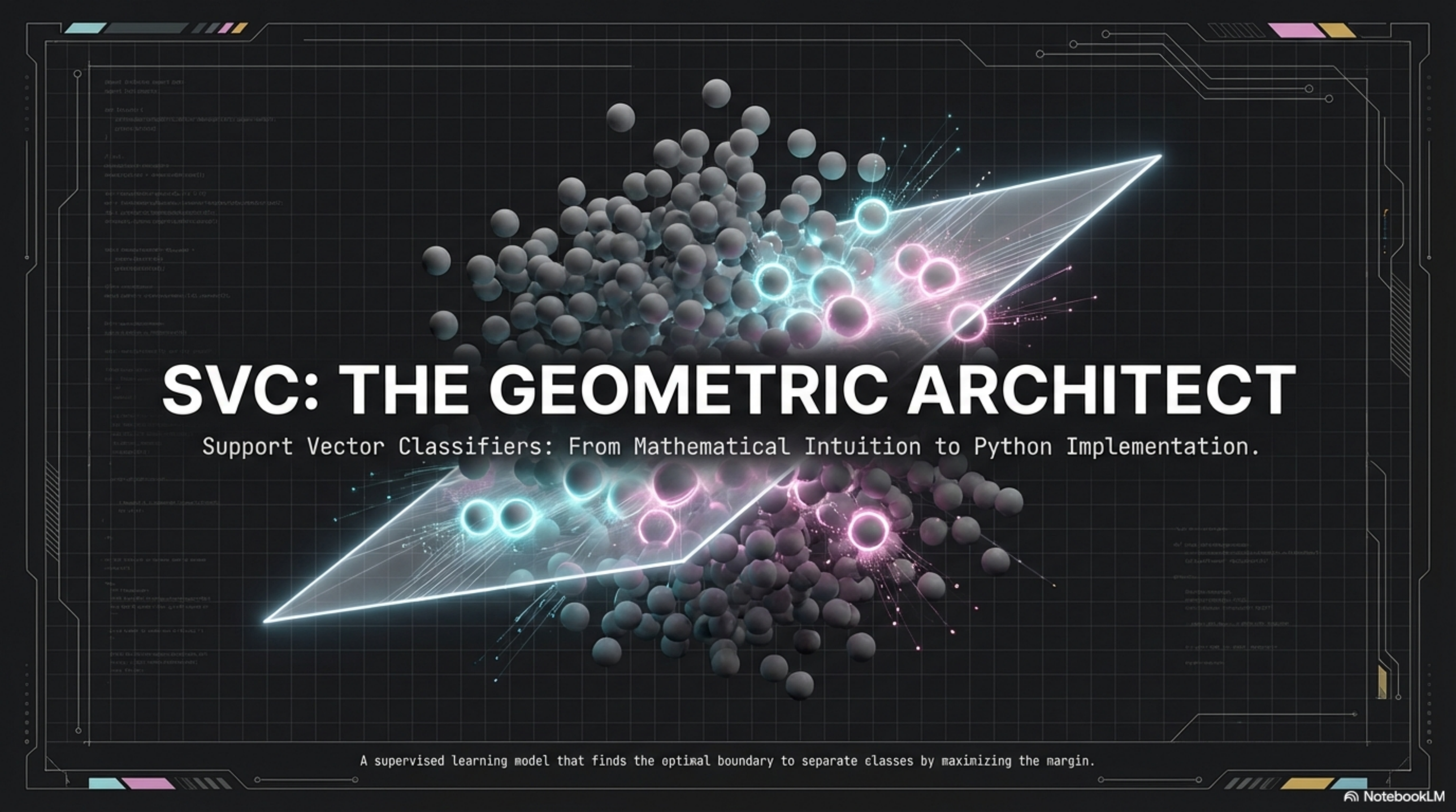
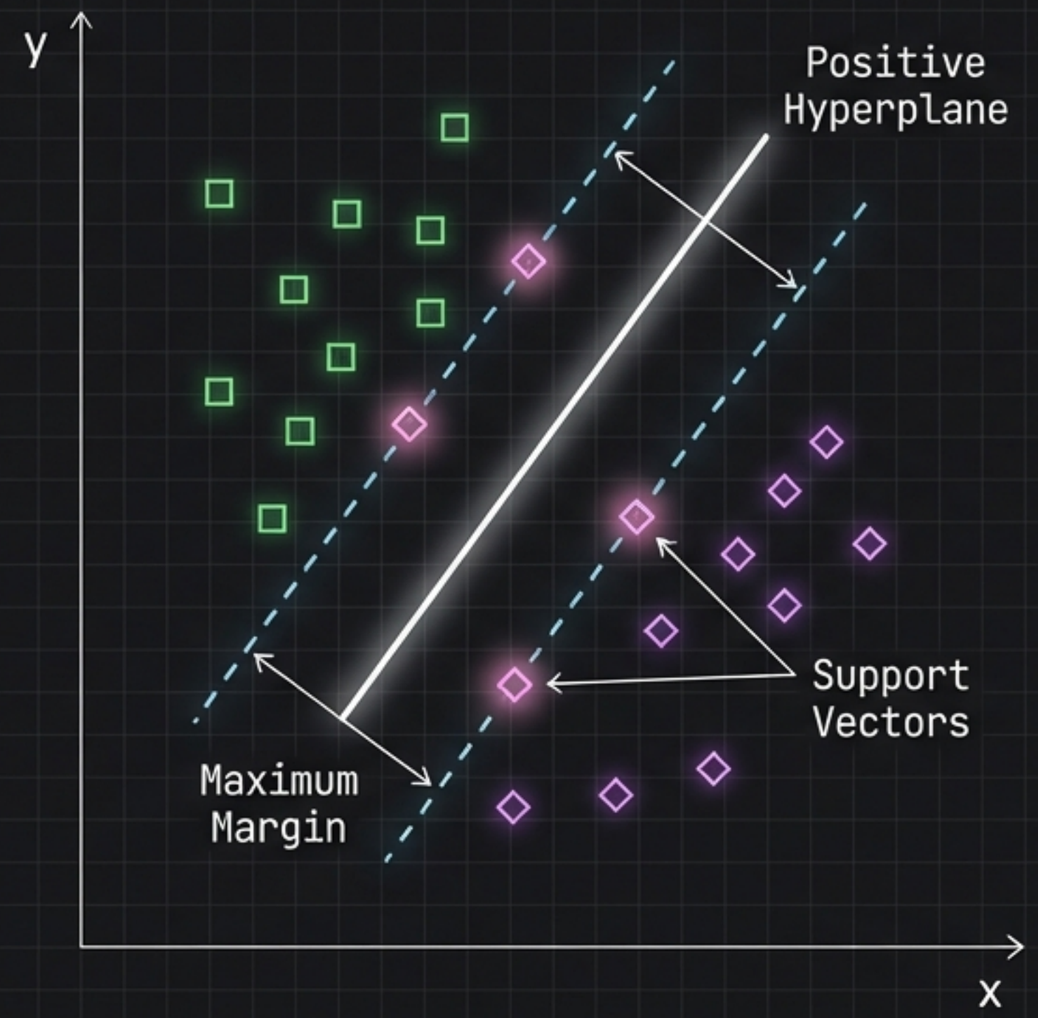

SVC: THE GEOMETRIC ARCHITECT
Support Vector Classifiers: From Mathematical Intuition to Python Implementation.
A supervised learning model that finds the optimal boundary to separate classes by maximizing the margin.

ANATOMY OF A BOUNDARY



The Hyperplane

The decision boundary that separates the classes. Located exactly in the middle of the marginal planes.

Support Vectors

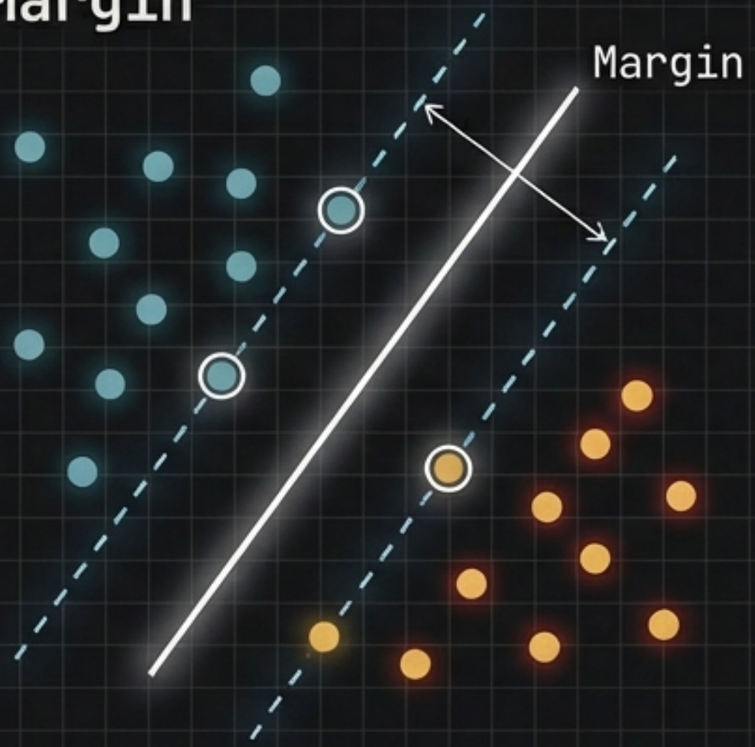
The specific data points closest to the hyperplane. These are the 'load-bearing' points defining the boundary.

Marginal Planes

Parallel lines touching the support vectors. The goal is to maximize the distance (street width) between them.

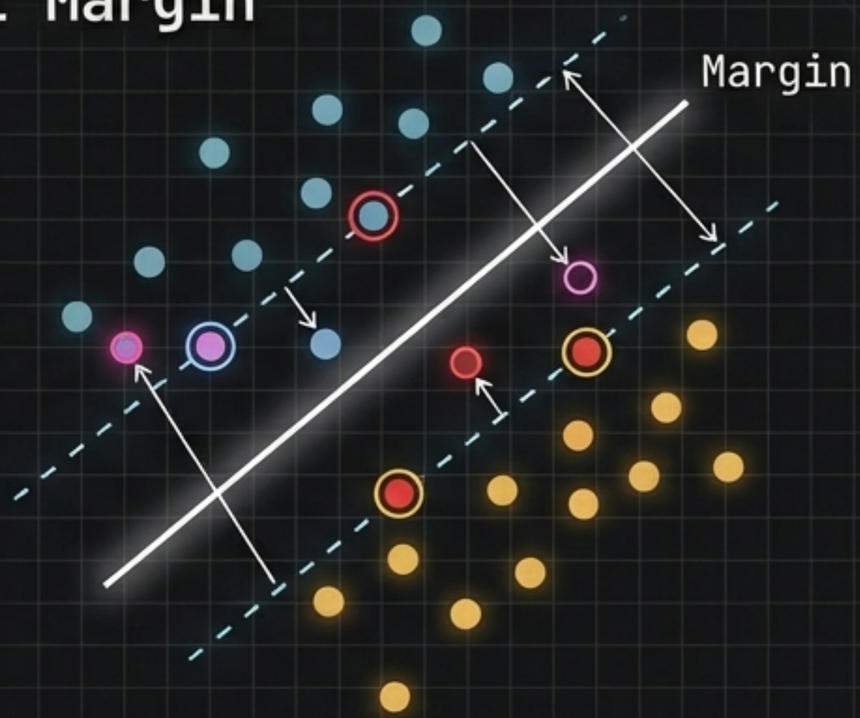
THE REALITY OF DATA: HARD VS. SOFT MARGINS

Hard Margin



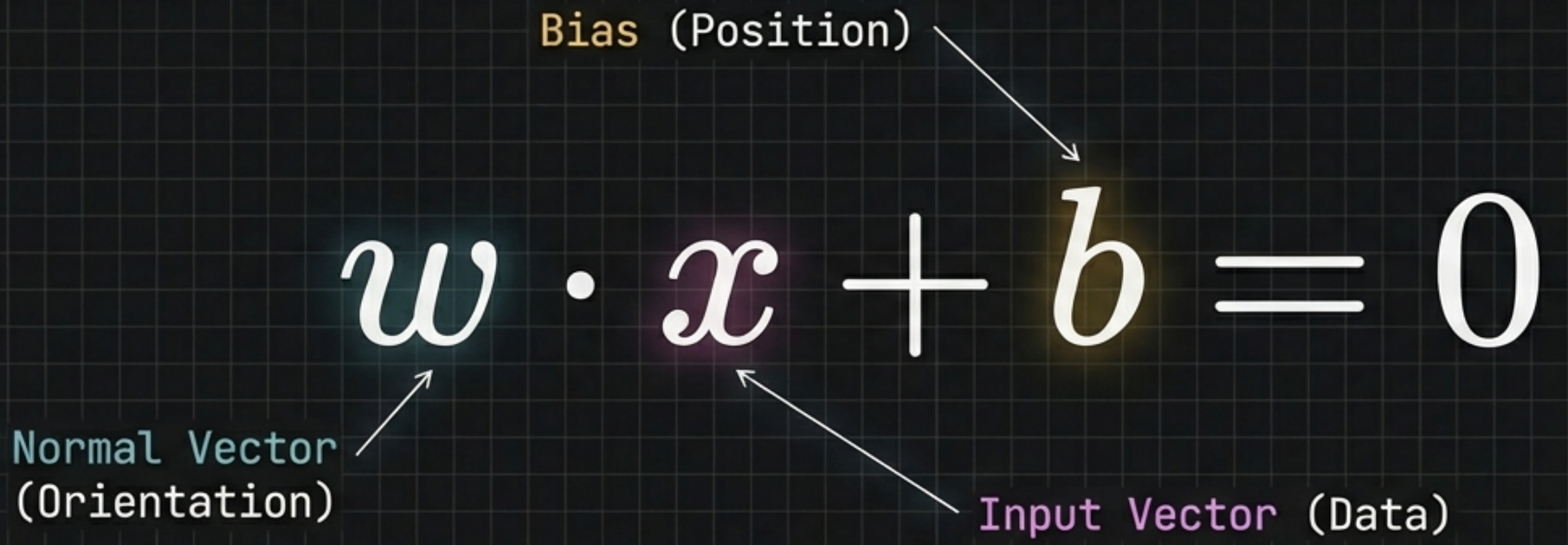
Unforgiving. Requires linear separability. High risk of overfitting to noise.

Soft Margin



Pragmatic. Uses a 'slack variable' to tolerate errors. The 'C' parameter controls the strictness of this tolerance.

DEFINING THE PLANE



The expression $w \cdot x$ is the dot product, computable via matrix multiplication. To generalize, we enforce the constraint: $y_i(w'x + b) \geq 1$, ensuring points lie on the correct side of the margin.

THE OPTIMIZATION GOAL

Objective: Maximize the Street Width.

Maximize Distance ($\frac{2}{|w|}$) → Minimize Magnitude ($\frac{|w|}{2}$)

The Hinge Loss Function

$$Loss = |w|/2 + C \cdot \sum(\zeta_i)$$

- $|w|/2$: Maximizes the margin.
- C: Strictness hyperparameter.
- $\sum(\zeta_i)$: Sum of distances of misclassified points (Slack Variable).

SETTING THE STAGE: DATA GENERATION

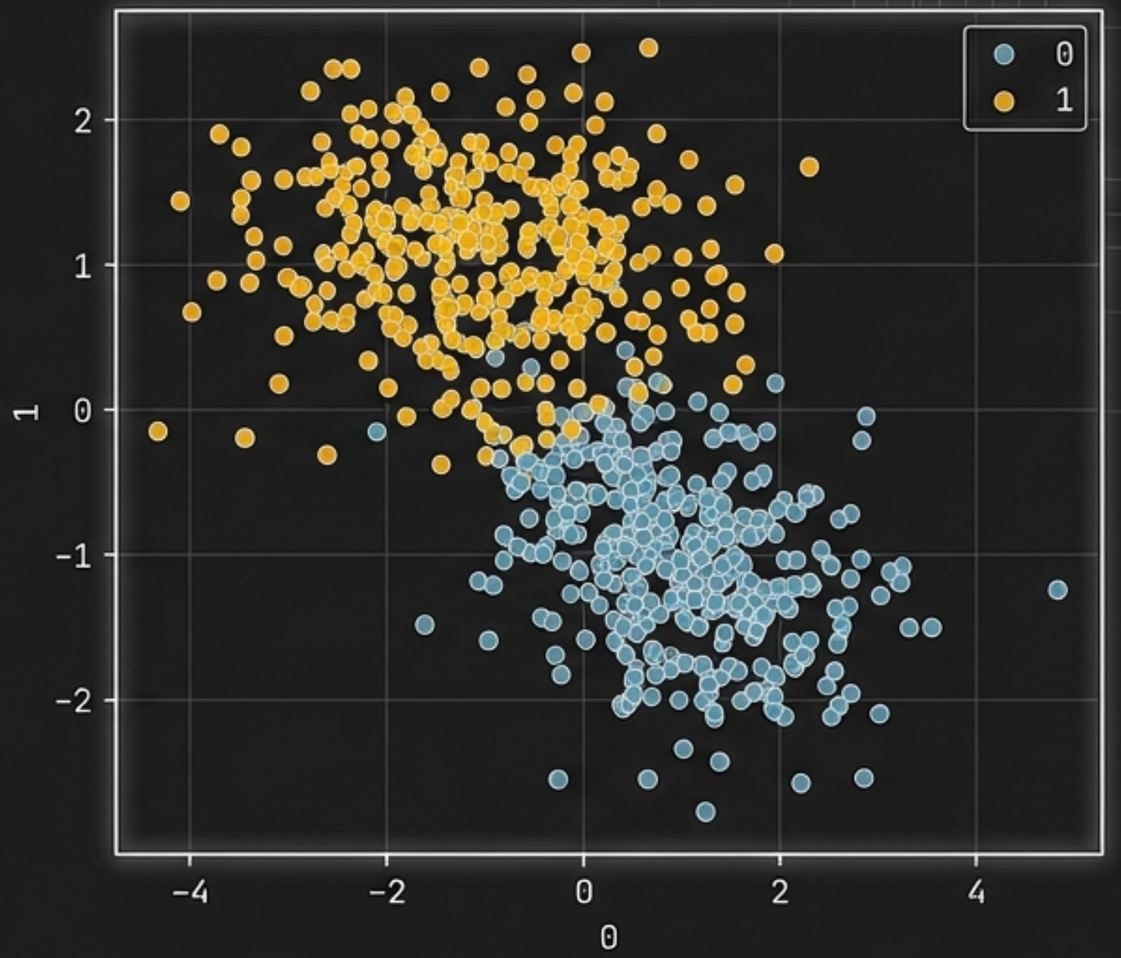
data_generation.py

```python
import pandas as pd
import seaborn as sns
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split

# Generate 1000 samples, 2 features, 2 classes
X, y = make_classification(n_samples=1000, n_features=2, n_classes=2)

# Split data (75% train, 25% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)
```

We use `make_classification` to create synthetic chaos: 1000 samples divided into 2 distinct classes.

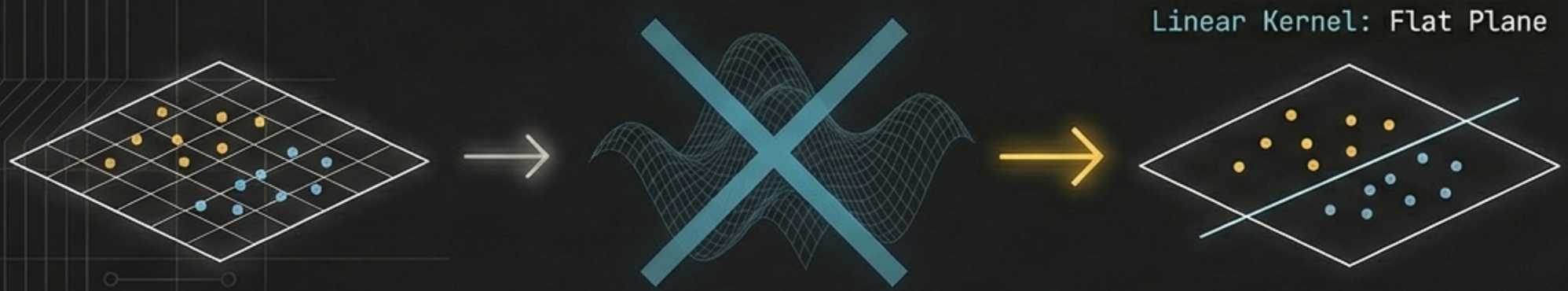
VISUALIZING THE RAW TERRAIN

```
sns.scatterplot(x=pd.DataFrame(X_train)[0],
                y=pd.DataFrame(X_train)[1],
                hue=y_train)
```



TRAINING THE ARCHITECT

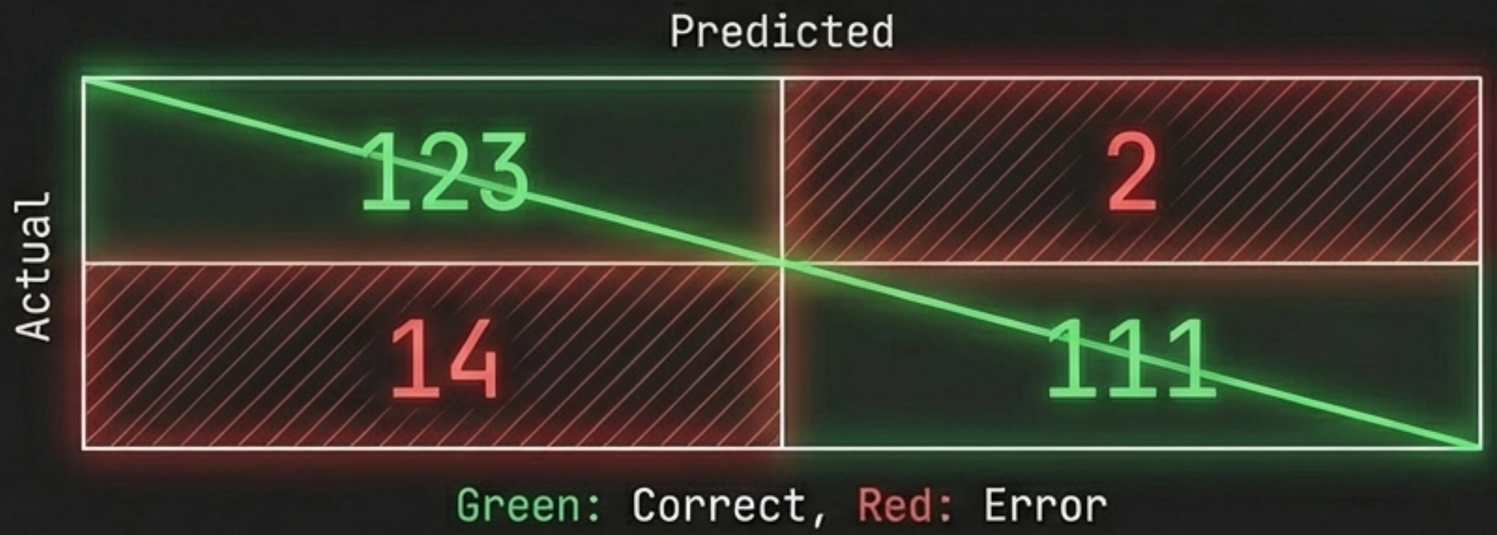
```python
from sklearn.svm import SVC

# Initialize with Linear Kernel (looking for a flat plane)
svc = SVC(kernel='linear')

# Train the model
svc.fit(X_train, y_train)
```



EVALUATING PERFORMANCE

```
y_pred = svc.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

ACCURACY: 93.6%



PLOTTING THE HYPERPLANE

The Math of Visualization

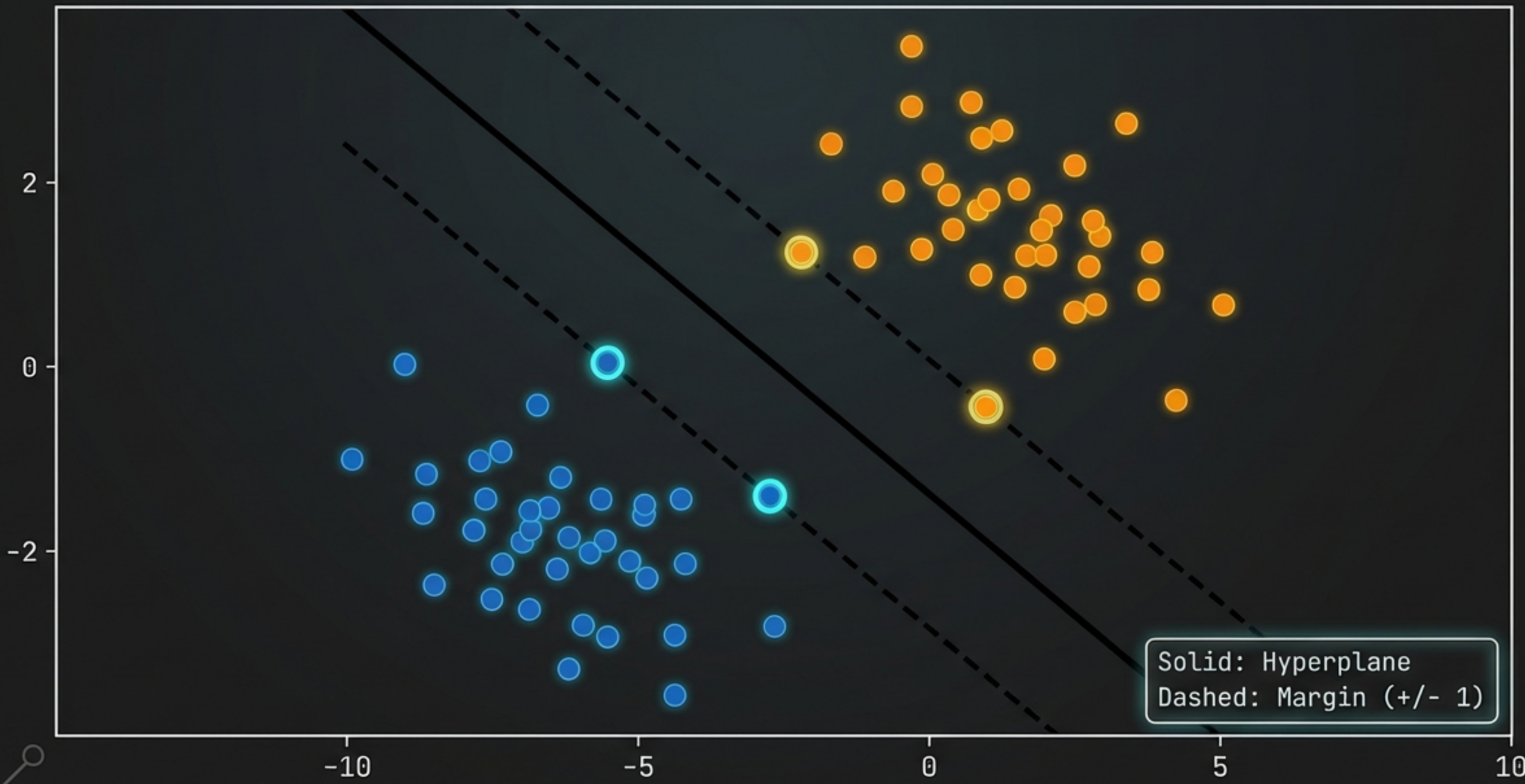
$$y = mx + c$$

Rearranged from $w \cdot x + b = 0$:

$$y = -(w_0 x + b) / w_1$$

```
# Extract weights and bias
w = svc.coef_[0]
b = svc.intercept_[0]

# Calculate y-values (Algebraic rearrangement)
yy = -(w[0]*xx + b) / w[1]

# Calculate Margins (+/- 1 shift)
yy_margin1 = -(w[0]*xx + b - 1) / w[1]
yy_margin2 = -(w[0]*xx + b + 1) / w[1]
```

THE FINAL CONSTRUCTION



KEY TAKEAWAYS

Goal: SVC maximizes the margin between classes to improve generalization.

Mechanics: Uses Support Vectors (closest points) to define the boundary.

Flexibility: "Soft Margin" (C parameter) allows for noise and outliers.

Math: Minimizes $|w|/2$ subject to Hinge Loss constraints.

Code: Visualization requires manually extracting coefficients (w) and intercepts (b).