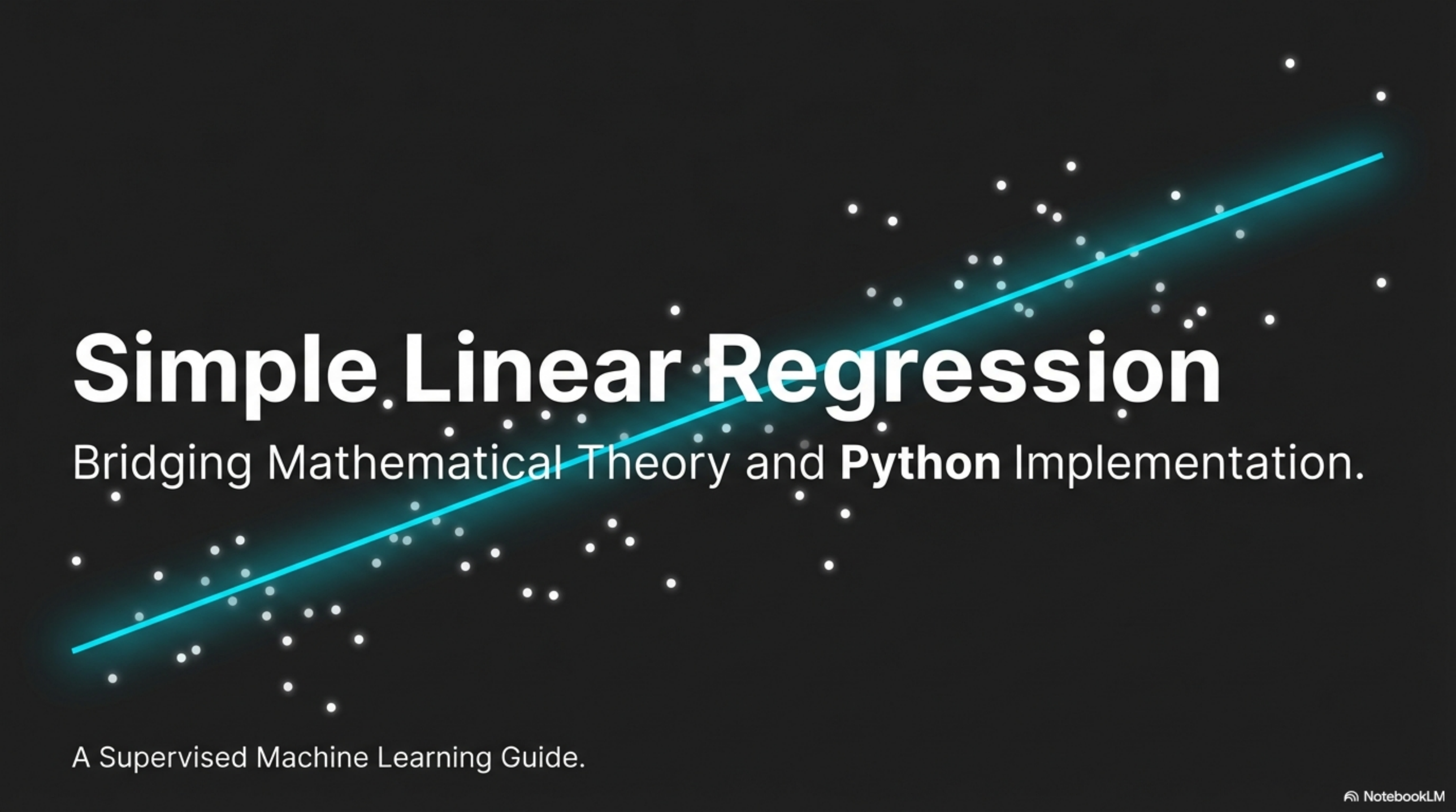
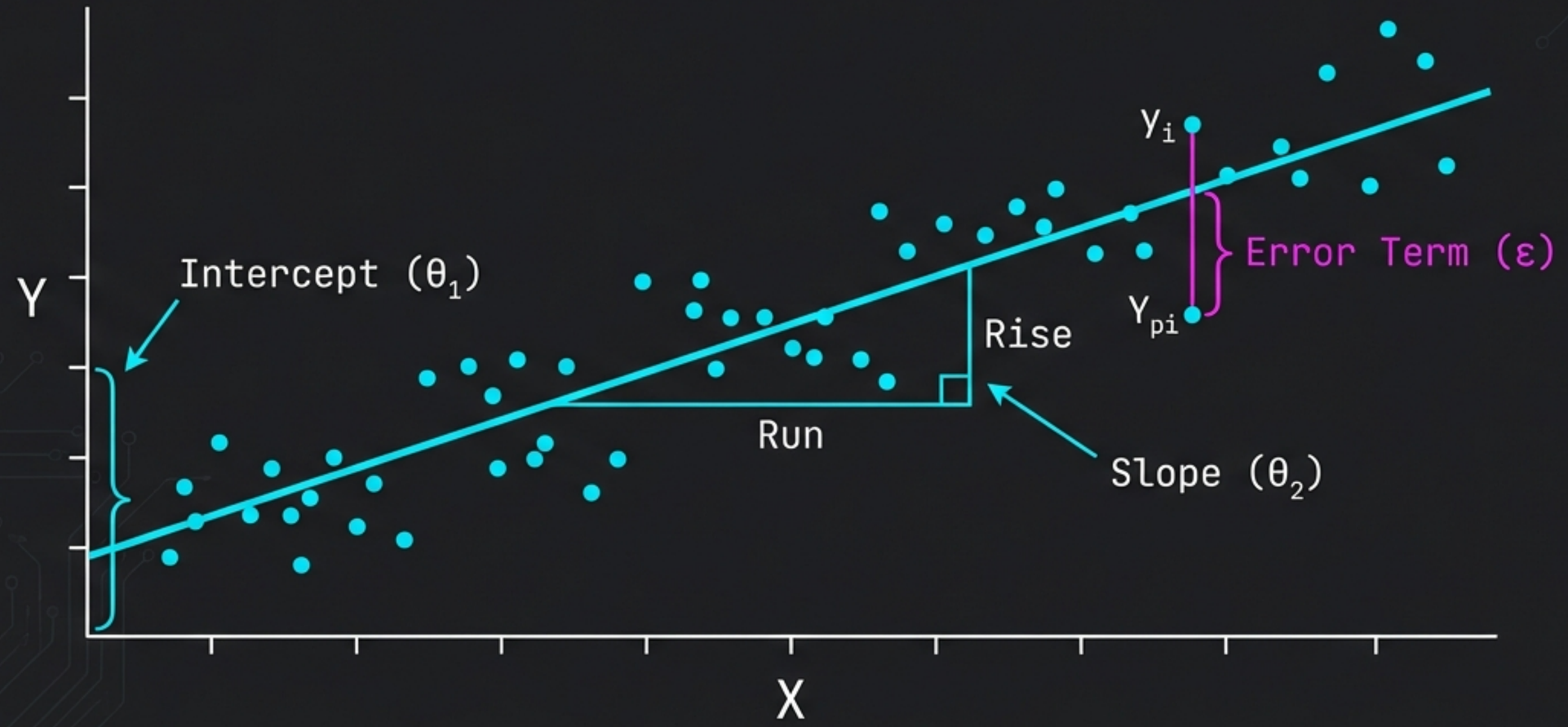

Simple Linear Regression
Bridging Mathematical Theory and **Python** Implementation.
A Supervised Machine Learning Guide.

The Predictive Power of Linearity

Linear regression is a supervised machine learning algorithm that models the relationship between independent variables (Input) and dependent variables (Output) by fitting an optimized linear equation.

Anatomy of the Equation

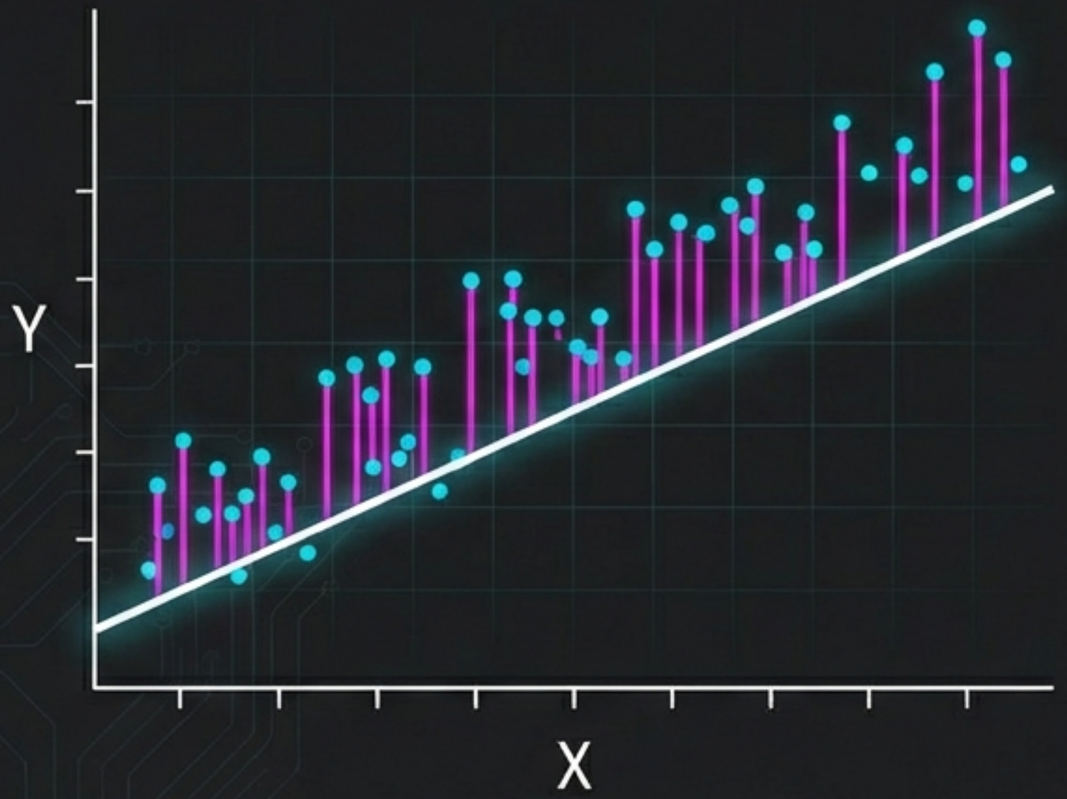
$$Y = \theta_1 + \theta_2 X + \varepsilon$$



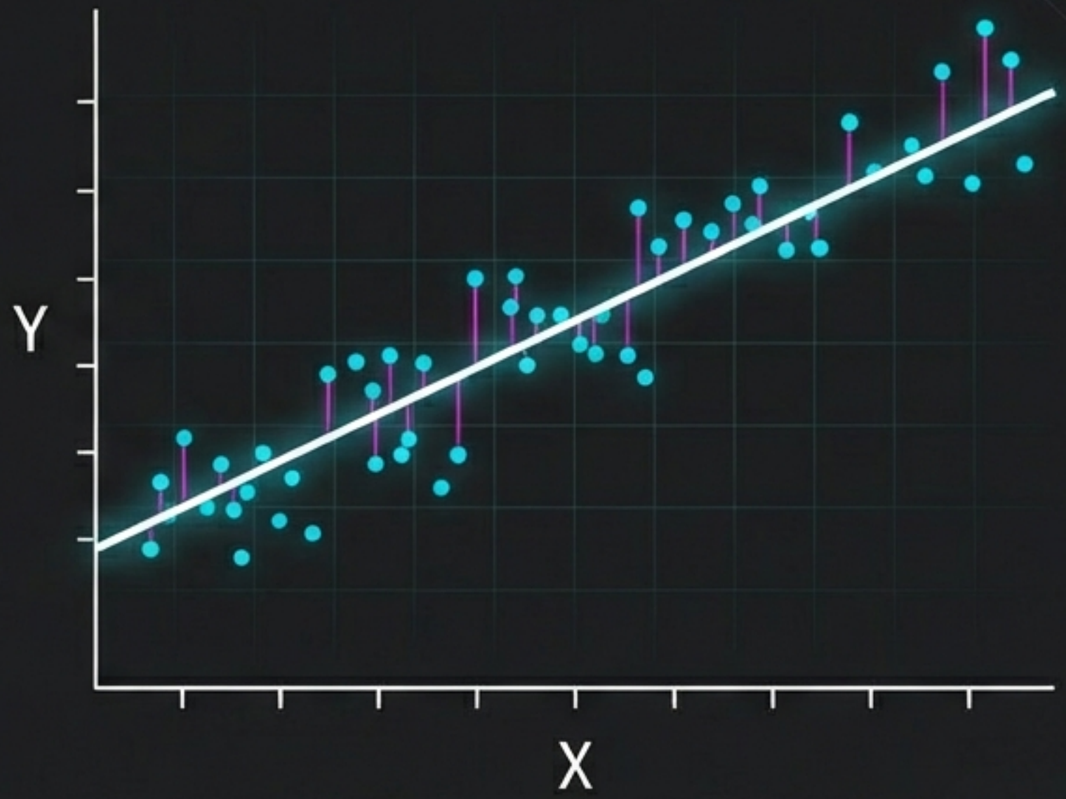
The Quest for the ‘Best Fit’

Goal: Minimize the Cost Function $C(\theta_1, \theta_2)$

High Cost (Bad Fit)



Minimizing Cost (Best Fit)



Quantifying Error: The Metrics

MSE (Mean Squared Error)

$$\frac{\sum(y - \hat{y})^2}{n}$$

Penalizes outliers heavily due to squaring. Differentiable with one global minimum.

MAE (Mean Absolute Error)

$$\frac{\sum|y - \hat{y}|}{n}$$

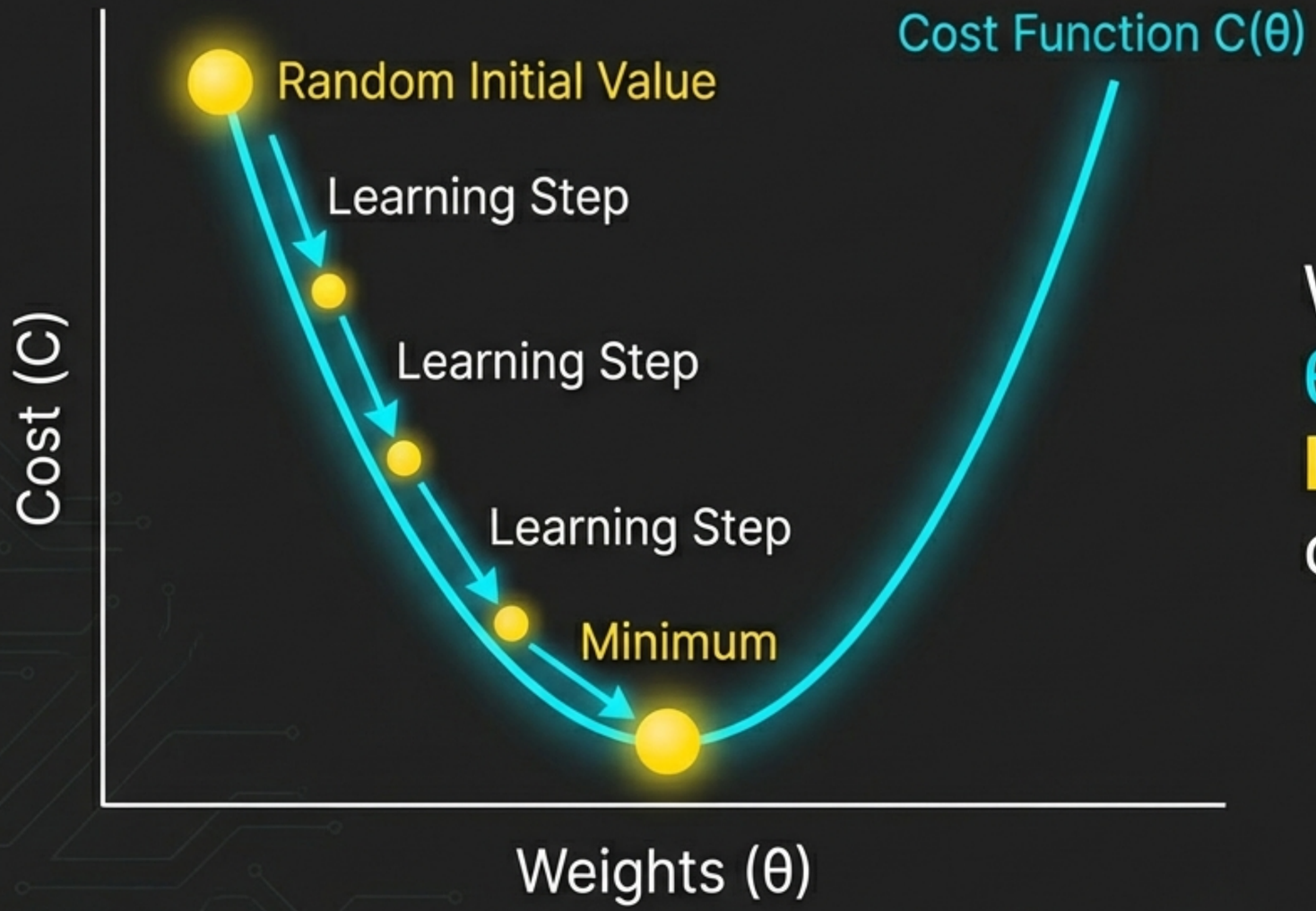
Robust to outliers. Result is in the same units as Y. Not differentiable at 0.

RMSE (Root Mean Squared Error)

$$\sqrt{MSE}$$

Interpretable in original units. Standard metric for regression tasks.

Gradient Descent & Convergence



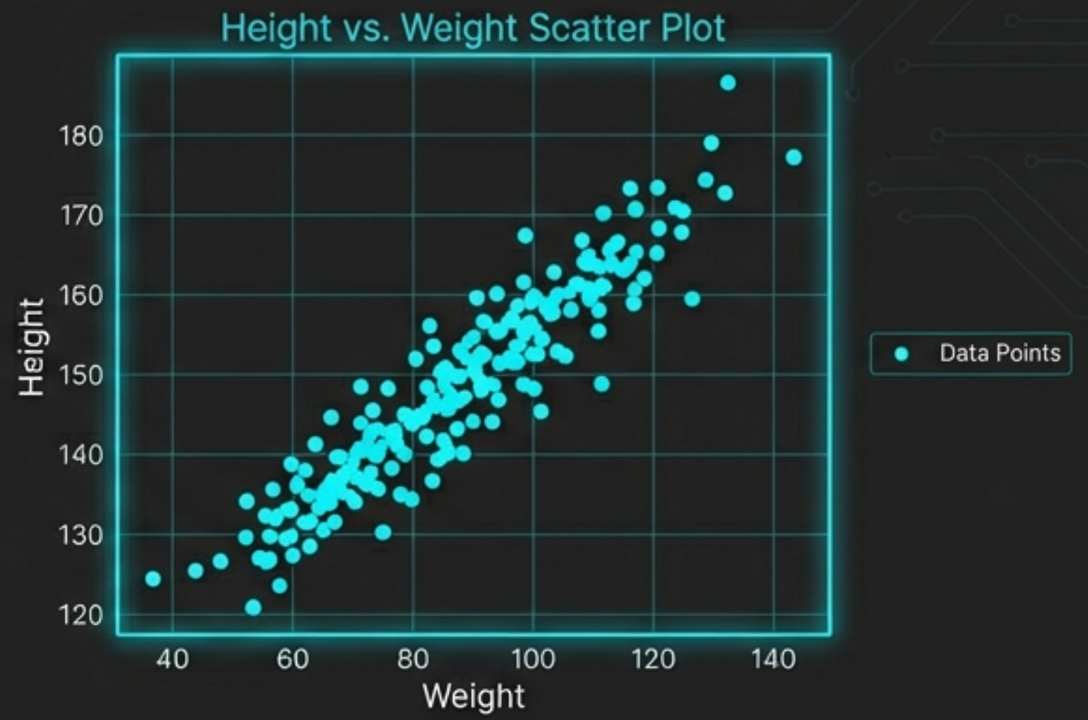
We iteratively adjust θ θ to reduce cost. The **Learning Rate** (α) controls the step size.

Phase 2: Implementation

Loading Data

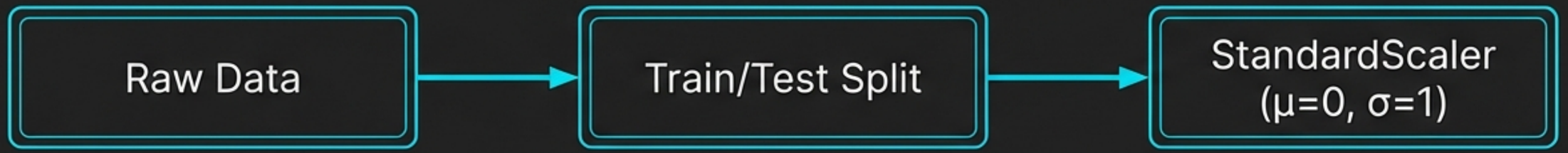
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('height-weight.csv')
plt.scatter(df['Weight'], df['Height'])
```



Visualizing the dataset shows a strong positive correlation between Weight and Height, confirming the expected relationship for analysis.

Standardization & Splitting




```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train) # Fit on Train
X_test = scaler.transform(X_test)       # Transform Test
```

Sample Standardized Output

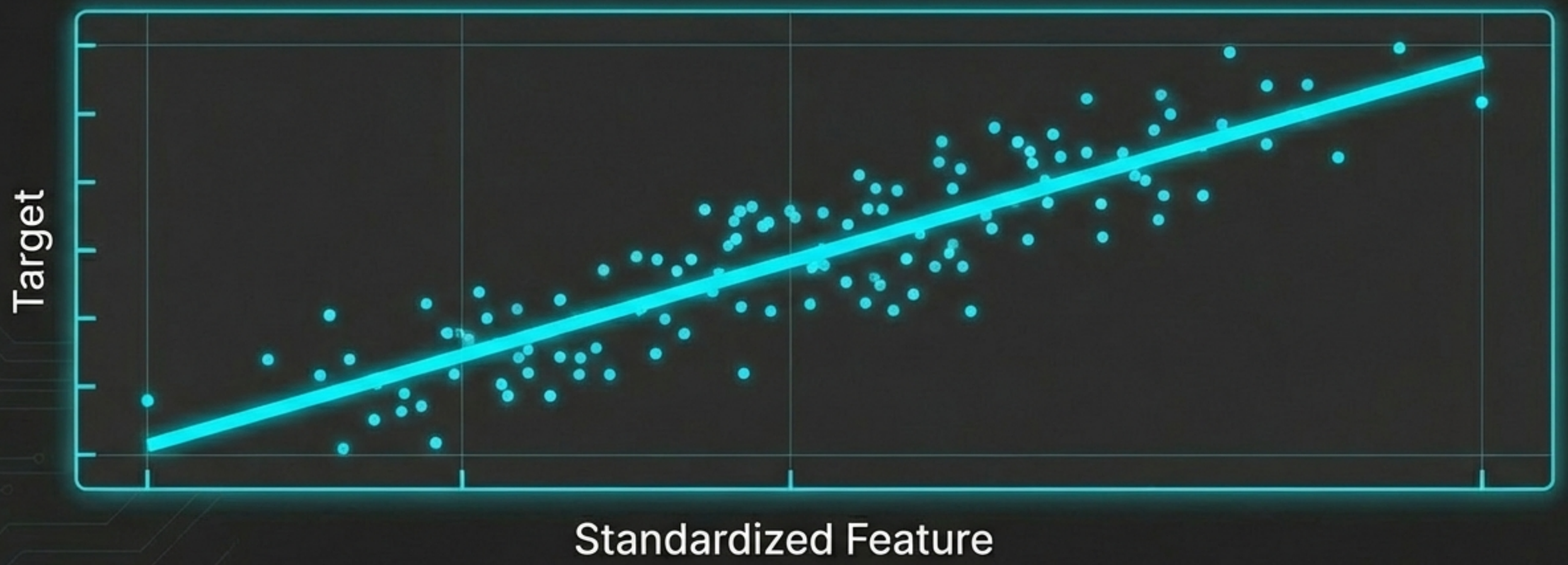
```
[ 0.79,
 -0.31,
  1.14,
 -0.56,
  0.22 ]
```

Training the Model

```
from sklearn.linear_model import LinearRegression
regression = LinearRegression()
regression.fit(X_train, y_train)
```

Slope / Coefficient (θ_2)	Intercept (θ_1)
17.85	163.31

Visualizing the Best Fit




```
plt.plot(X_train, regression.predict(X_train))
```

Performance Evaluation

0.917

R^2 Score (Accuracy: 91.8%)

MSE: 33.87

MAE: 4.05

Adjusted R^2

Adjusted R^2 penalizes the addition of useless features.

$$1 - \frac{(1-R^2)(N-1)}{(N-P-1)}$$

The Prediction Pitfall

Common Error

```
regression.predict([[70]])
```

Output: 1413.19
(Impossible Value!)

Always scale new inputs using the same scaler!

Correct Approach

```
regression.predict(scaler.
    .transform([[70]]))
```

Output: 163.31 (Correct Prediction)

The OLS Alternative (Statsmodels)

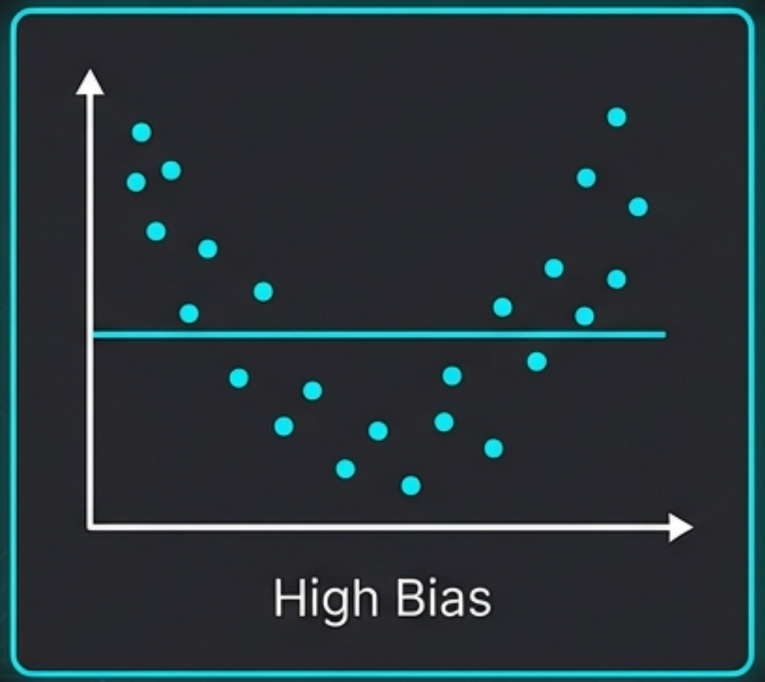
Model: OLS	**Method:** Least Squares
R-squared: 0.012	**Prob (F-statistic):** 0.443

coef	std err	P>|t|
17.8555	23.104	0.443

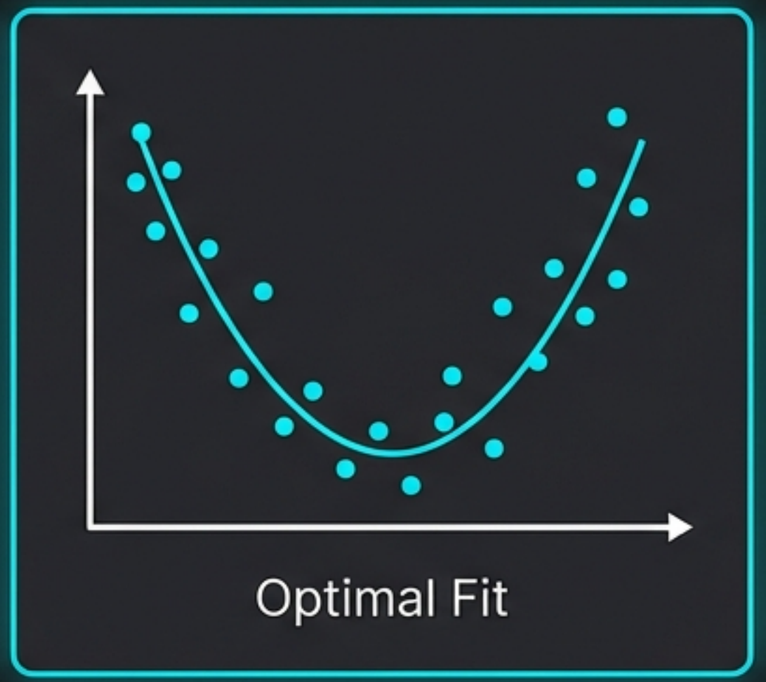
OLS provides **p-values** and **detailed statistical tests** that sklearn hides by default.

Bias vs. Variance Trade-off

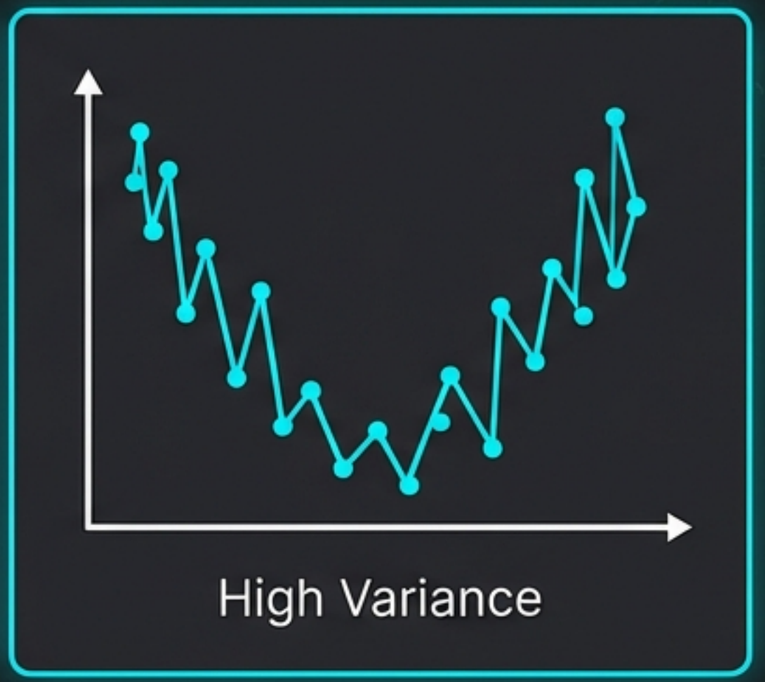
High Bias (Underfitting)



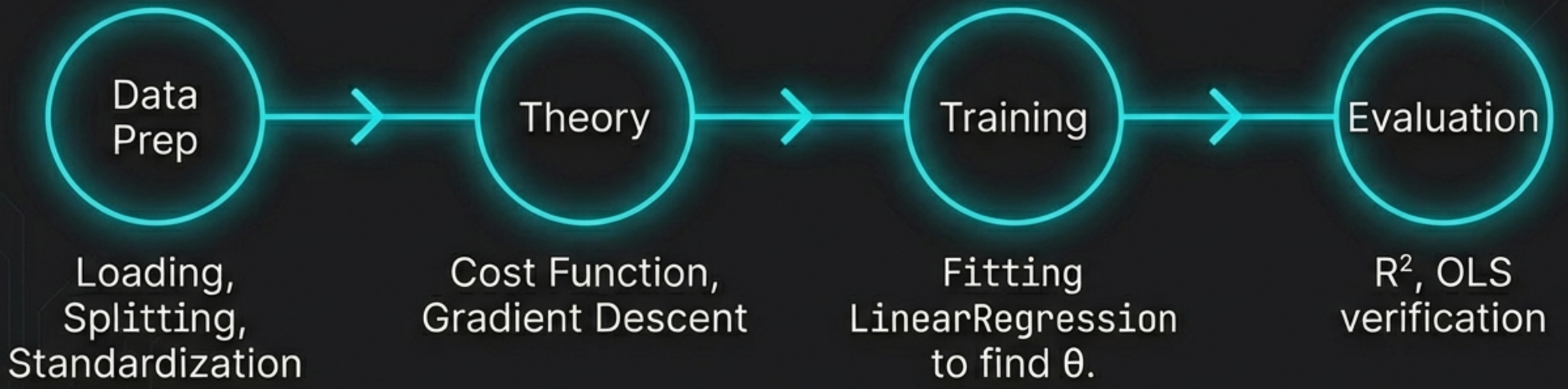
Optimal Fit



High Variance (Overfitting)



Summary & Roadmap



Linear Regression forms the mathematical foundation of supervised learning.