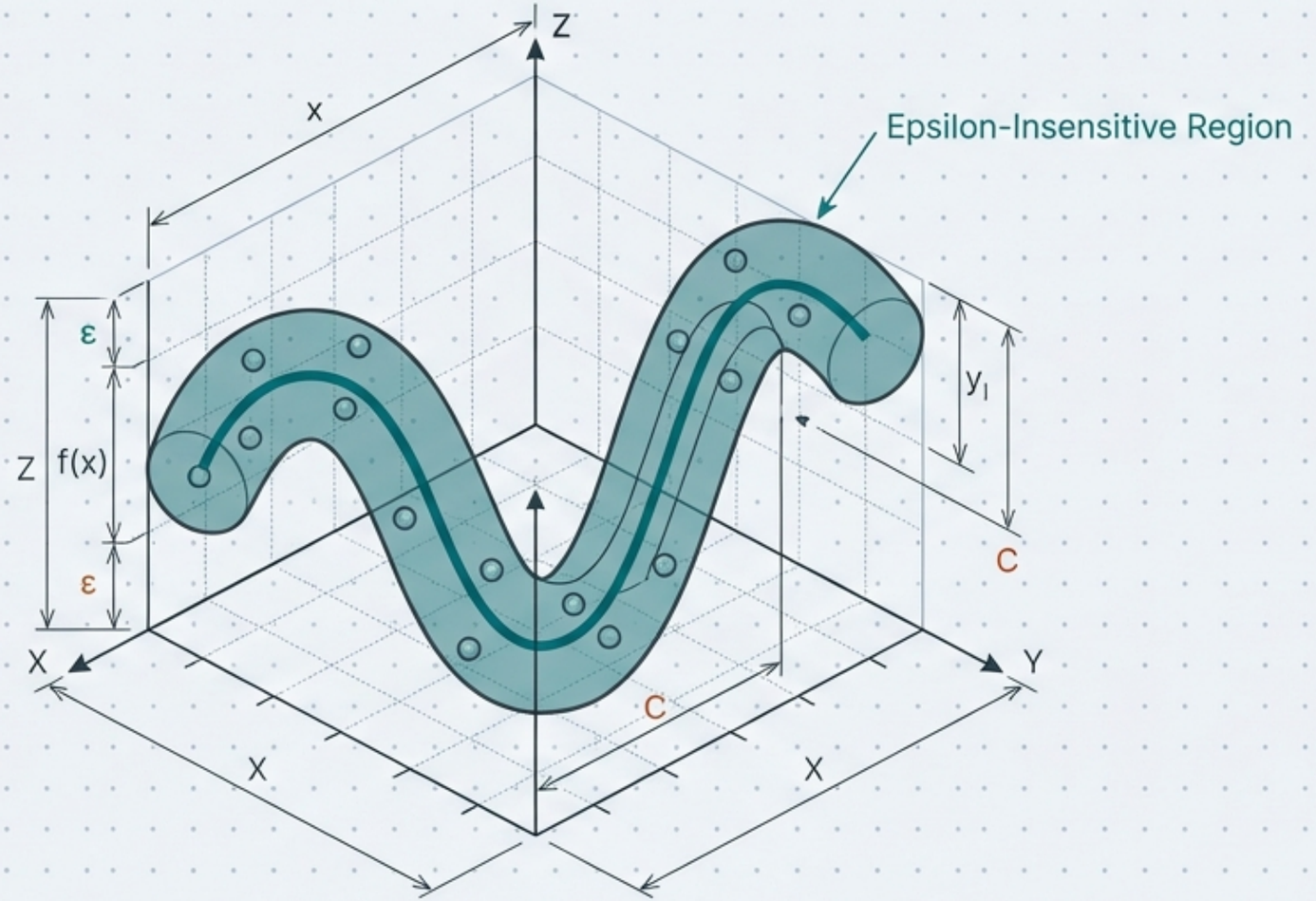
Mastering Support Vector Regression (SVR)

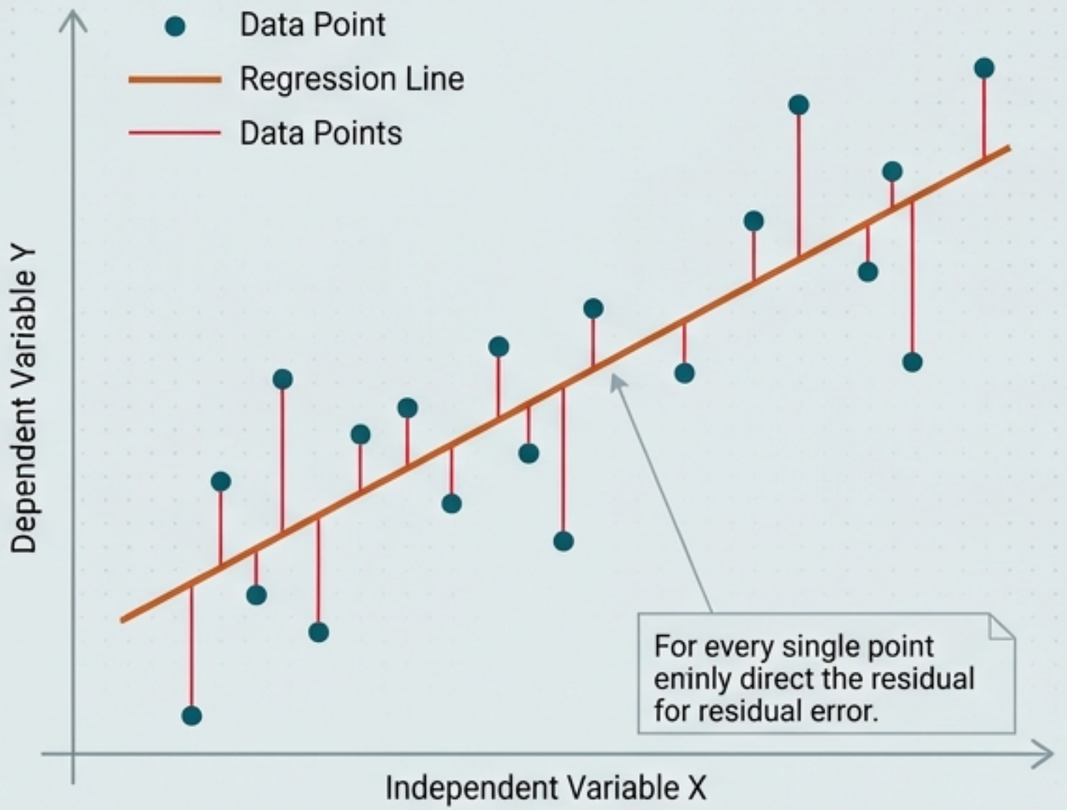
A Guide to Predictive Modeling with Tolerance



Concept-First, Code-Second. From Intuition to sklearn.svm

The Paradigm Shift: From Bullseye to Dartboard

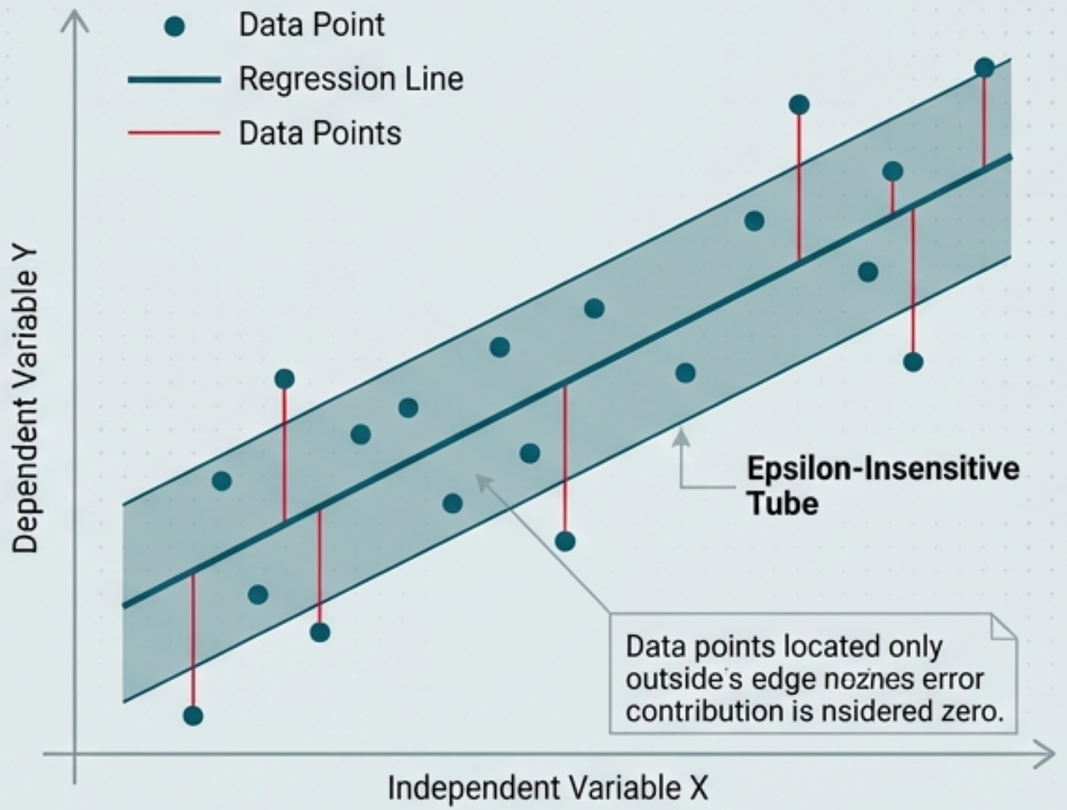
OLS (Ordinary Least Squares)



Minimizing Sum of Squared Errors
OLS tries to hit the bullseye.
Every error counts, making it sensitive to outliers.

Empirical Risk Minimization

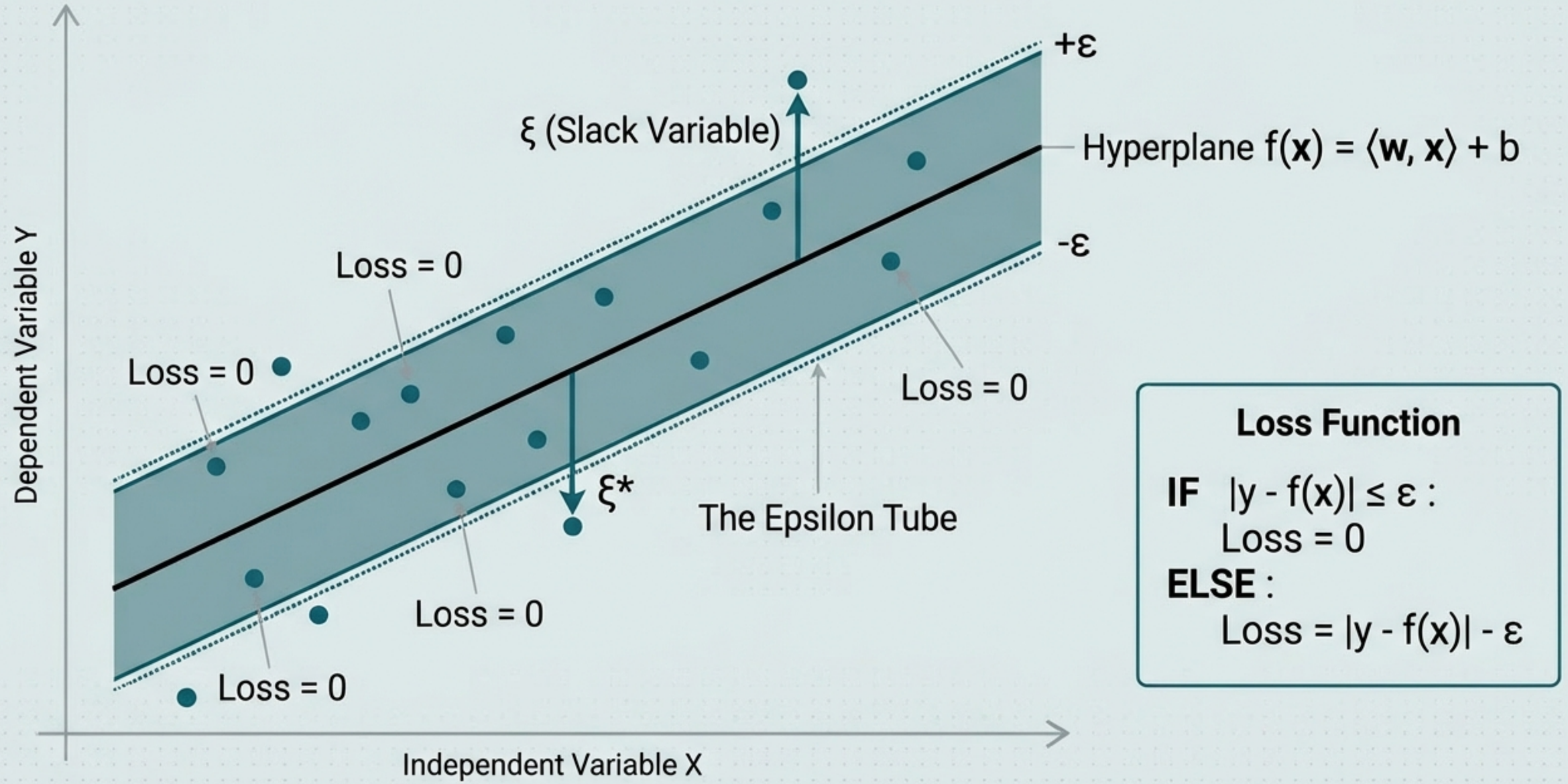
SVR (Support Vector Regression)



Epsilon-Insensitive Loss
SVR tries to hit the dartboard.
"Close enough" is good enough.
Errors inside the tube are ignored.

Structural Risk Minimization

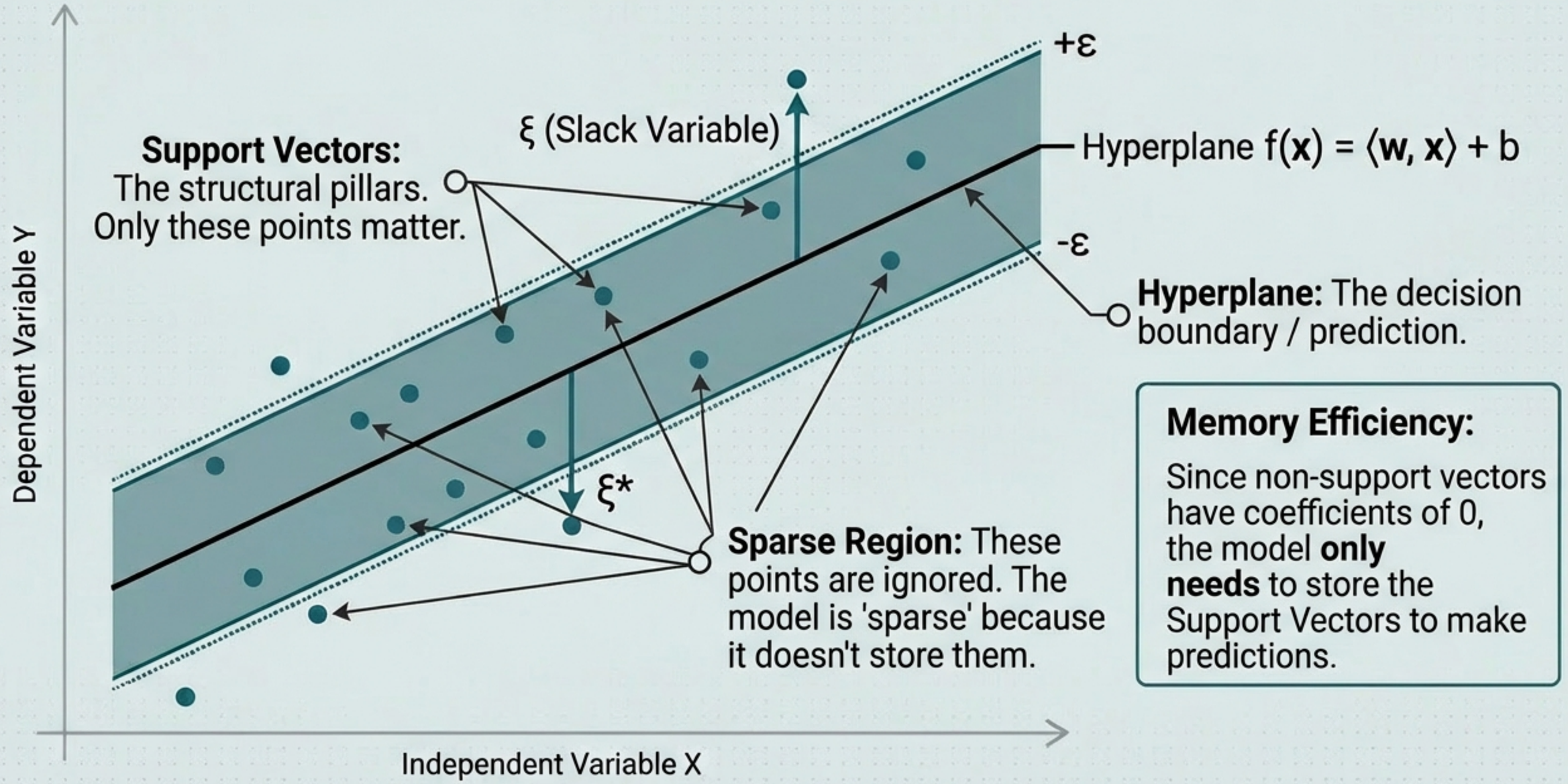
The Geometry of Tolerance: The ε-Insensitive Tube



Loss Function

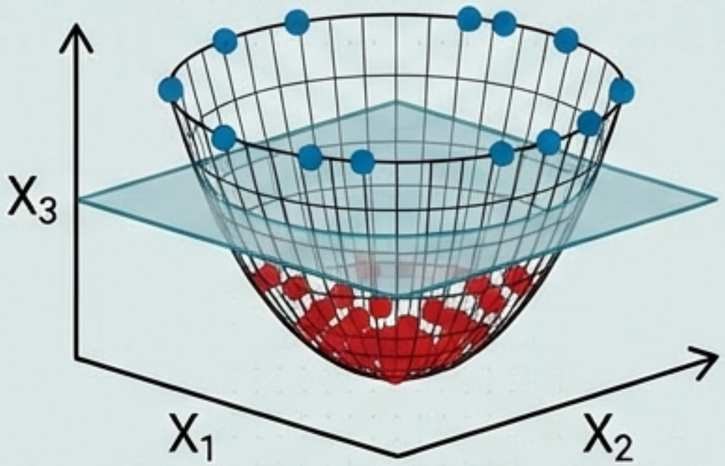
IF $|y - f(x)| \leq \varepsilon$:
Loss = 0
ELSE :
Loss = $|y - f(x)| - \varepsilon$

The Cast of Characters



The Kernel Trick: Fitting Curves with Straight Sheets

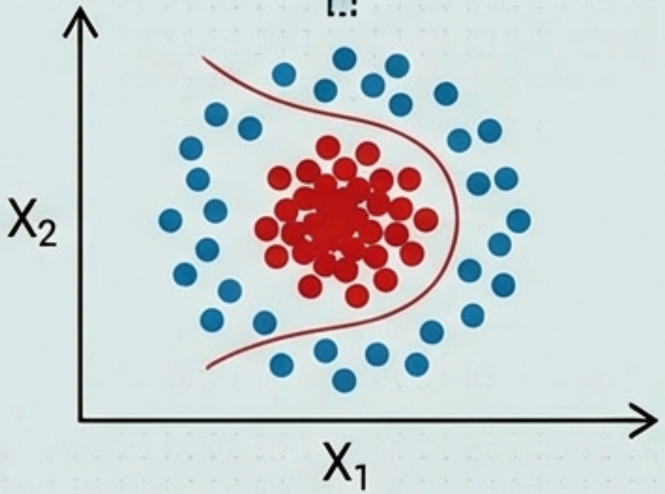
Step 2:



Feature Space (3D)

Mapping Function $\phi(x)$

Step 1:



Input Space (2D)

Linearly Separable.

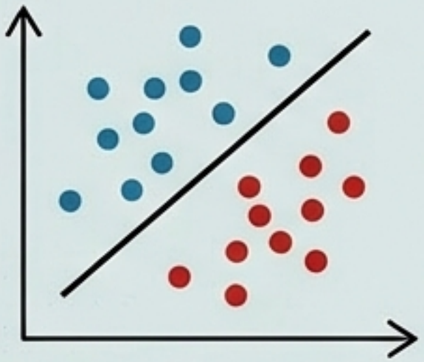
The kernel trick computes dot products in this space without without explicitly calculating coordinates.

Linearly Inseparable.
No straight line can separate red from blue.

The Kernel Menu: Choosing Your Geometry

Linear Kernel

`kernel='linear'`

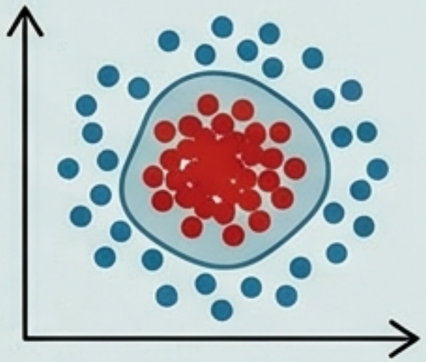


Math: $\langle \mathbf{x}, \mathbf{x}' \rangle$.

Best for: High-dimensional data (Text), massive datasets. Fast.

RBF Kernel (Radial Basis Function)

`kernel='rbf'`

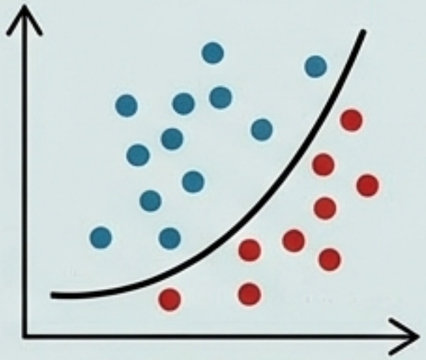


Math: $\exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$

Best for: The Default choice. Maps to infinite dimensions. Captures local behavior.

Polynomial Kernel

`kernel='poly'`

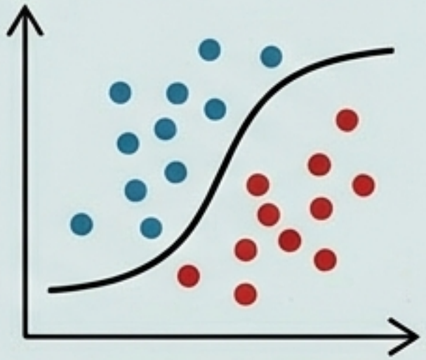


Math: $(\gamma\langle \mathbf{x}, \mathbf{x}' \rangle + r)^d$

Best for: Image processing. Captures feature interactions.

Sigmoid Kernel

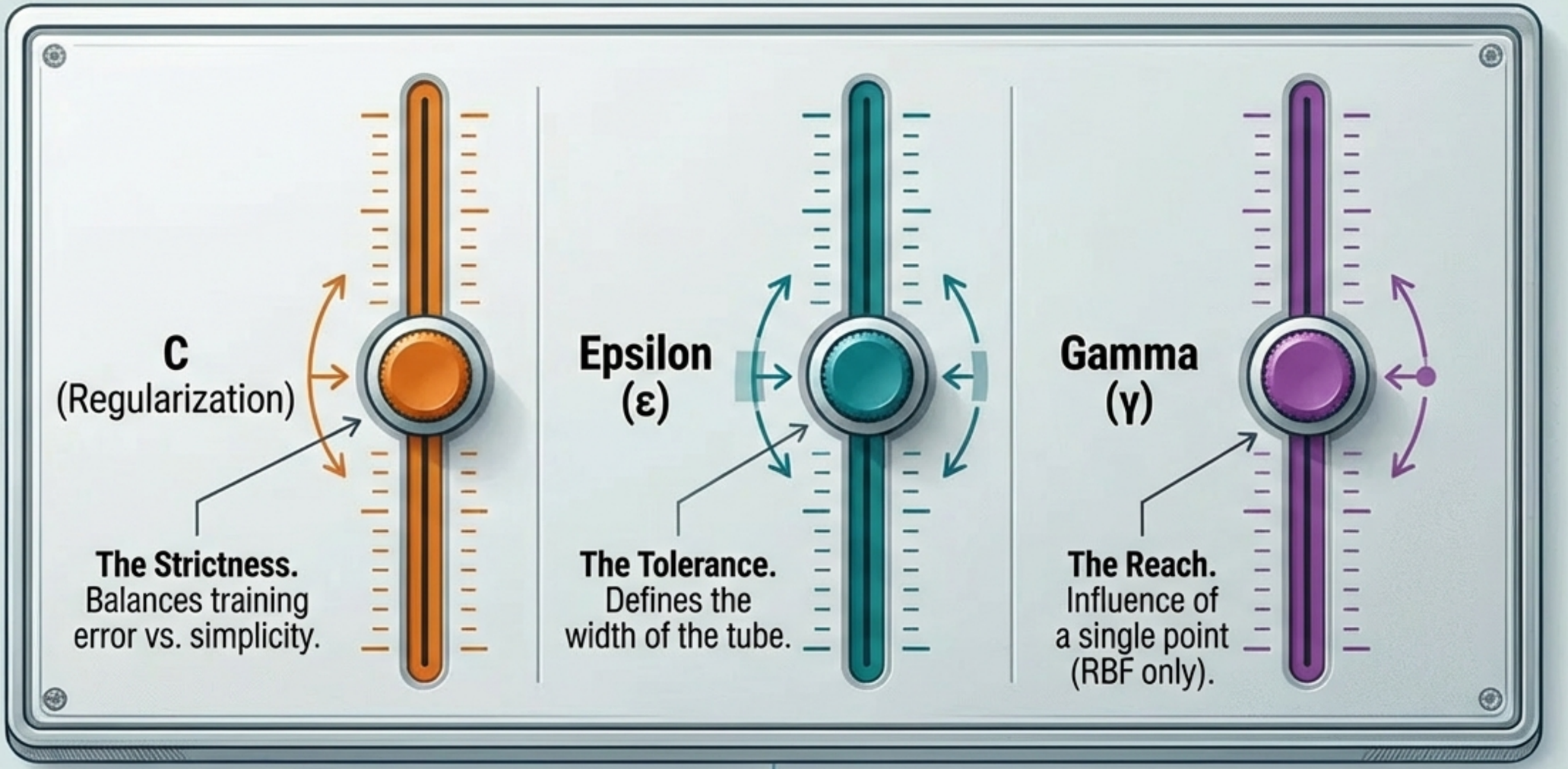
`kernel='sigmoid'`



Math: $\tanh(\gamma\langle \mathbf{x}, \mathbf{x}' \rangle + r)$

Best for: Neural Network proxy. Often unstable for general regression.

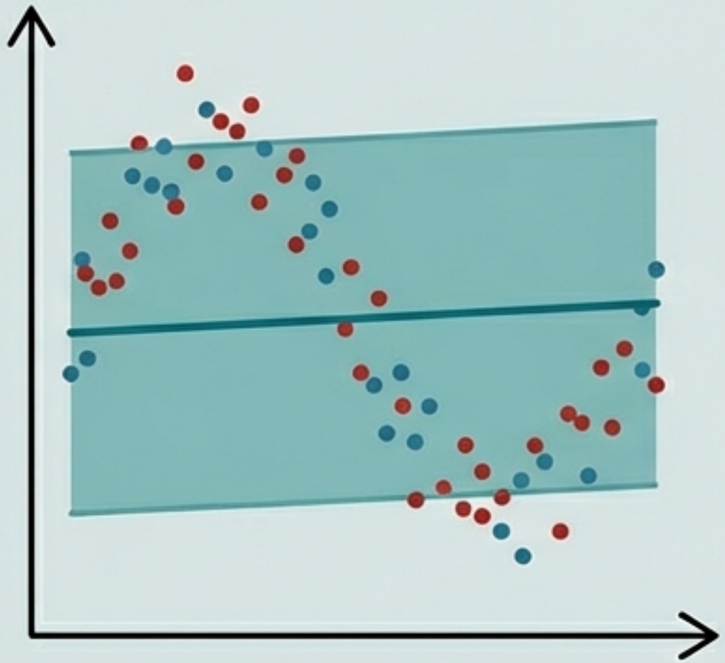
The Hyperparameter Ecosystem



Ecology: These parameters are interconnected. Changing one often requires tuning the others.

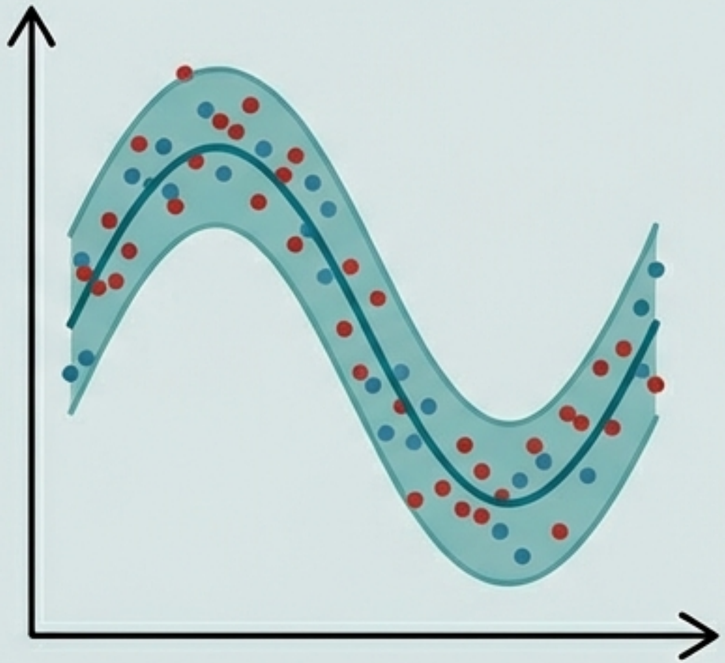
Tuning Epsilon (ε): Setting the Width of the Street

Large ε



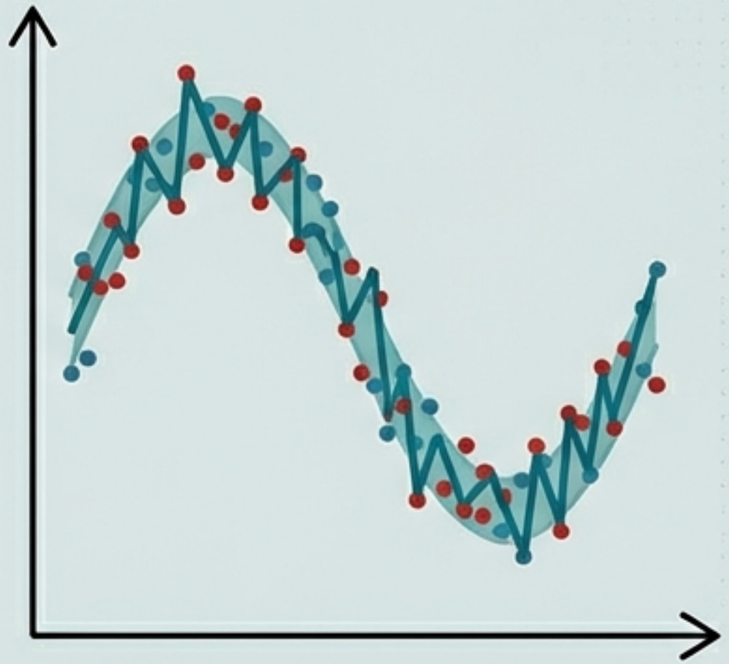
Underfitting. High Bias. Too tolerant.

Optimal ε



Balanced. Captures signal, ignores noise.

Small ε

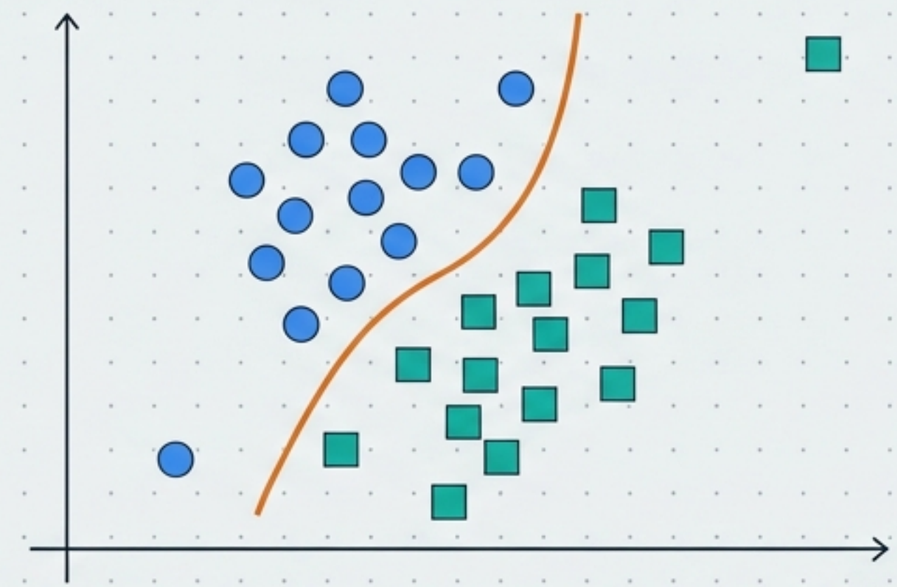


Overfitting. High Variance. Captures noise.

Wider Tube = Fewer Support Vectors = Simpler Model.

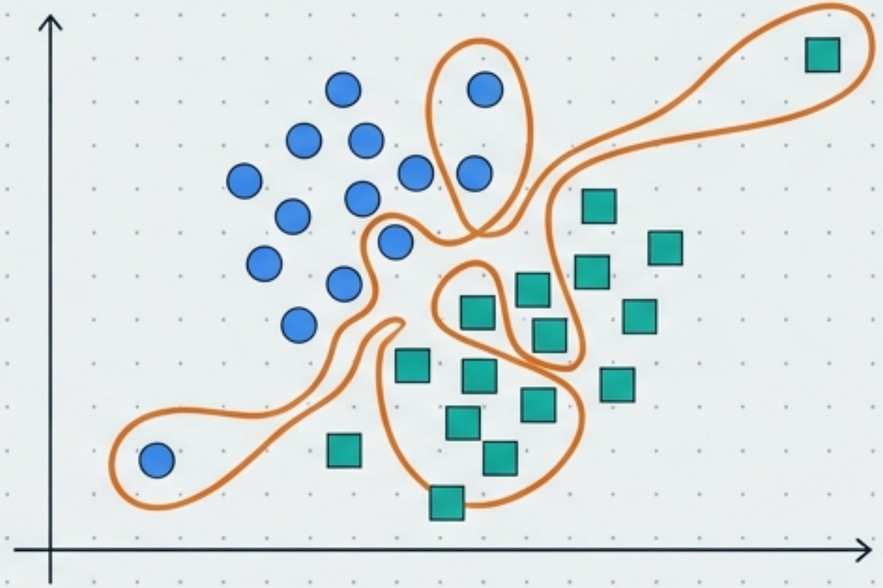
Tuning Regularization (C): The Penalty for Error

Low C (e.g., 0.1)



Soft Margin. “I’m chill, errors are allowed.” Prioritizes simplicity.

High C (e.g., 1000)

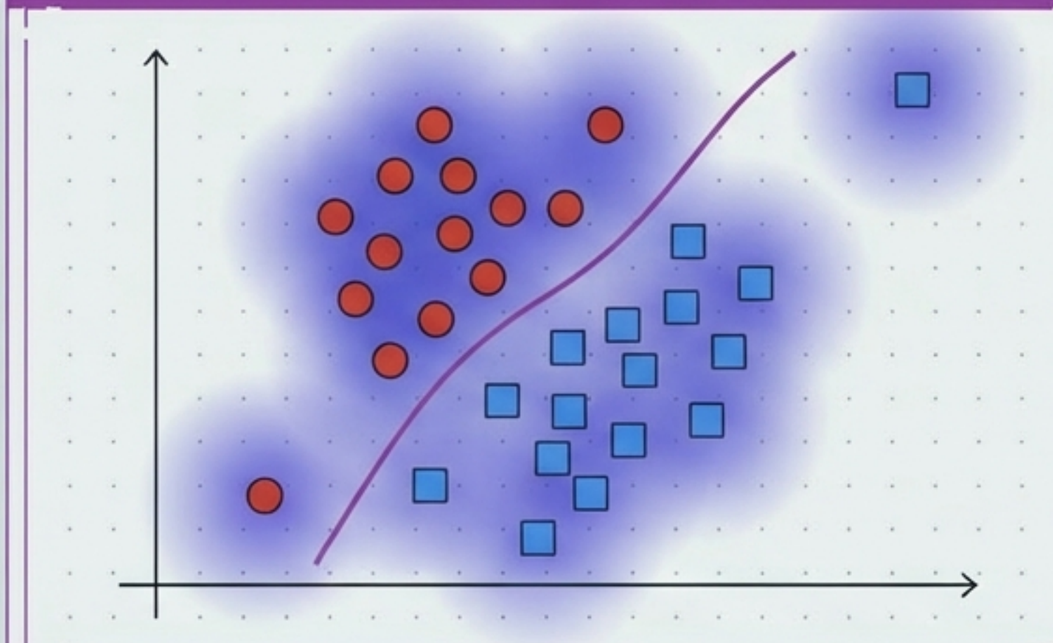


Hard Margin. “Strict, zero errors allowed.” Prioritizes training accuracy (risk of overfitting).

Search Range: Typically Logarithmic (10^{-3} to 10^{5}).

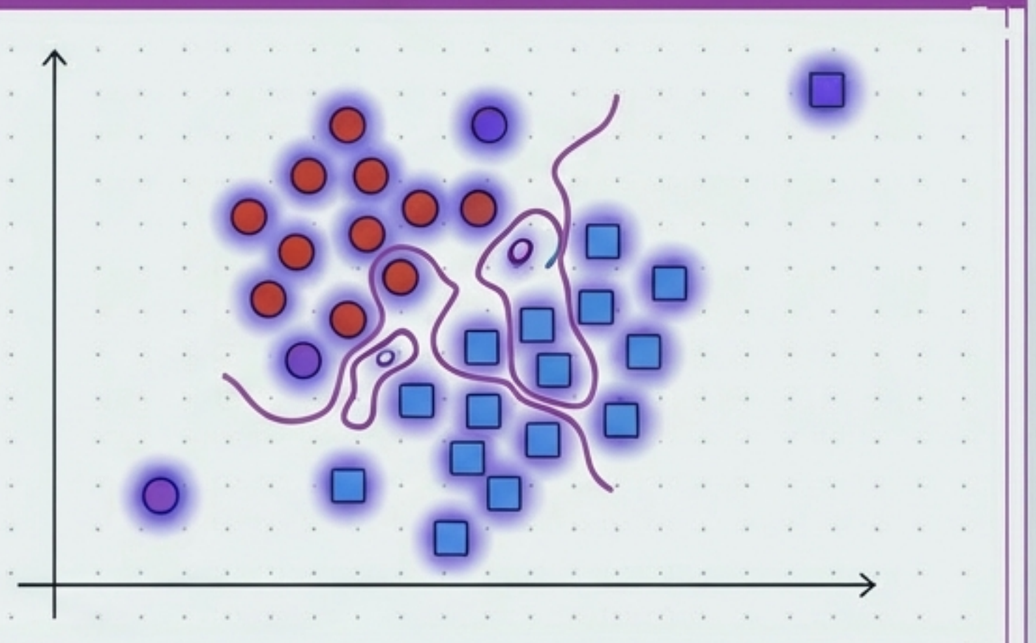
Tuning Gamma (γ): The Radius of Influence

Low Gamma



Far Reach. “I see the whole picture.”
Smoother boundary.

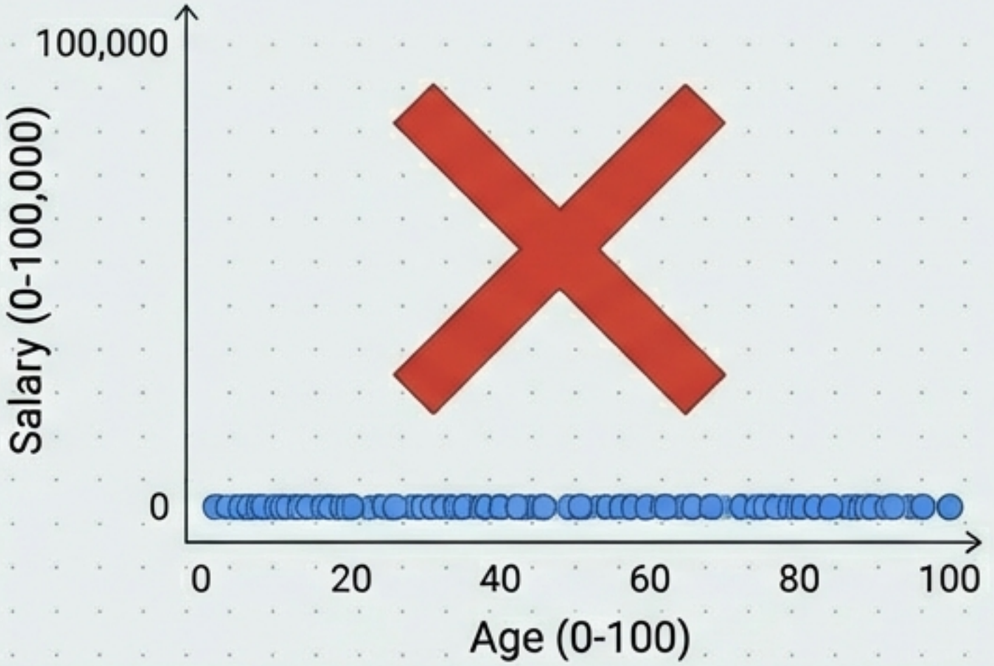
High Gamma



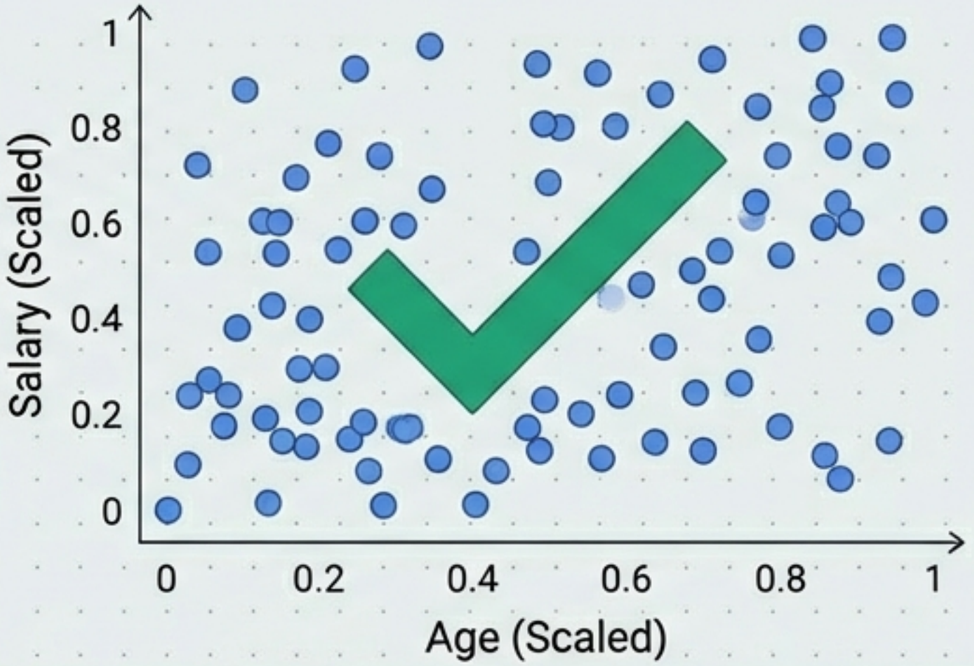
Local Reach. “My neighbor is my world.”
Wiggly boundary.

Default: gamma='scale' (1 / (n_features * X.var()))

Preprocessing is Not Optional



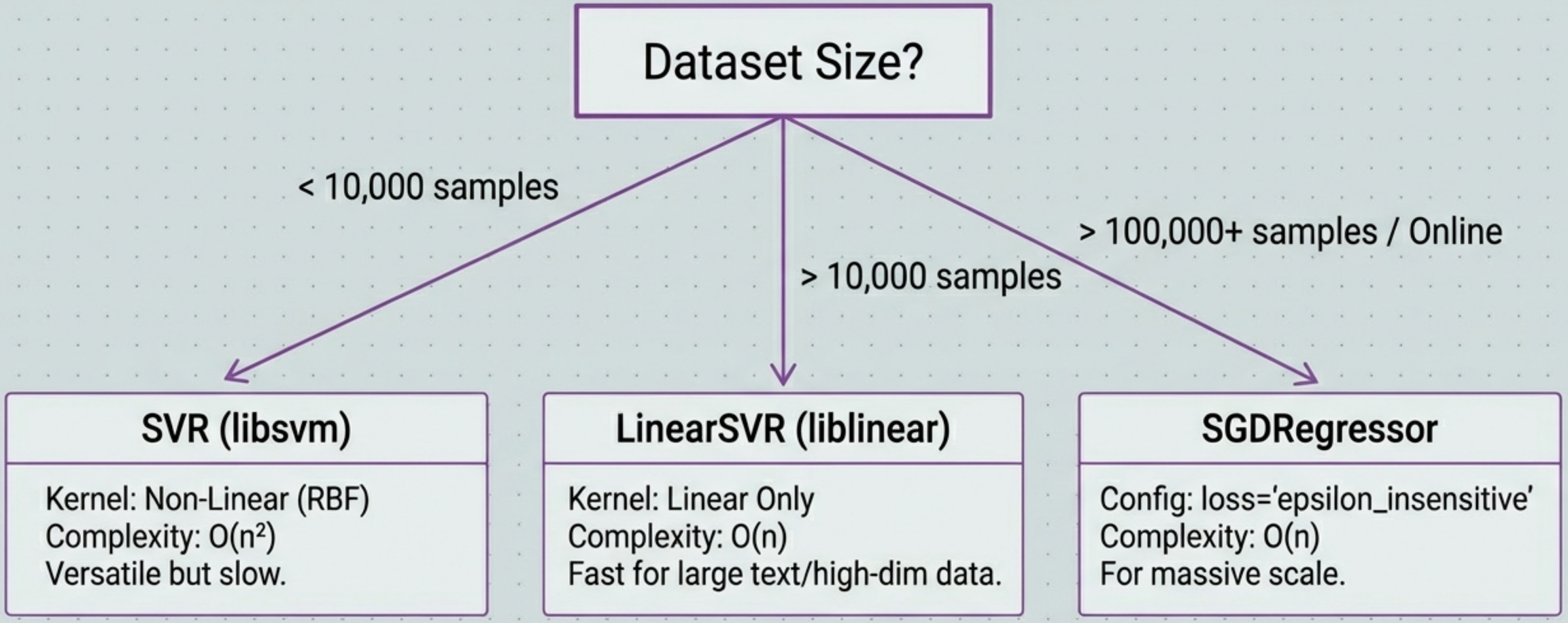
Without scaling, Salary dominates the Euclidean distance calculation.



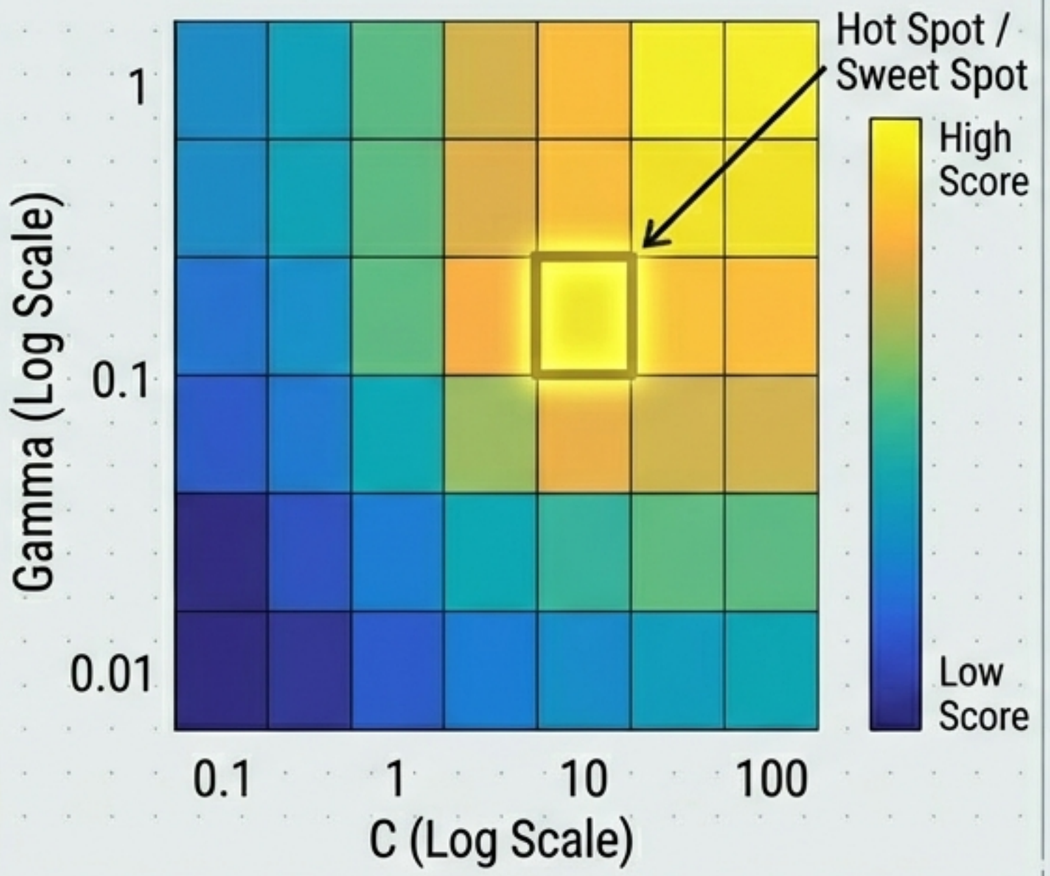
StandardScaler ensures equal feature contribution.

```python
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR

# Always pipeline your scaler!
regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
```

Scale & Complexity: Choosing the Right Estimator



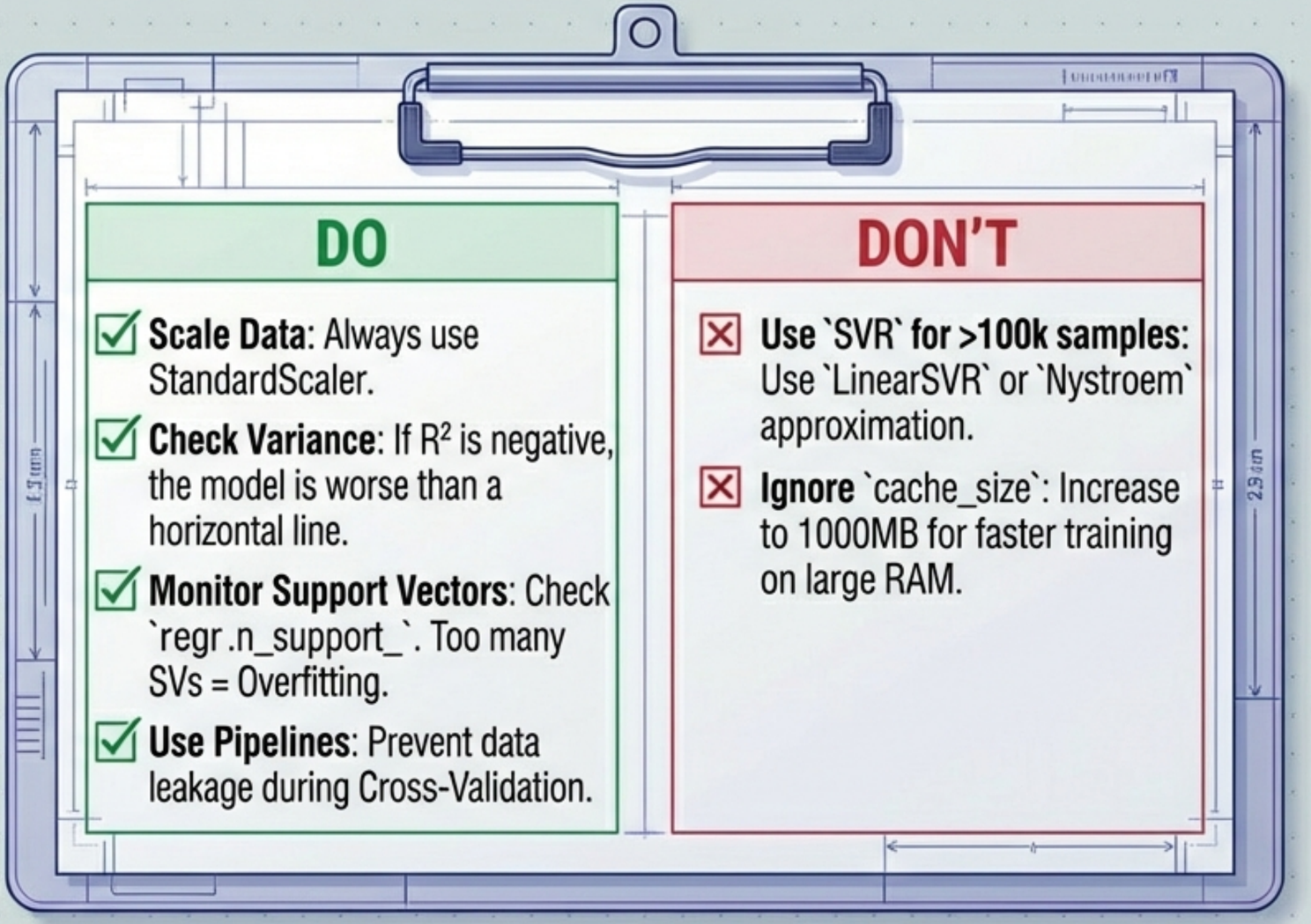
Search Strategies: Finding the Sweet Spot



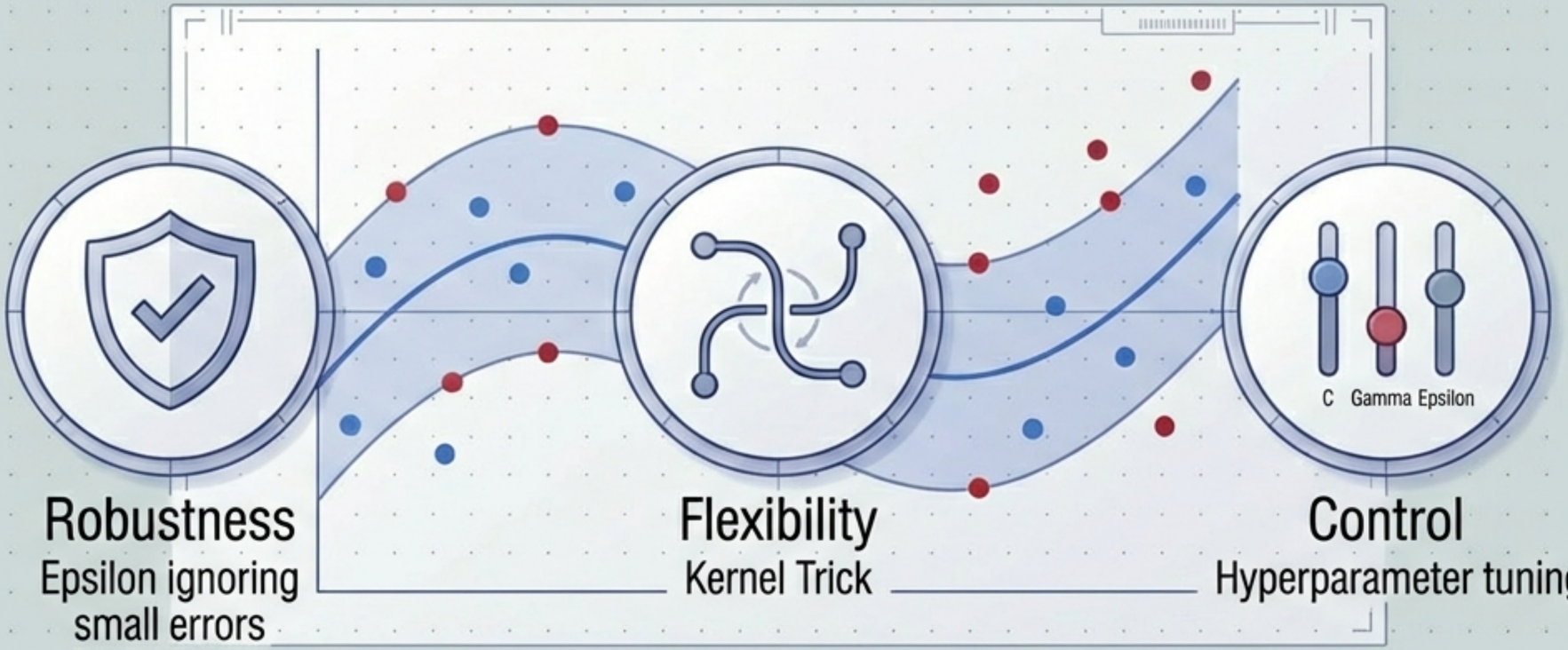

```
param_grid = {
  'svr__C': [0.1, 1, 10, 100],
  'svr__epsilon': [0.01, 0.1, 1],
  'svr__gamma': ['scale', 0.1, 0.01]
}

grid = GridSearchCV(regr_pipeline,
param_grid, cv=5)
grid.fit(X, y)
```

Don't guess. Search.
Use Logarithmic scales for C and Gamma.

The Practitioner’s Checklist



Conclusion: The Power of the Tube



SVR is the tool of choice for robust, non-linear modeling when precision matters more than speed.

```
model = SVR(kernel='rbf', C=10, epsilon=0.1)
```