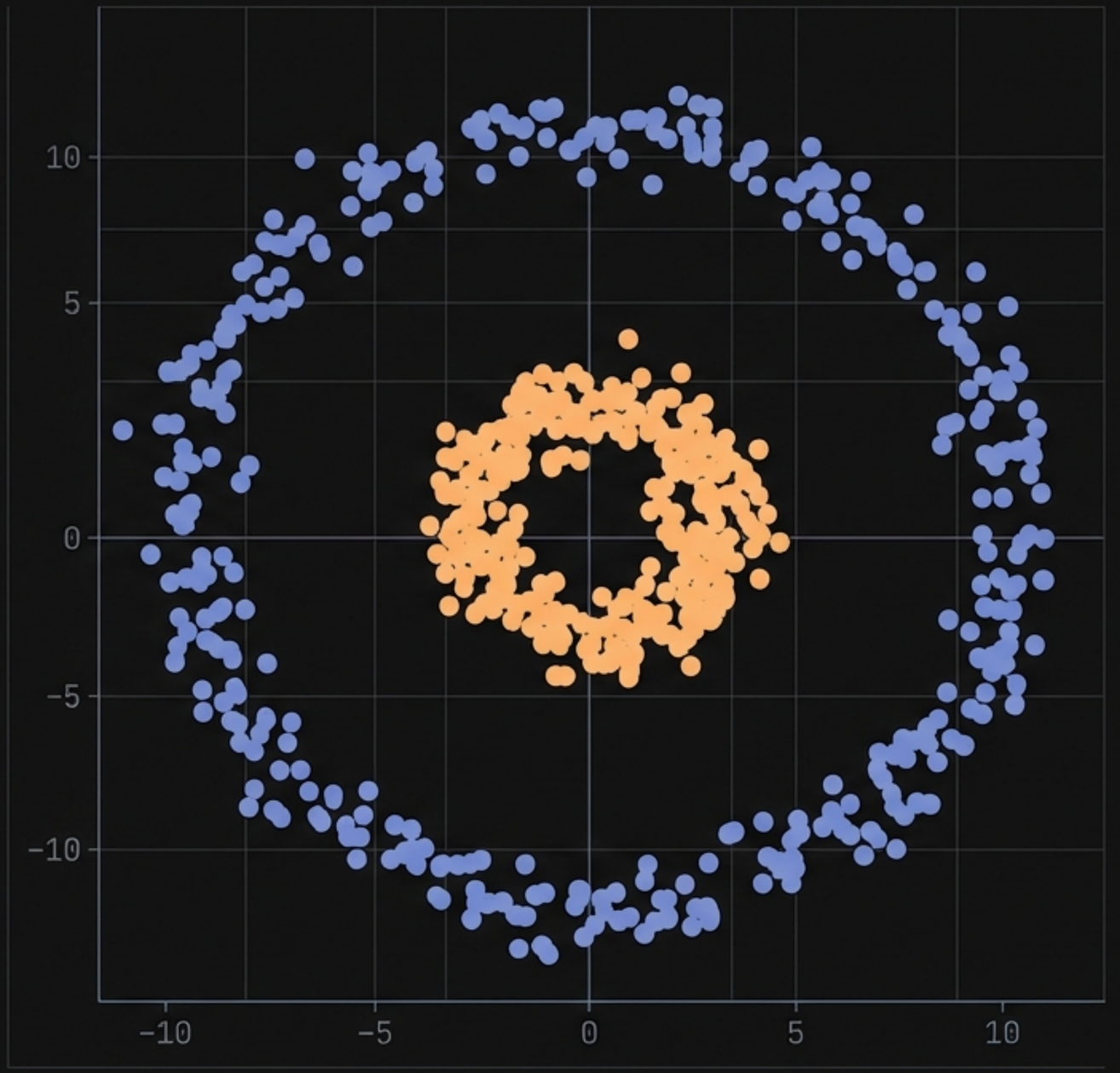
Unlocking Dimensions

A Guide to SVM Kernels

```
// From Linear Failure to
Non-Linear Mastery
```

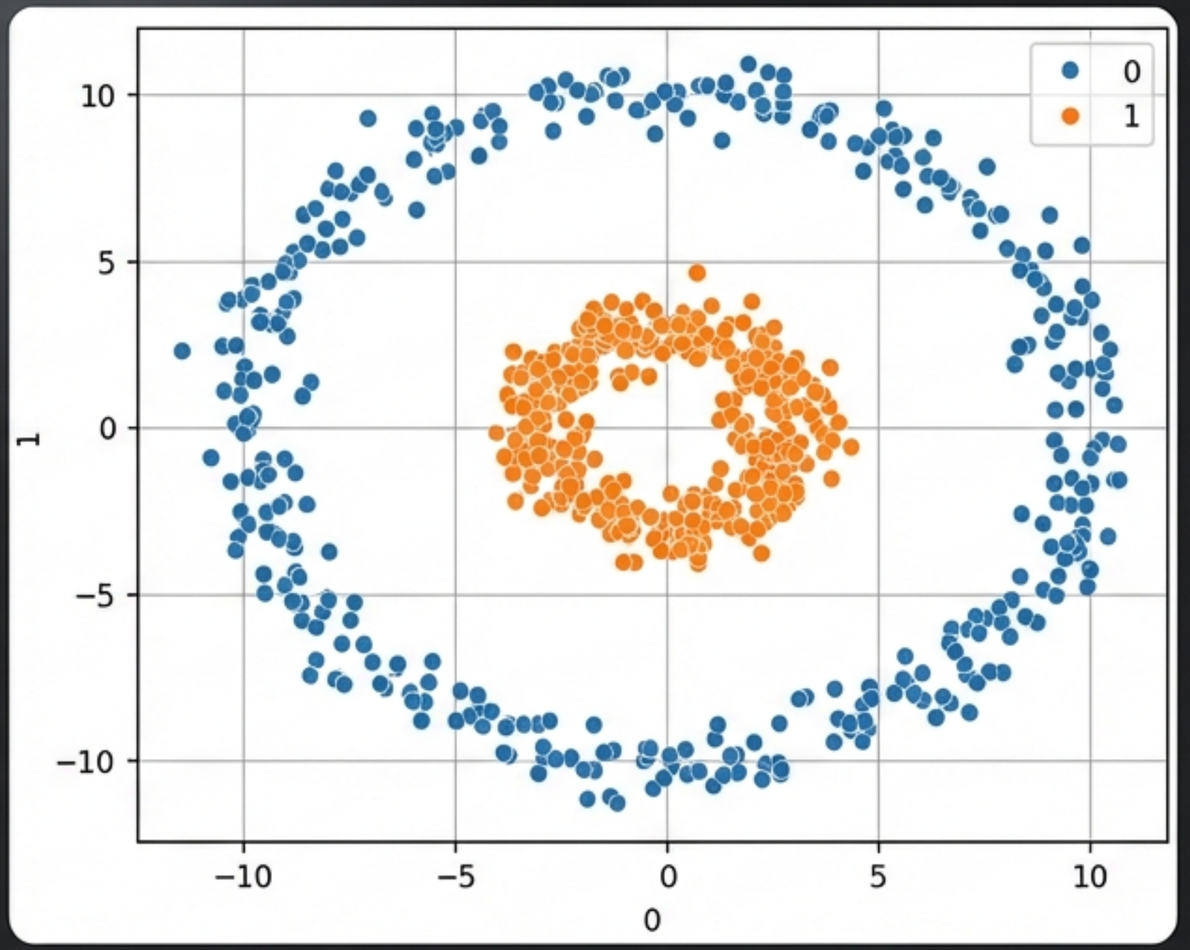
Support Vector Machines (SVMs) are powerful, but they struggle with complex patterns in low dimensions. This deck visualizes the mathematics and code required to escape the Linear Trap.



The Linear Trap: Non-Linearly Separable Data

```python
from sklearn.datasets import
make_circles

# Generating the 'Trap'
X, y = make_circles(n_samples=1000,
                    noise=0.06,
                    factor=0.3,
                    random_state=42)
```



Analysis: Class 1 (Orange) is completely surrounded by Class 0 (Blue). No single straight line can separate these classes in 2D space.

The Failure of the Flat Plane (Linear Kernel)

```
svc = SVC(kernel='linear') # The Naive
Approach

svc = SVC(kernel # The Naive Approach
svc.fit(X_train, y_train)
```

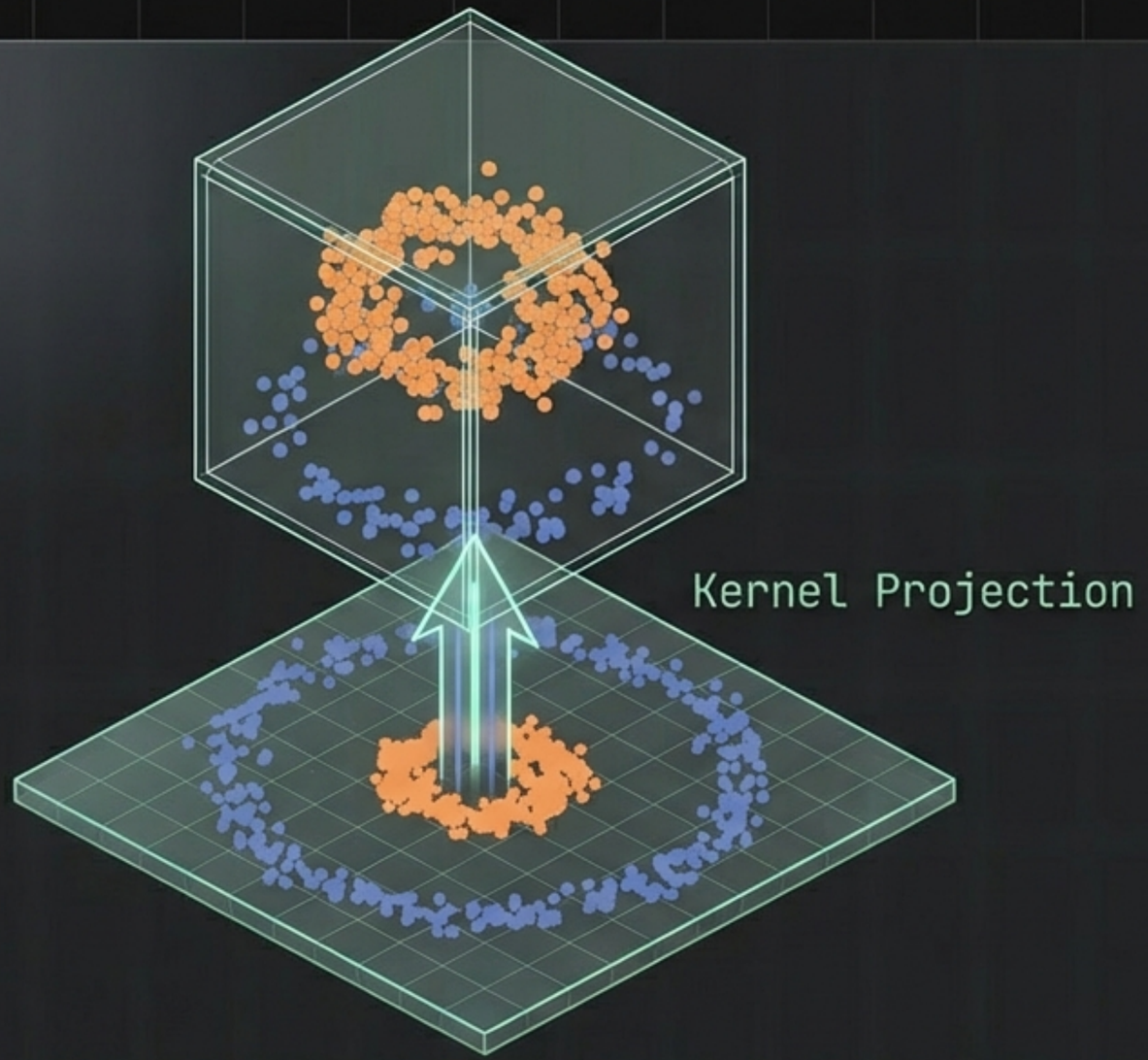
0.66

ACCURACY

	Precision	Recall
Class 0	0.98	0.35
Class 0	0.98	0.35
Class 1	0.59	0.99

The model guesses a midline, correctly identifying one side but failing the other.

The Kernel Trick: Escaping to Higher Dimensions



1. **Linear**: Works only for simple, flat splits.
2. **Polynomial**: Maps data based on geometric degrees (curves).
3. **RBF**: Maps data based on proximity and similarity.

The Polynomial Kernel: Engineering New Dimensions

The Math

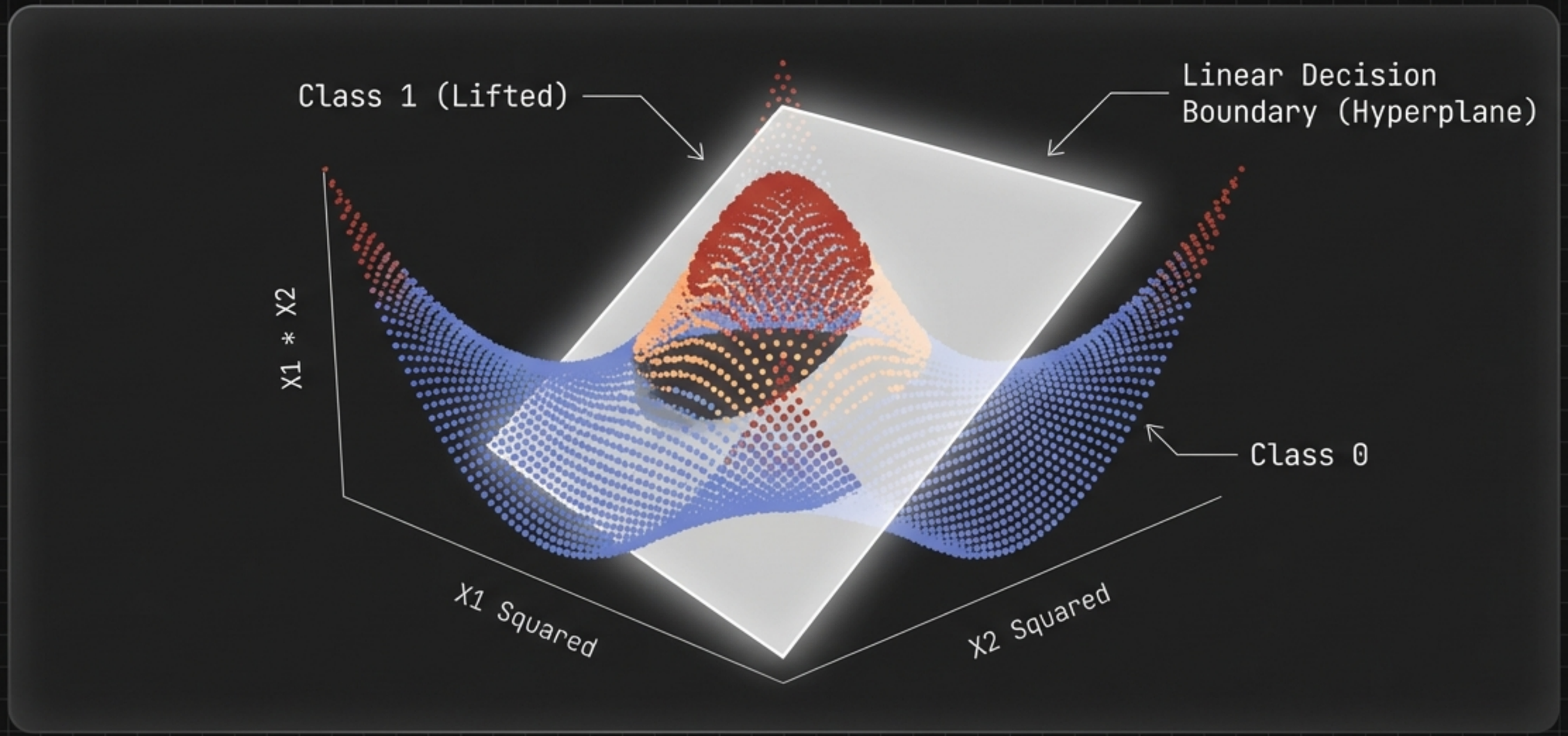
$$K(x, y) = (\gamma * x^T y + c_0)^d$$

By squaring the features, we transform the circle geometry into a parabola. Small values stay low; large values shoot up.

The Code

```
# Manual projection to visualize the math
X['X1_sq'] = X['X1']**2
X['X2_sq'] = X['X2']**2
X['X1*X2'] = X['X1']*X['X2']
```

Visualizing the Ascension



The Polynomial kernel projects data into 3D space. The inner circle becomes a distinct hill, easily separable by a flat sheet.

Polynomial Results: Mathematical Precision

```
svc = SVC(kernel='poly', degree=2)
# Degree 2 aligns with the shape of a circle
svc.fit(X_train, y_train)
```

1.00

ACCURACY

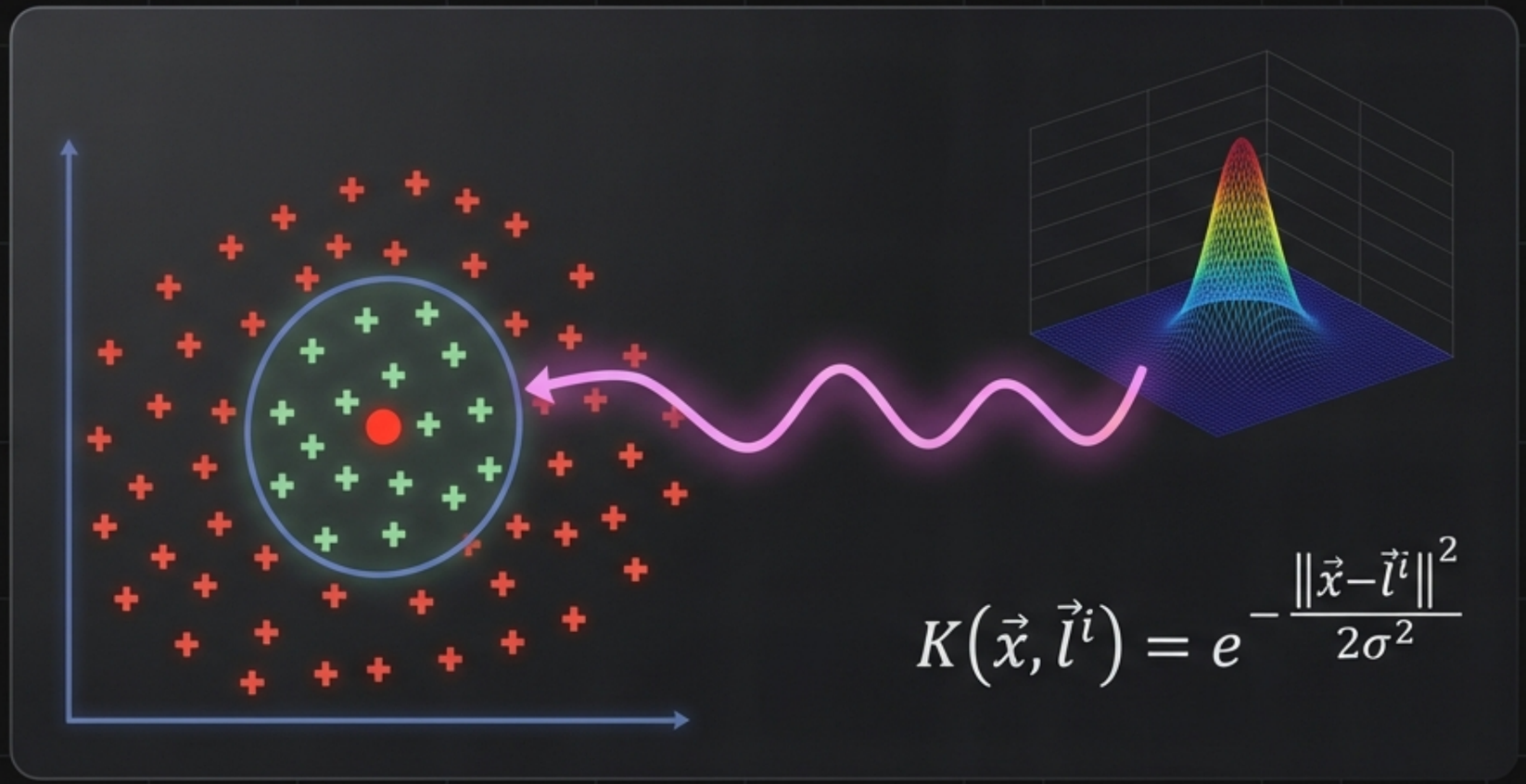
	Precision	Recall
Class 0	1.00	1.00
Class 1	1.00	1.00

By understanding the geometry (circles = squared terms), we achieved perfect classification.

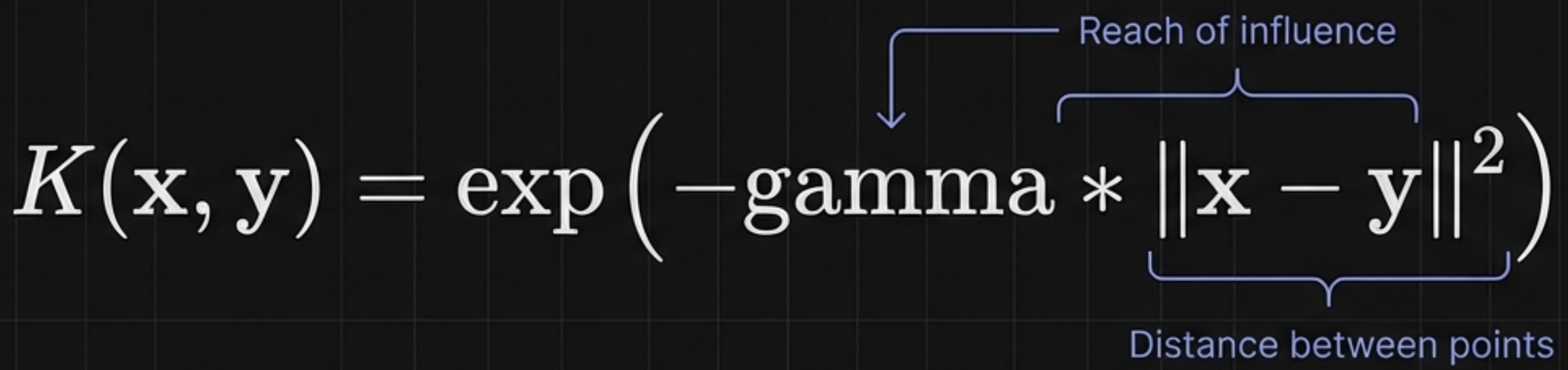
The RBF Kernel: The Universal Solver

Radial Basis Function (Gaussian)

What if we don't know the geometry? RBF measures similarity based on distance. It creates local "hills" of influence around data points.



The Mechanics of Similarity




```
svc = SVC(kernel='rbf') # The Default Powerhouse
svc.fit(X_train, y_train)
```

Points close to the center of the distribution are lifted; points further away drop off.
This naturally isolates the "inner circle" without manual feature engineering.

RBF Results: Effortless Accuracy

“Again we are getting 100% accuracy with this kernel. It inferred the non-linear boundary automatically.”

1.00

ACCURACY

RBF is the robust standard for unknown data distributions.

Kernel Comparison Matrix

LINEAR

Use when data is linearly separable.

66% Accuracy (Failed)

Pros: Fast, simple

POLYNOMIAL

Use when geometric relationship is known.

100% Accuracy

Pros: Explicit mapping to dimensions

RBF (Radial Basis)

Use when shape is complex or unknown.

100% Accuracy

Pros: Universal, adaptable

The Code Recipe

```
# 1. The Trap
X, y = make_circles(noise=0.06, factor=0.3)

# 2. The Linear Failure
model = SVC(kernel='linear') # Acc: 0.66

# 3. The Polynomial Fix (Geometric)
model = SVC(kernel='poly', degree=2) # Acc: 1.0

# 4. The RBF Solution (Universal)
model = SVC(kernel='rbf') # Acc: 1.0
```

Choose the kernel that fits your data's geometry.