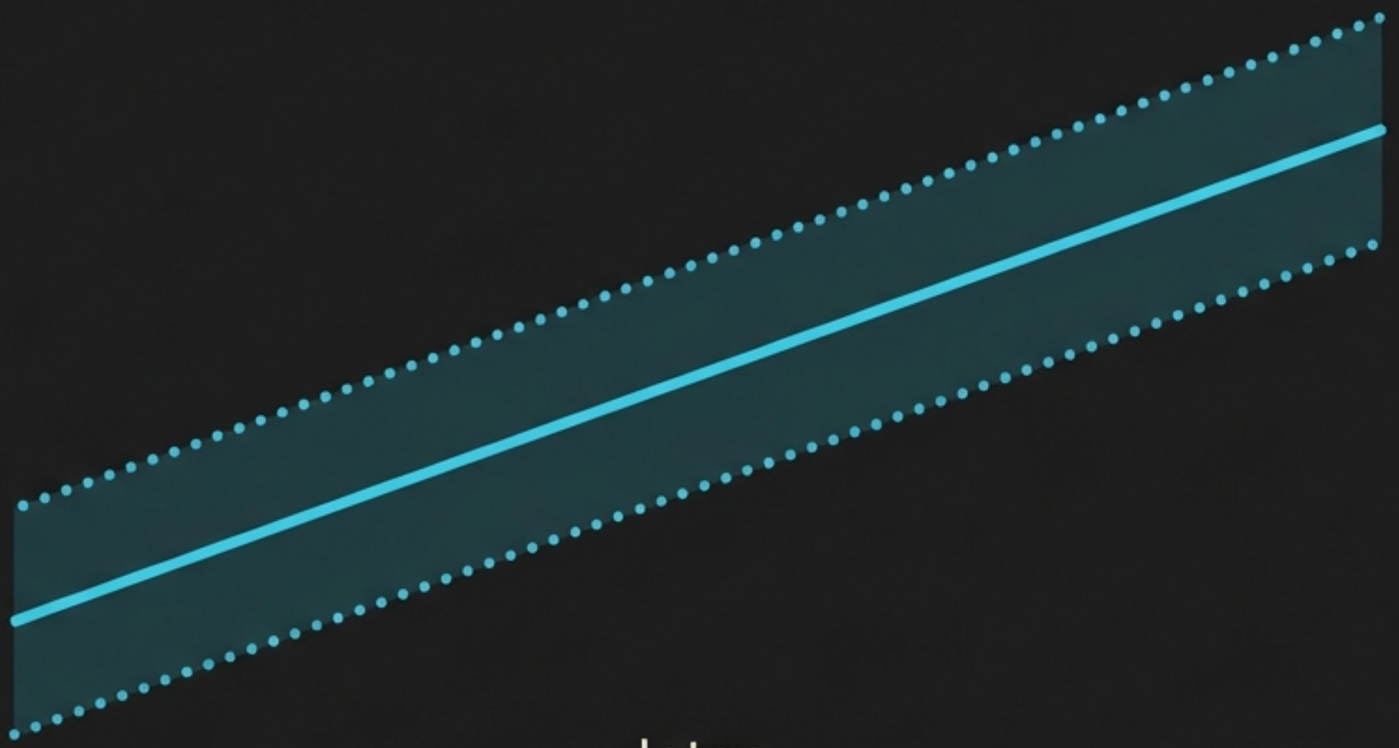
Support Vector Regression



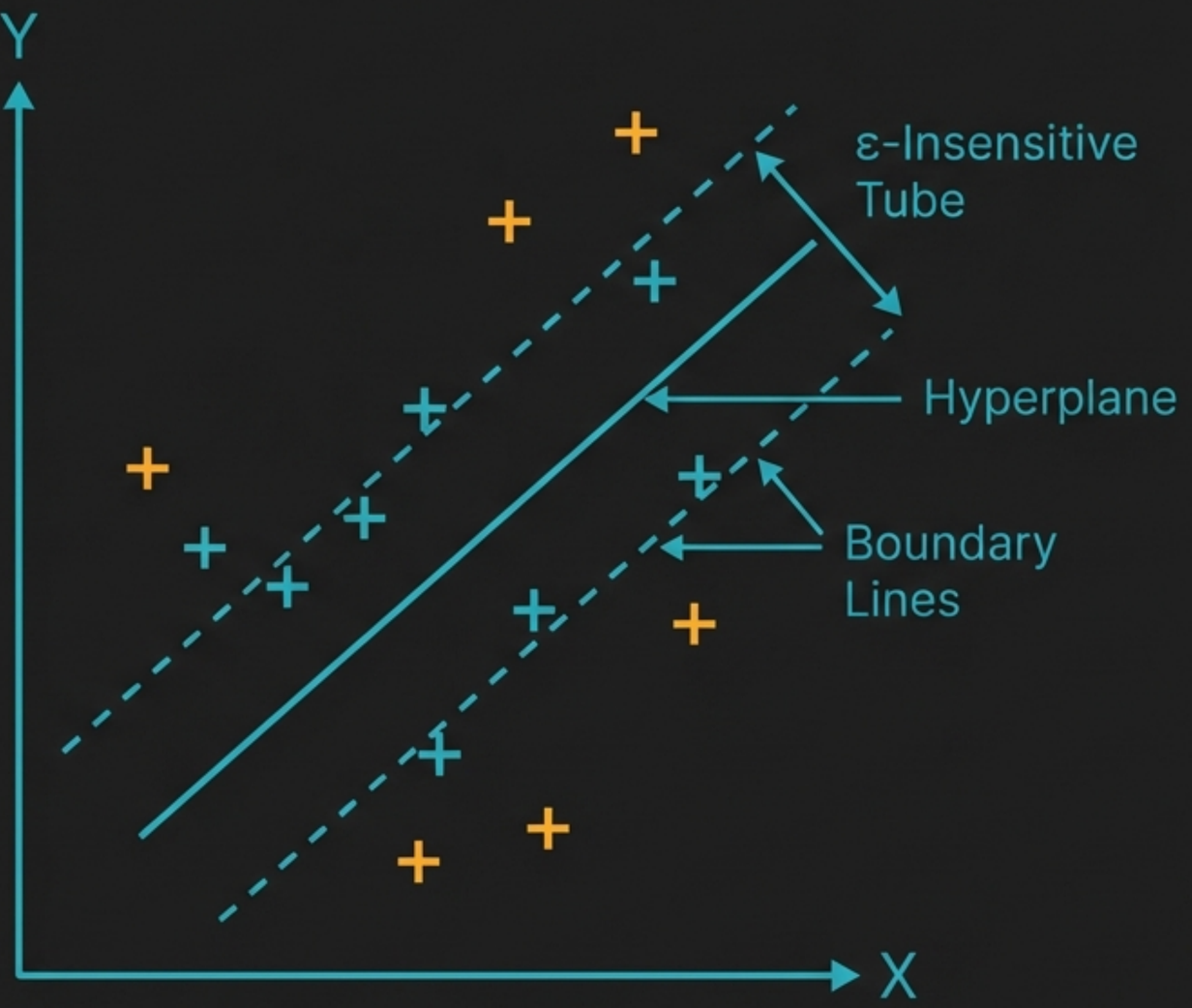
Inter

Mastering the Epsilon-Insensitive Tube

Predicting continuous values by finding a ‘best-fit’ hyperplane that tolerates a specific margin of error (ε). A method that ignores safe data to focus on critical outliers.

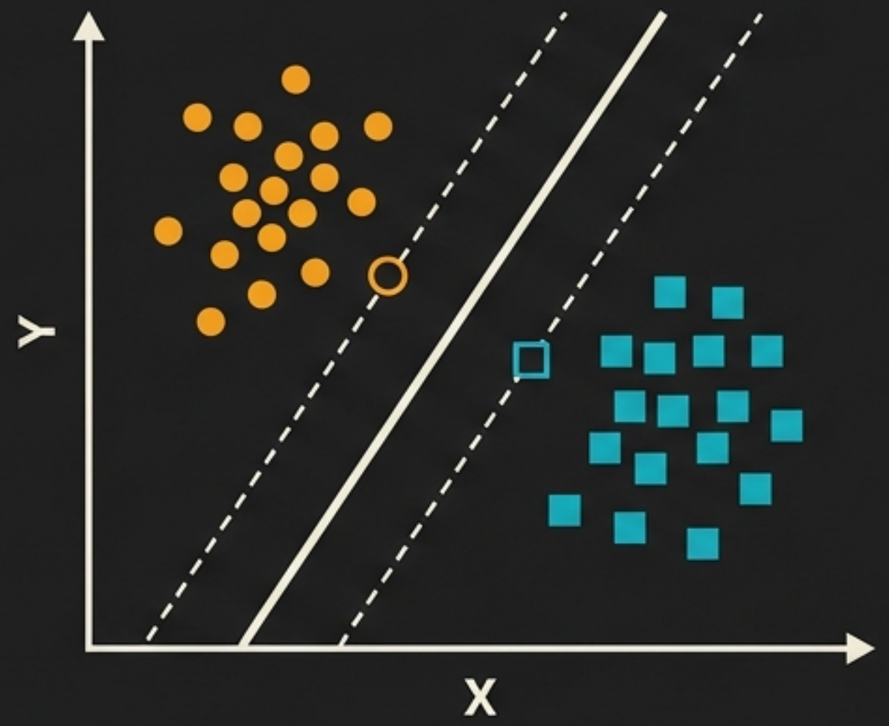
The Street Analogy

Unlike standard regression, which minimizes error for every point, SVR defines a 'street' of width ε (epsilon). If a data point fits inside the street, we disregard it. We only care about points strictly outside the street. SVR prioritizes tolerance to ensure better generalization.



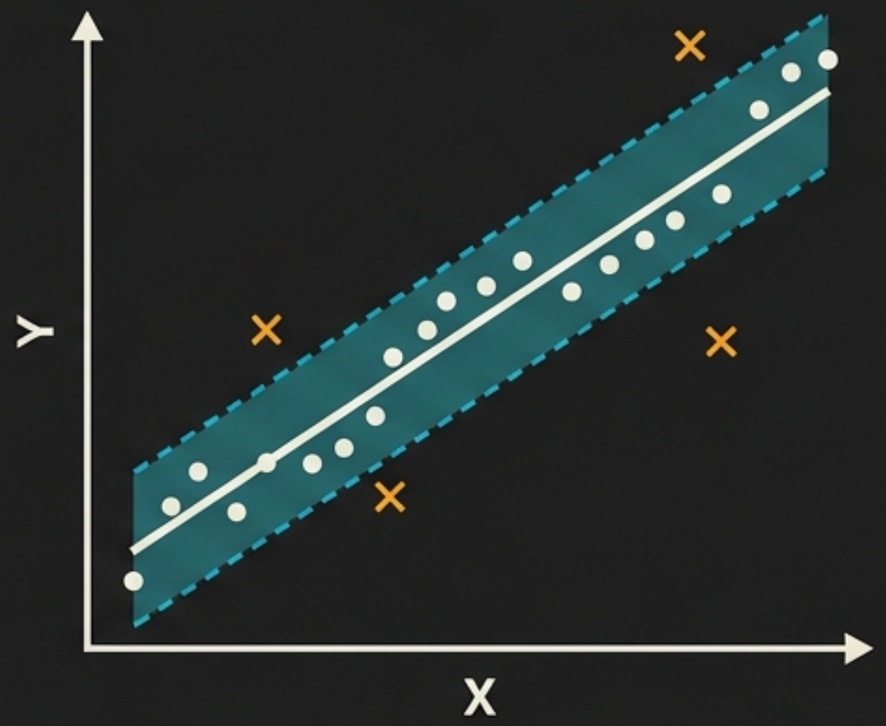
Flipping the Objective: SVC vs. SVR

Support Vector Classification (SVC)



Objective: SEPARATE classes with a maximal margin.

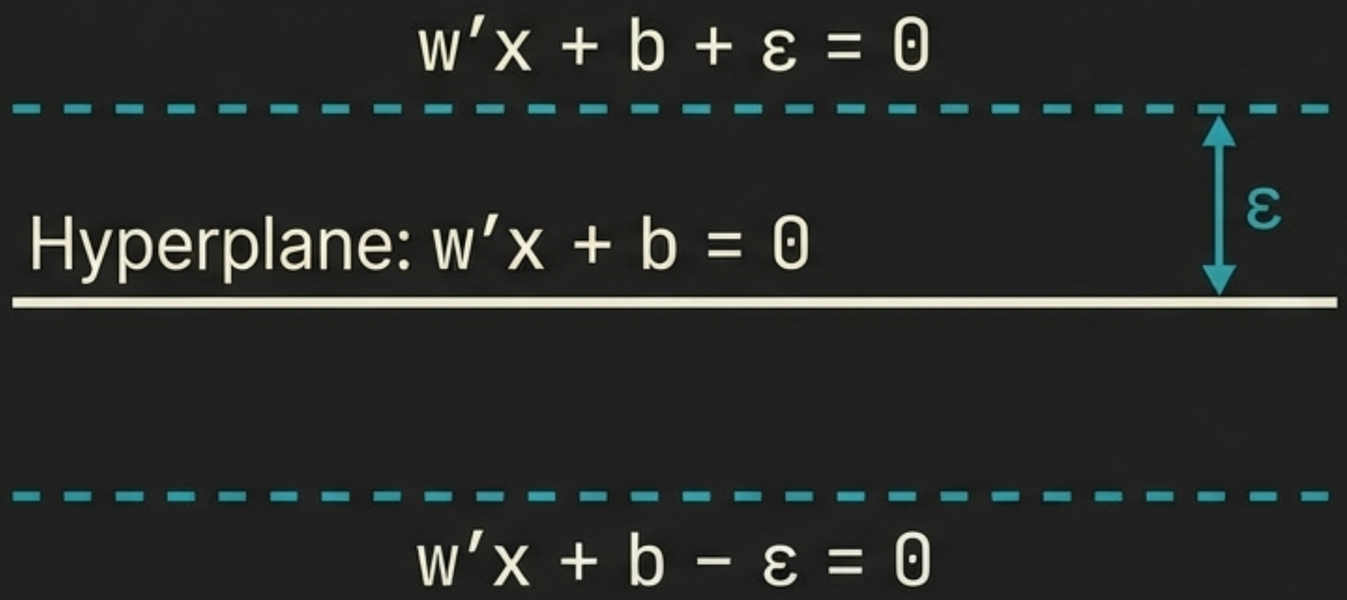
Support Vector Regression (SVR)



Objective: FIT a tube around the trend within a tolerance.

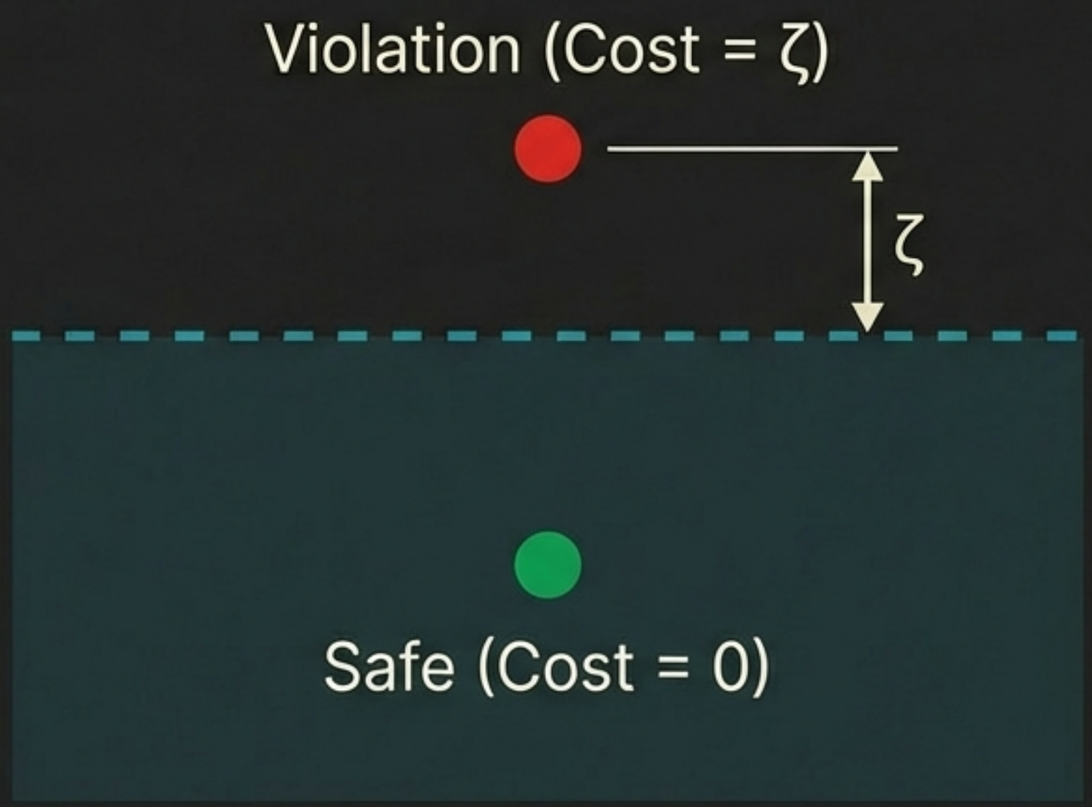
The mathematical engine is nearly identical, but the objective function inverts from separation to fitting.

Geometry of the Tube



Epsilon (ε) is a hyperparameter that defines the range of error we tolerate. Inside this range, the penalty is zero.

The Cost of Violation



The Constraints:

For points within ε:
$|y - wx| \leq \varepsilon$ (Cost = 0)

For points outside ε:
$|y - wx| \leq \varepsilon + \zeta$

ζ represents the “slack”—the distance strictly beyond the allowed margin. We only “pay” for this excess error.

The Objective Function

$$\text{Minimize: } \underbrace{\frac{1}{2}|\mathbf{w}|^2}_{\text{Regularization.}} + \underbrace{C * \sum(\zeta_i)}_{\text{Error Penalty.}}$$

Regularization. Keeps the model flat and simple.

Error Penalty. Minimizes the sum of slack variables (outliers).

The Trade-off: 'C' is a hyperparameter. A high C penalizes outliers heavily (risk of overfitting). A low C allows more slack (smoother tube).

Data Ingestion

```
1  import pandas as pd
2  import numpy as np
3  import matplotlib.pyplot as plt
4
5  data = pd.read_csv('../data/height-weight.csv')
6  X = data['Weight']
7  y = data['Height']
```

Goal: Predict Height (y) using Weight (X).

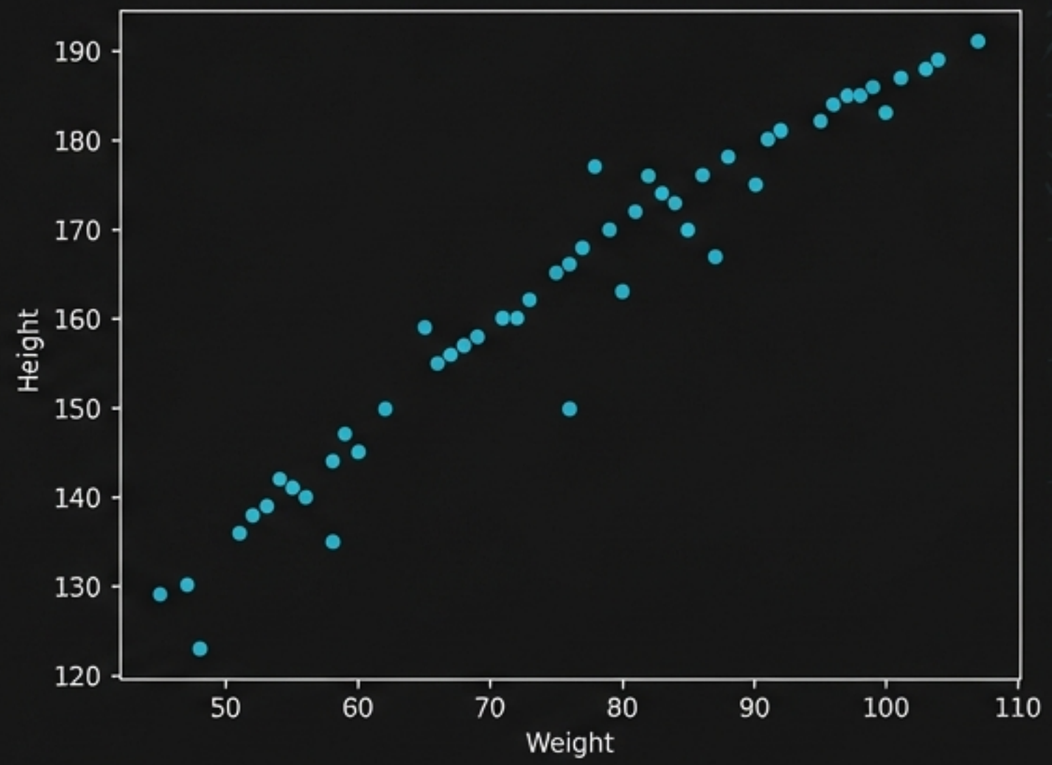
Dataset: “height-weight.csv” containing samples with weights (e.g., 45-107kg) and heights (e.g., 120-191cm).

Visual Exploration

```python
from sklearn.model_selection import train_test_split

# Split data 75% training / 25% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25
)

sns.scatterplot(x=X_train, y=y_train)
```



Observation: The relationship is clearly linear. This justifies using kernel='linear' in the SVR model.

Configuring the Regressor

```
from sklearn.svm import SVR

# Epsilon = 5 means 5 units of tolerance
svr = SVR(kernel='linear', epsilon=5)
```

Matches the visual trend of the data.

The Tube Width. If prediction is within 5cm, error is zero.

This parameter directly controls the geometric ‘tube’ visualized in the previous diagrams.

The Reshape Constraint


```
# Scikit-Learn expects a 2D array (Matrix)
X_train = np.array(X_train).reshape(len(X_train), 1)
X_test = np.array(X_test).reshape(len(X_test), 1)

svr.fit(X_train, y_train)
```

The Fix: Passing a 1D pandas Series often causes dimension errors. We must reshape the data into a 2D matrix, even for a single feature.

Training and Evaluation

```
y_pred = svr.predict(X_test)

from sklearn.metrics import r2_score
score = r2_score(y_pred, y_test)
print(score)
```

```
>>> 0.92344273
```

An R^2 score of ~0.92 indicates the model explains 92% of the variance. The linear 'tube' fits the trend well.

Reconstructing the Hyperplane

```
# Extract learned parameters
w = svr.coef_[0][0]
b = svr.intercept_[0]

# Generate input range for plotting
xx = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)

# Equation of the center line
yy = w * xx + b
```

To visualize the result, we extract the learned weights (w) and bias (b) to manually reconstruct the line equation y = wx + b.

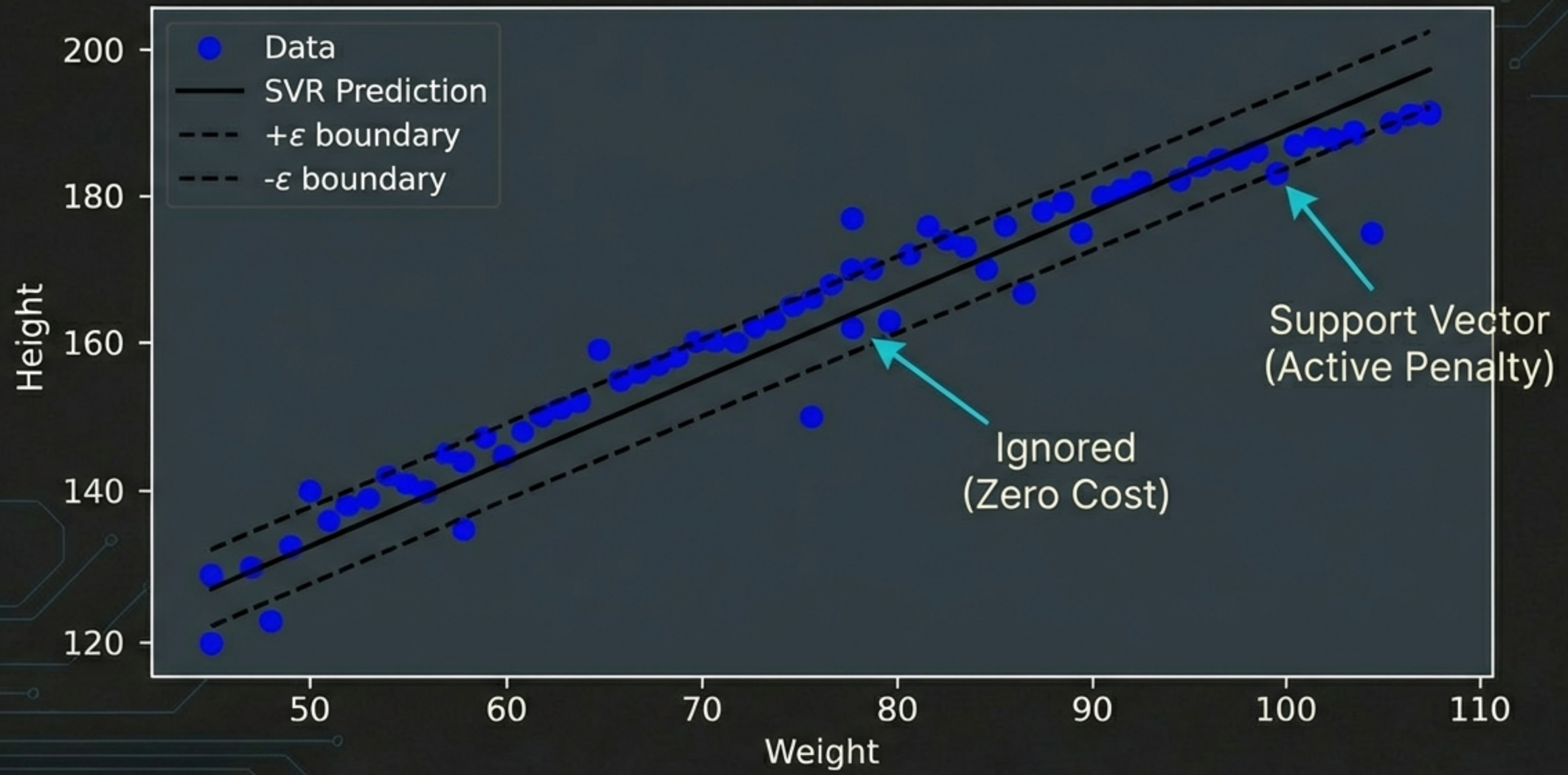
Reconstructing the Tube

```
# Epsilon tube boundaries
yy_upper = yy + svr.epsilon
yy_lower = yy - svr.epsilon

plt.plot(xx, yy, 'k-', label='SVR Prediction')
plt.plot(xx, yy_upper, 'k--', label='+ε boundary')
plt.plot(xx, yy_lower, 'k--', label='-ε boundary')
```

We calculate the upper and lower boundary lines by explicitly adding and subtracting the epsilon value from the prediction line. This generates the visual proof of our tolerance zone.

The Final Model



Key Takeaways

SVR fits a flexible tube, not just a single line.

ε

Epsilon controls the tube width (error tolerance).

Slack variables (ζ) penalize only the points outside the tube.

Great for capturing trends while ignoring noise within the tolerance.