Reducing Complexity in Data

UNDERSTANDING THE NEED FOR DIMENSIONALITY REDUCTION

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Need for dimensionality reduction in building ML models

Bias-variance trade-off

Overfitting and the curse of dimensionality

Drawbacks of excessively complex models

Choosing dimensionality reduction techniques based on use case

Prerequisites and Course Outline

Prerequisites

Working with Python and Python libraries

Basic understanding of machine learning algorithms

Prerequisites

Understanding Machine Learning by David Chappell

Building Machine Learning Models in Python with scikit-learn by Janani Ravi

Understanding Machine Learning with Python by Jerry Kurata

Course Outline

Need for dimensionality reduction

Statistical techniques for feature selection

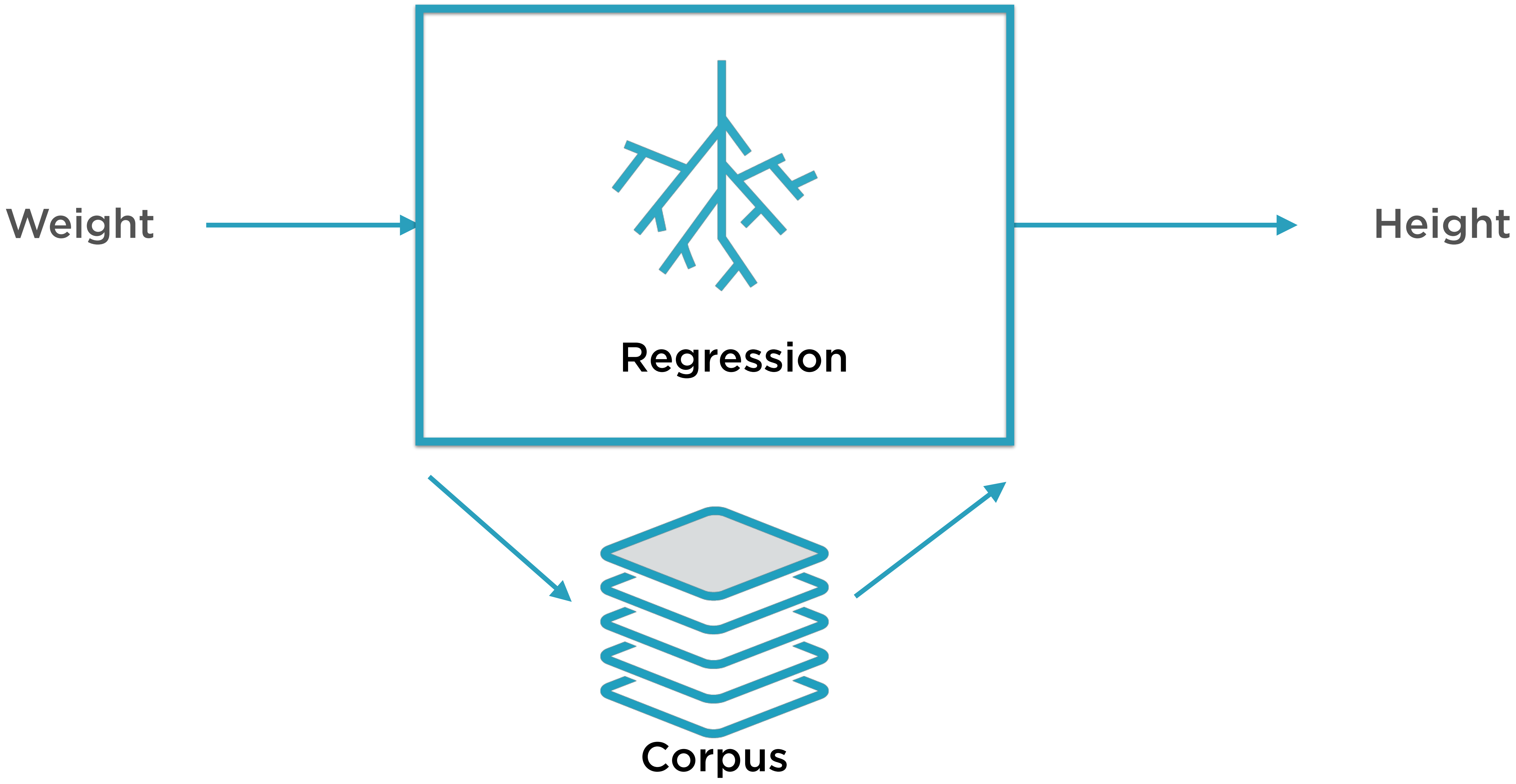
Reducing complexity in linear data

Reducing complexity in non-linear data

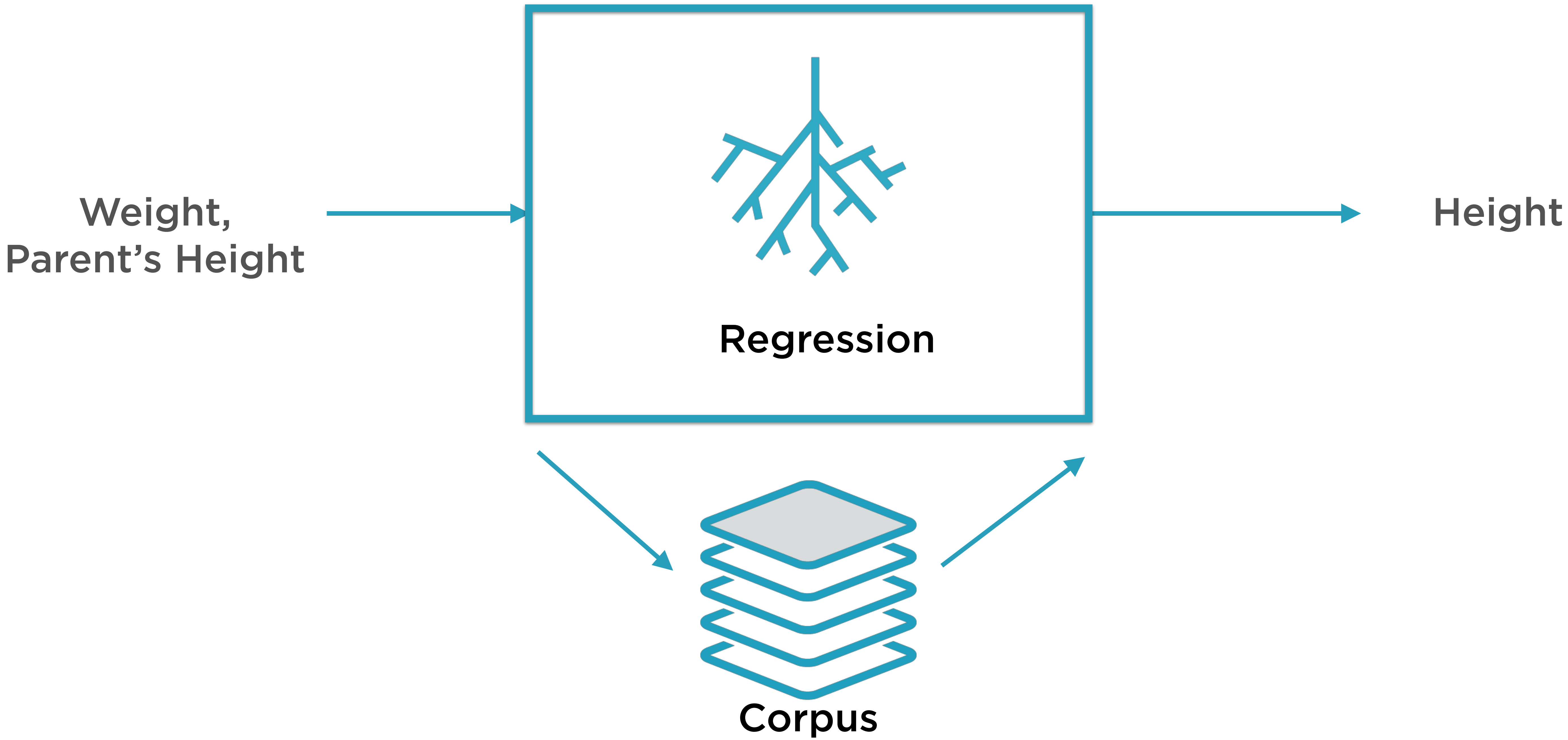
Clustering and autoencoding

The Curse of Dimensionality

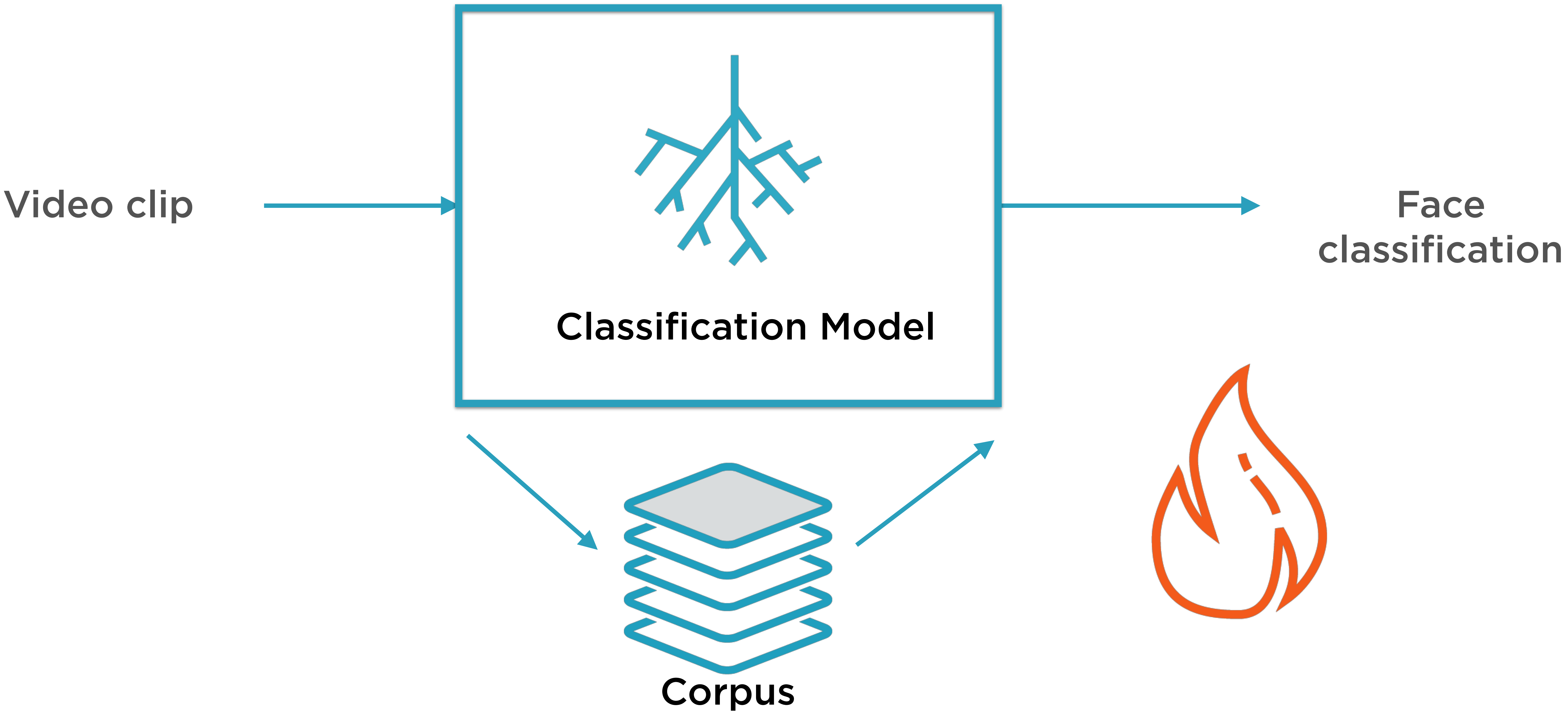
One X Variable



Two X Variables



Dimensionality Explosion



Curse of Dimensionality: As number of **x** variables grows, several problems arise

Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Problems in Visualization

Exploratory Data Analysis (EDA) is an essential precursor to model building

Essential for

- Identifying outliers
- Detecting anomalies
- Choosing functional form of relationships

Problems in Visualization

Two dimensional visualizations are powerful aids in EDA

Even three-dimensional data is hard to meaningfully visualize

Higher dimensional data is often imperfectly explored prior to ML

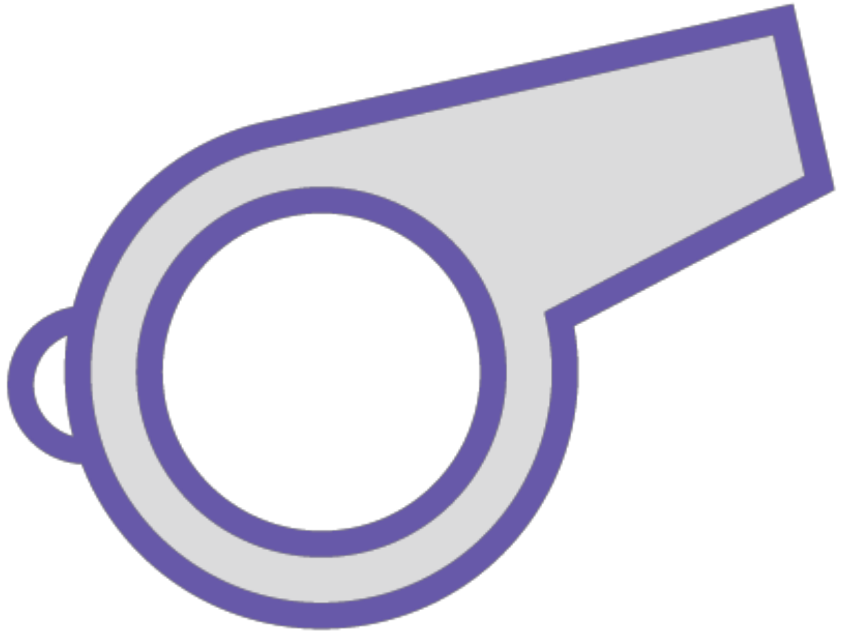
Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Problems in Training

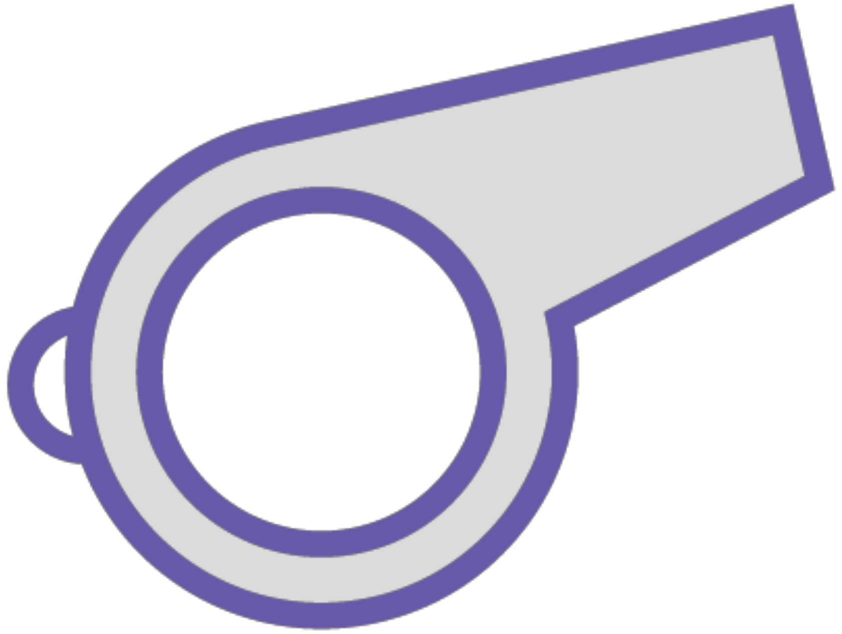


Training is the process of finding best model parameters

Complex models have thousands of parameter values

Training for too little time leads to bad models

Problems in Training



Number of parameters to be found grows rapidly with dimensionality

Extremely time-consuming

For on-cloud training, also extremely expensive

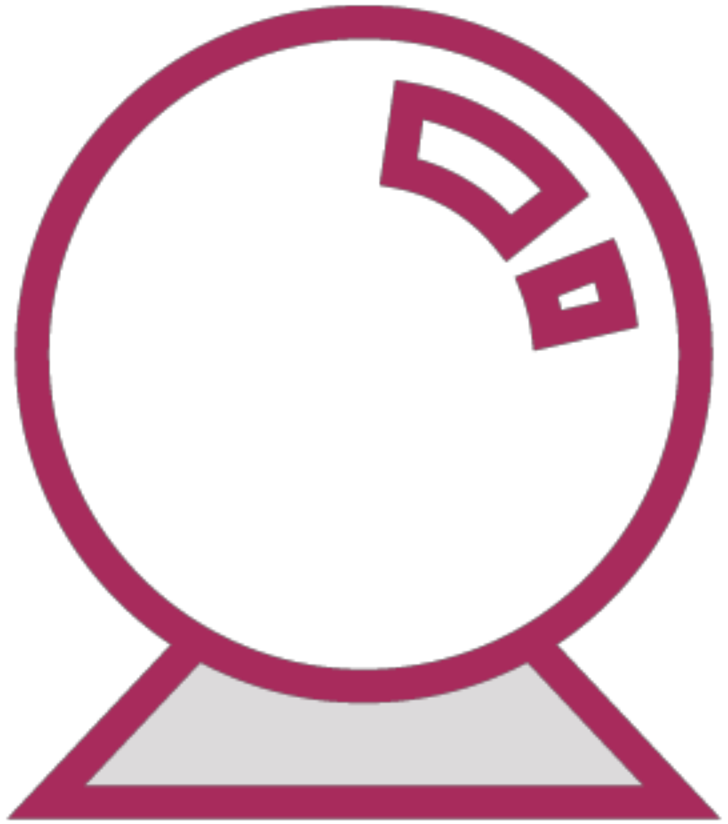
Curse of Dimensionality

Problems in Visualization

Problems in Training

Problems in Prediction

Problems in Prediction



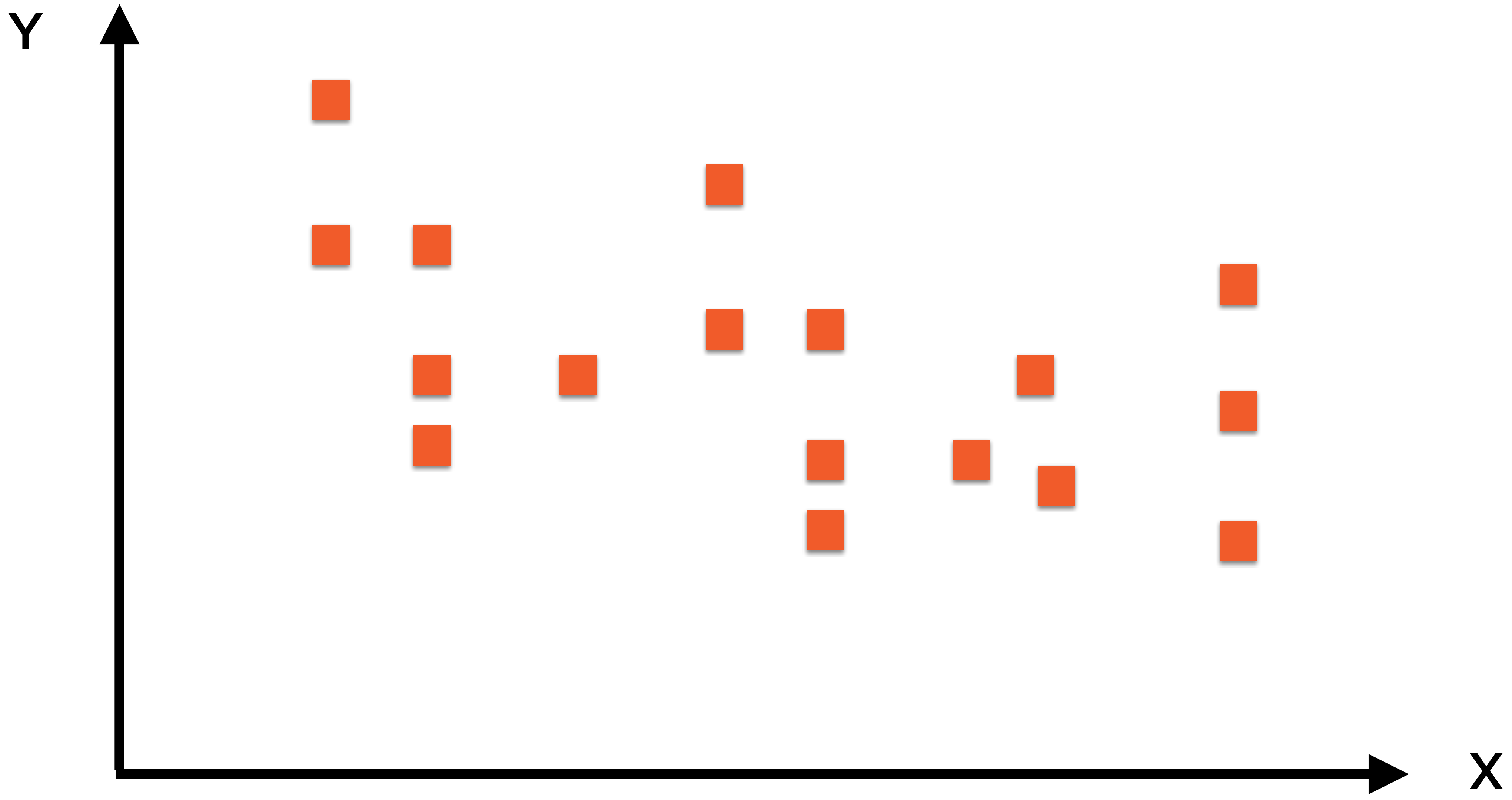
Prediction involves finding training instances similar to test instance

As dimensionality grows, size of search space explodes

Higher the number of X variables, higher the risk of overfitting

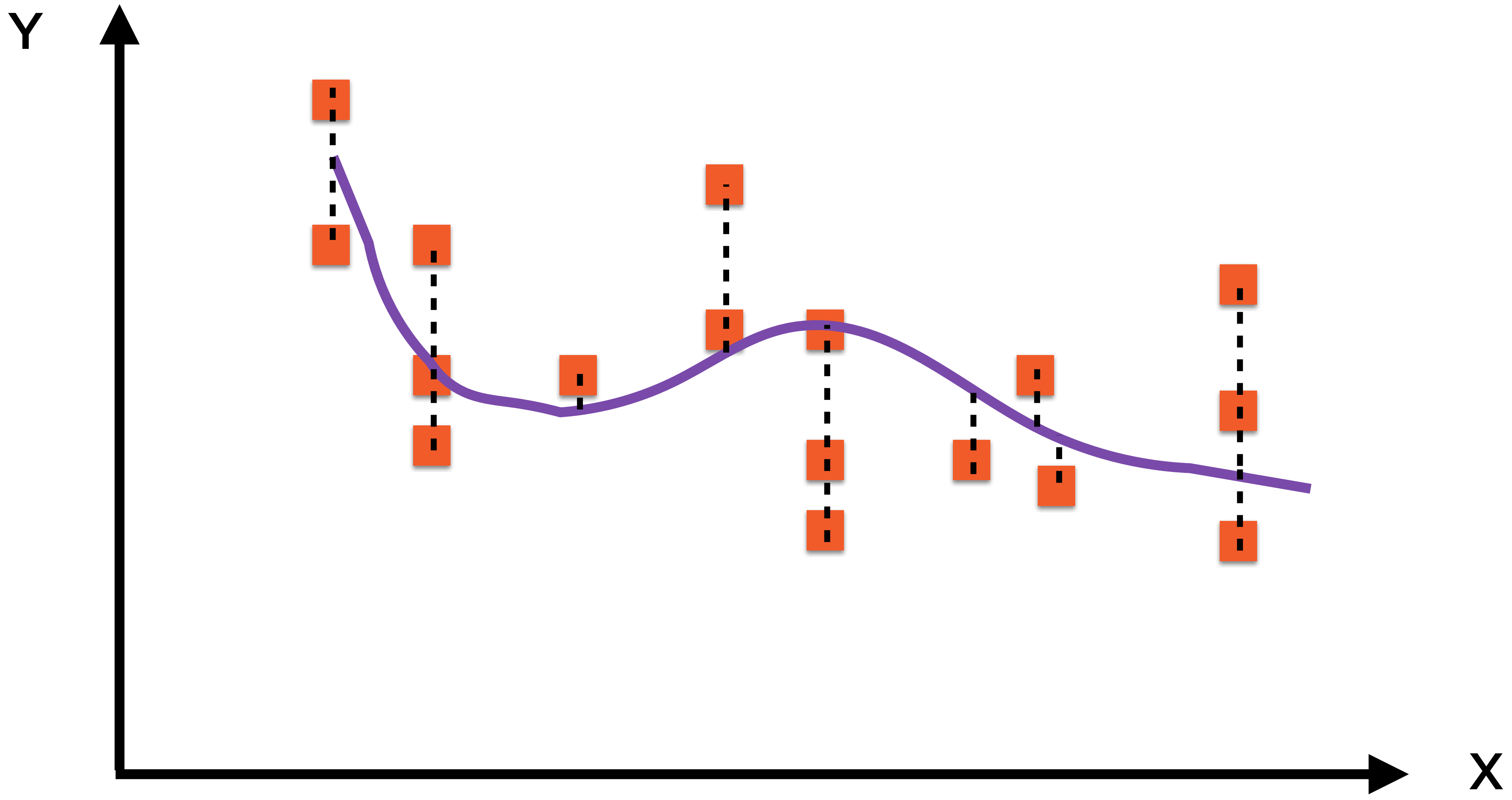
Overfitting and the Bias-variance Trade-off

Connecting the Dots



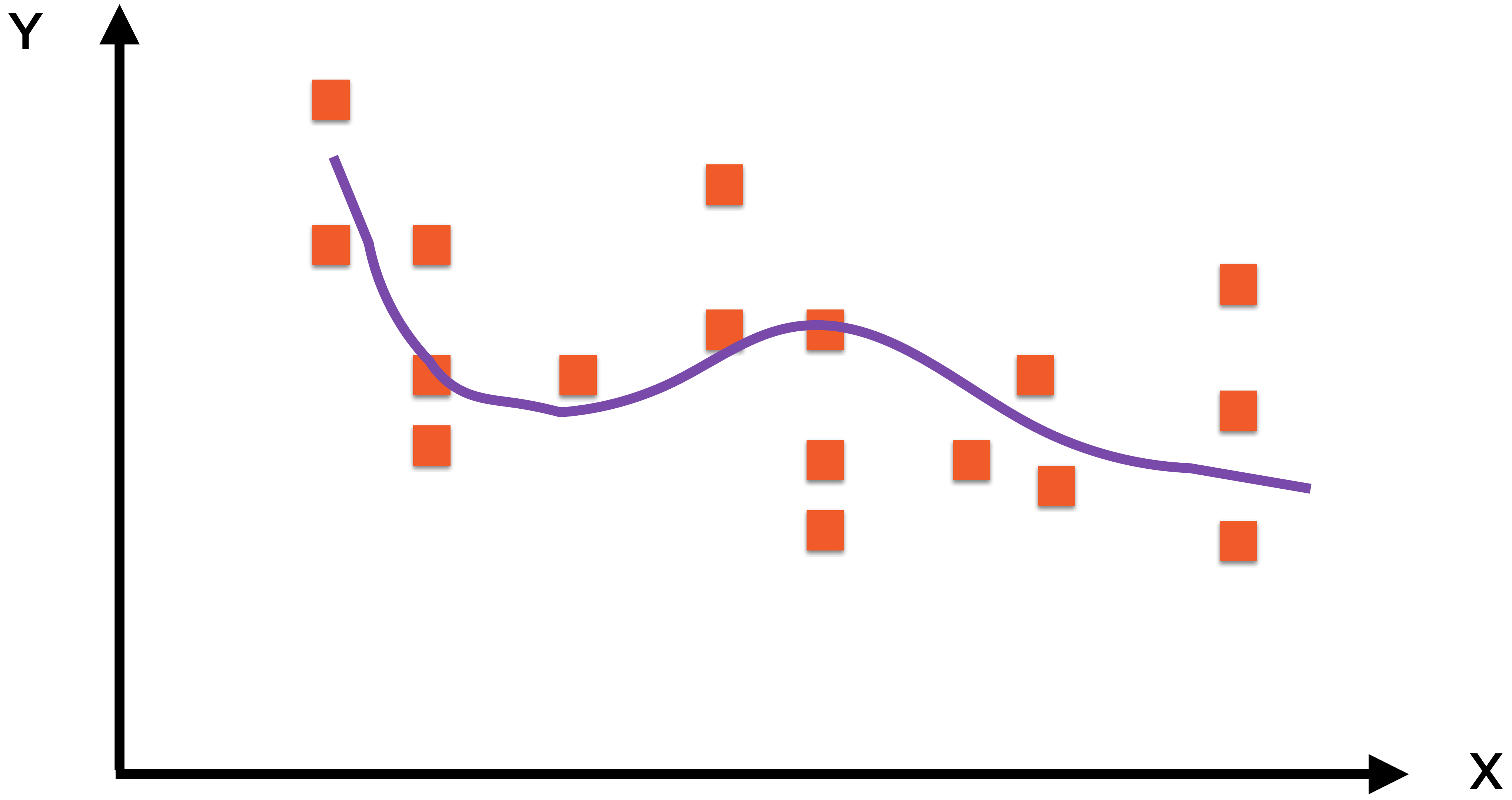
Challenge: Fit the "best" curve through these points

Good Fit?



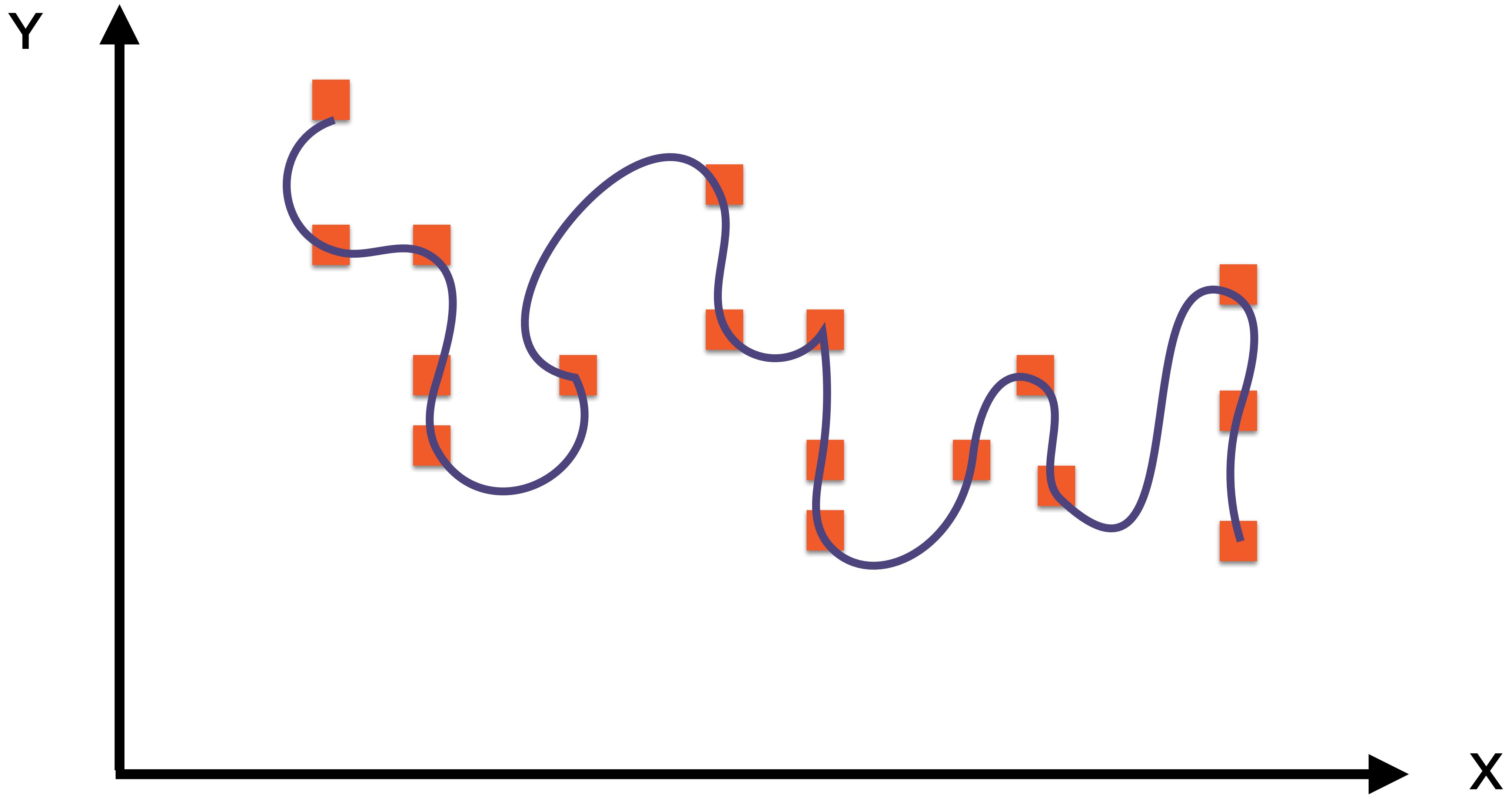
A curve has a “good fit” if the distances of points from the curve are small

Connecting the Dots



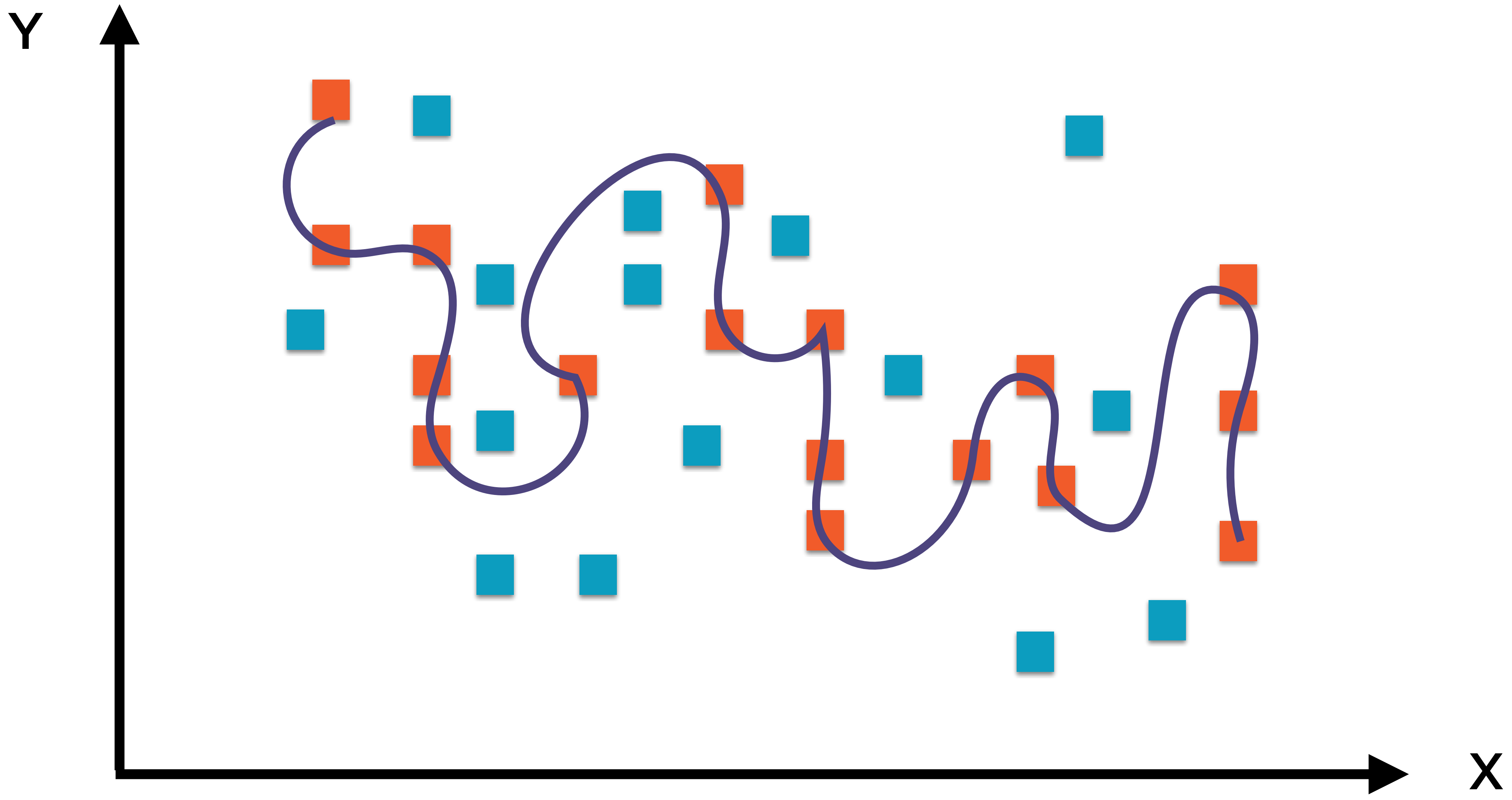
We could draw a pretty complex curve

Connecting the Dots



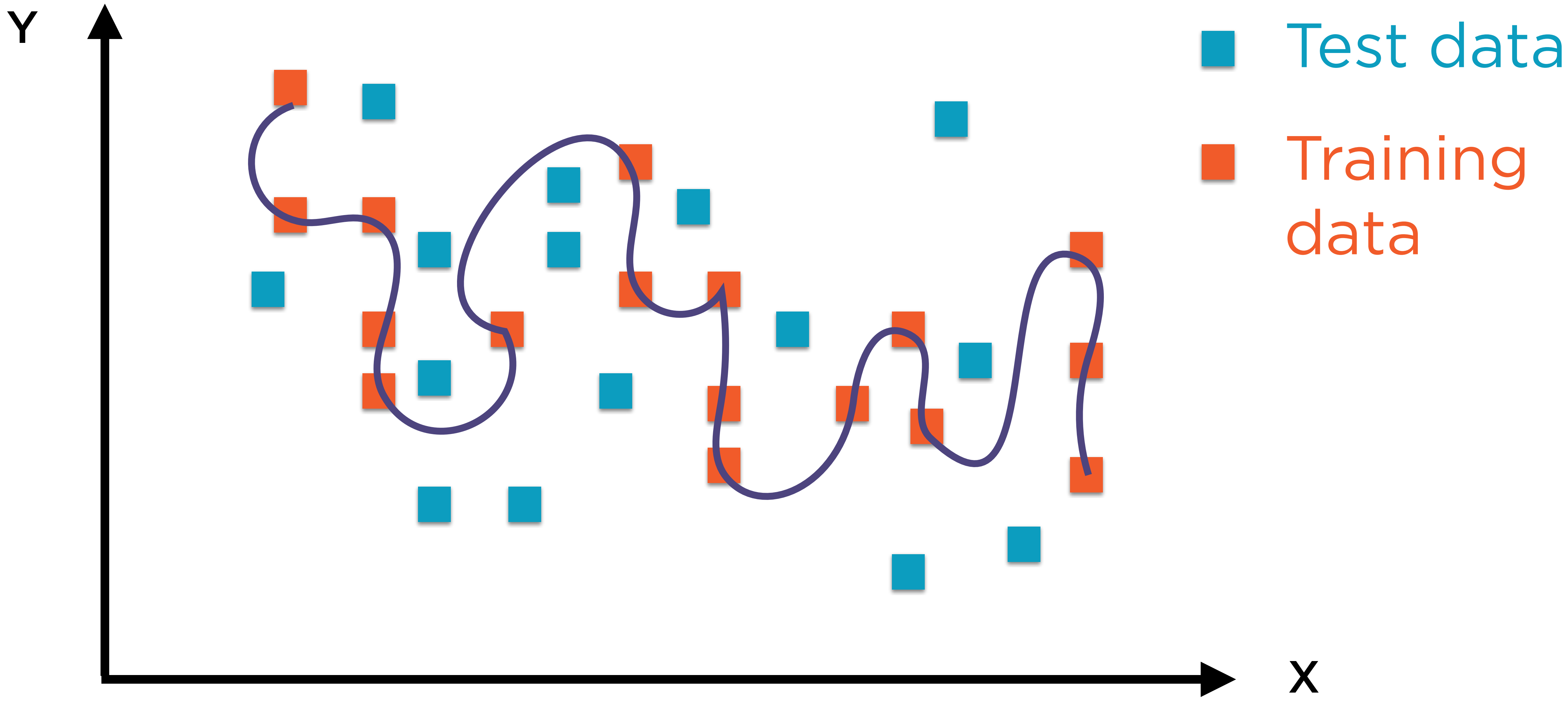
We can even make it pass through every single point

Connecting the Dots



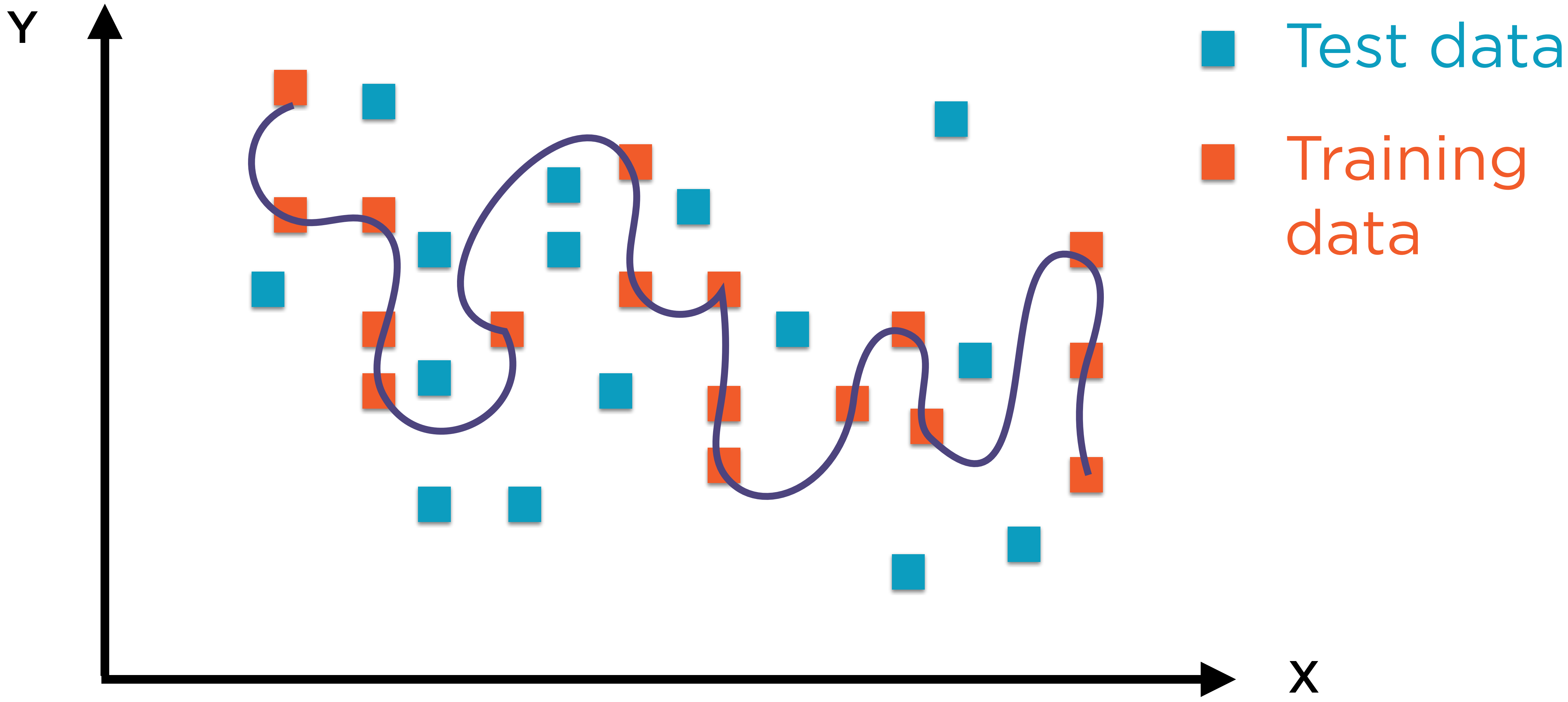
But given a new set of points, this curve might perform quite poorly

Connecting the Dots



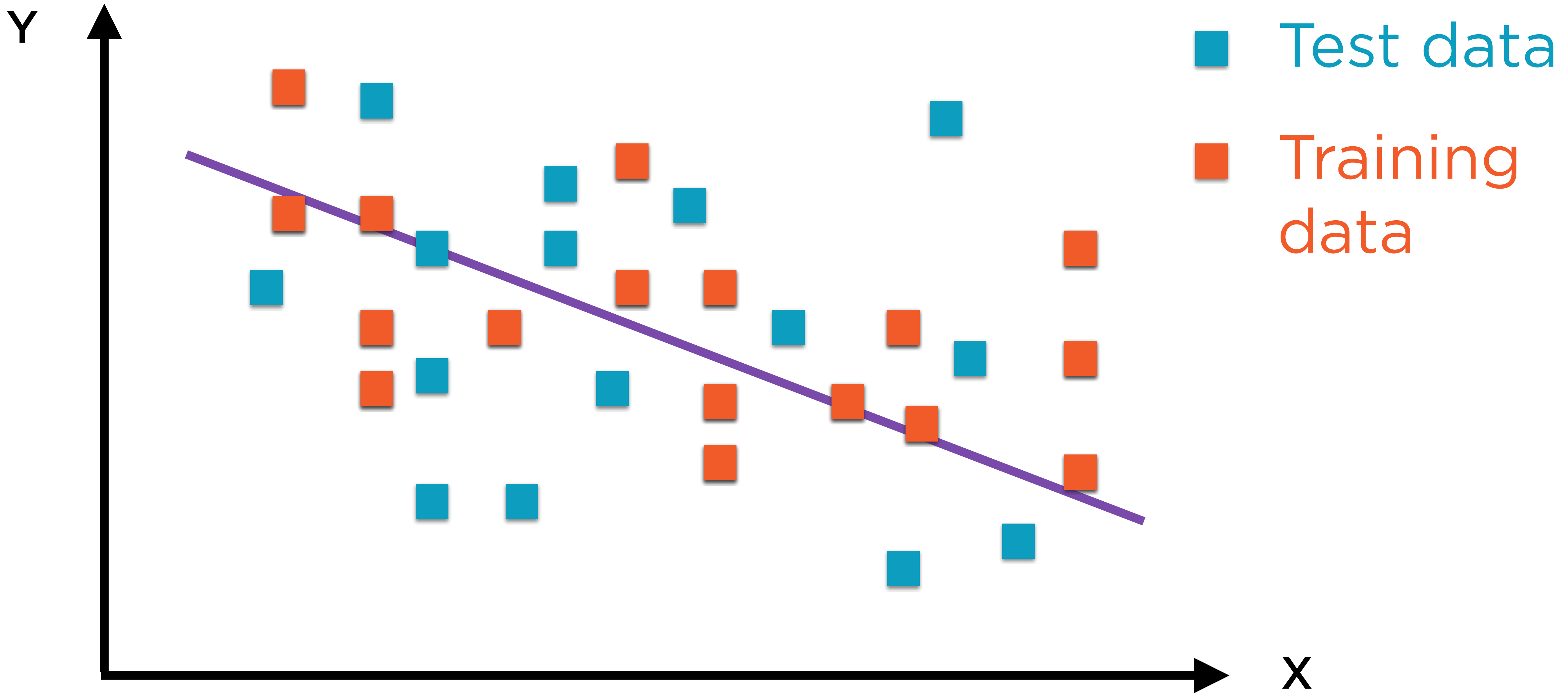
The original points were "training data", the new points are "test data"

Overfitting



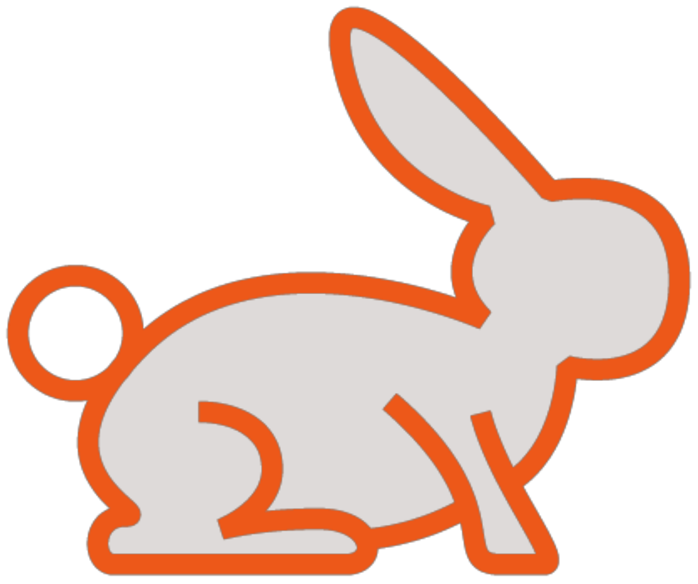
Great performance in training, poor performance in real usage

Connecting the Dots



A simple straight line performs worse in training, but better with test data

Overfitting

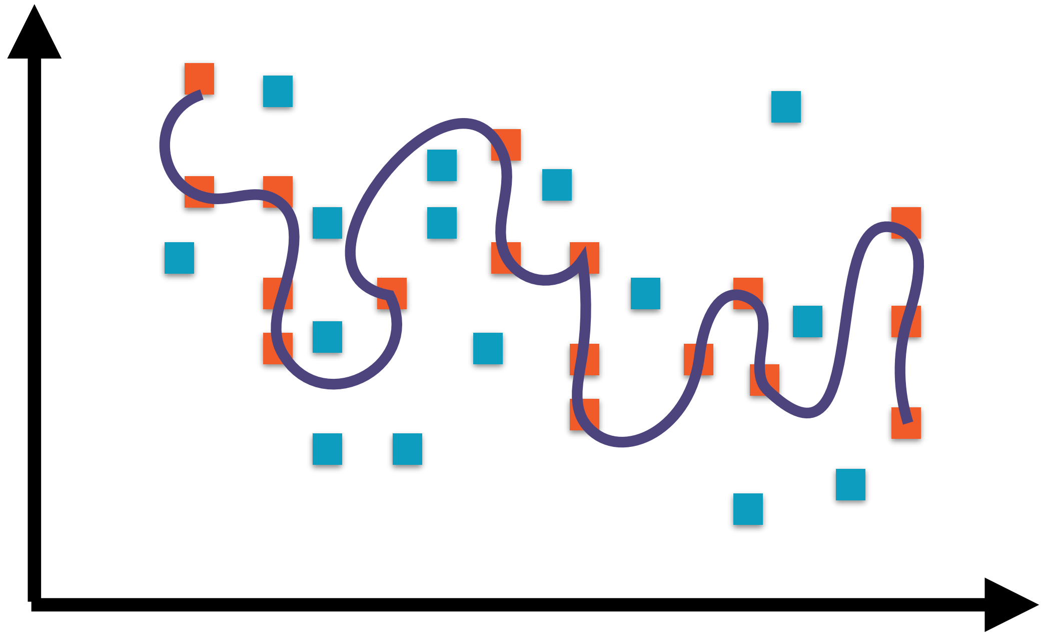


Low Training Error

Model does very well in training…

High Test Error

…but poorly with real data



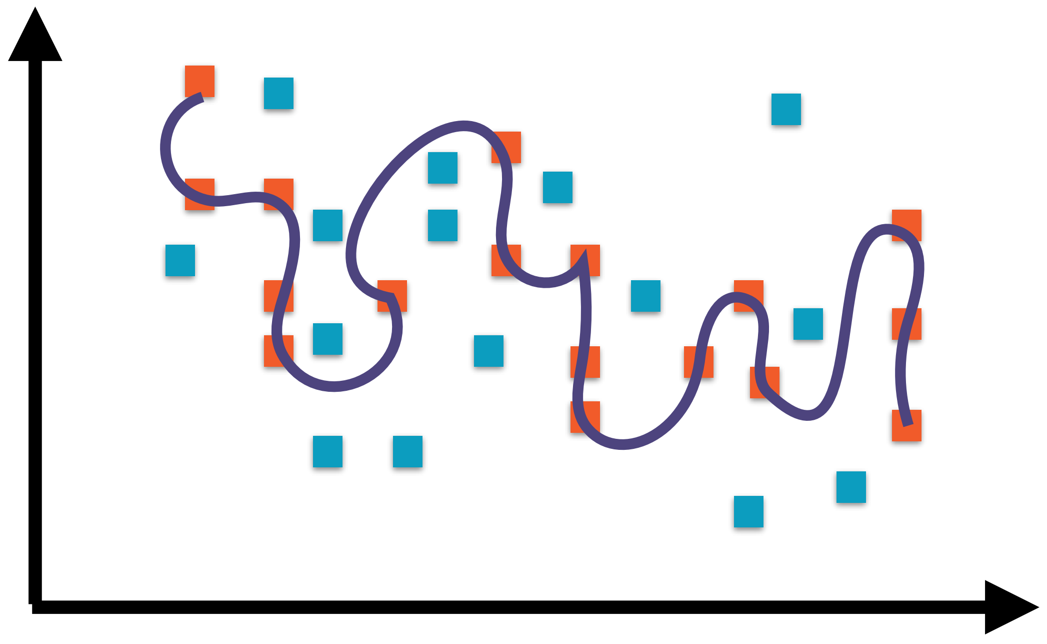
Cause of Overfitting

Sub-optimal choice in the bias-variance trade-off

An overfitted model has:

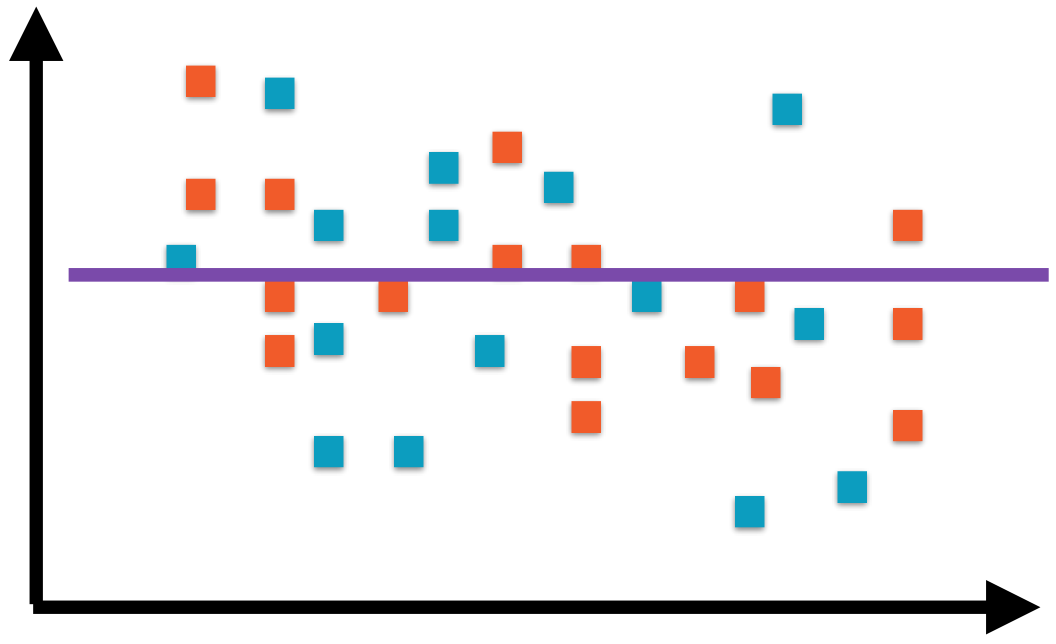
- High variance error
- Low bias error

Bias



Low bias

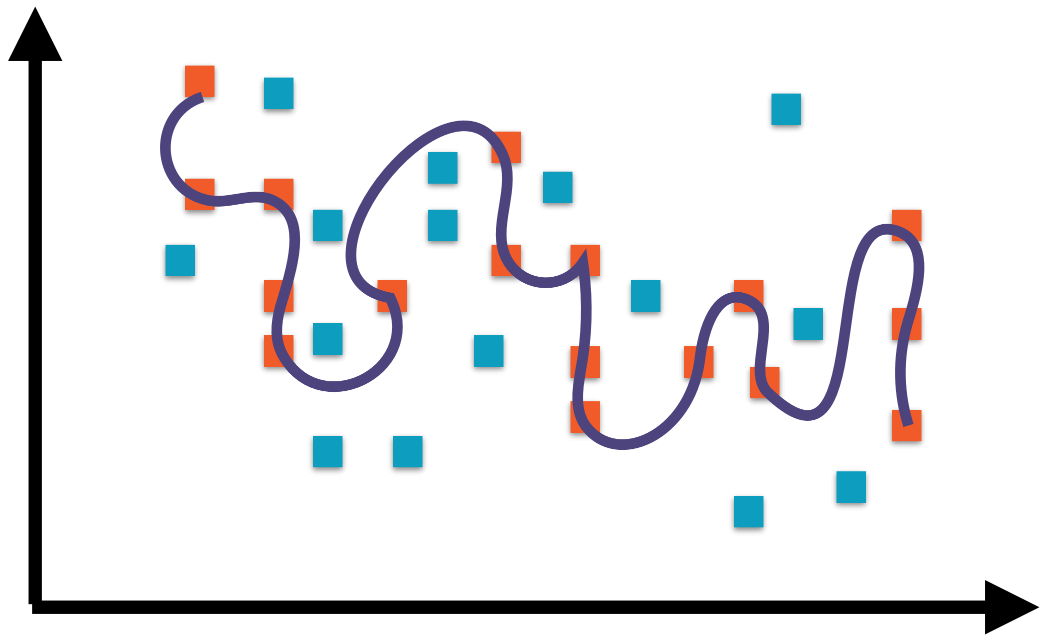
Few assumptions about the underlying data



High bias

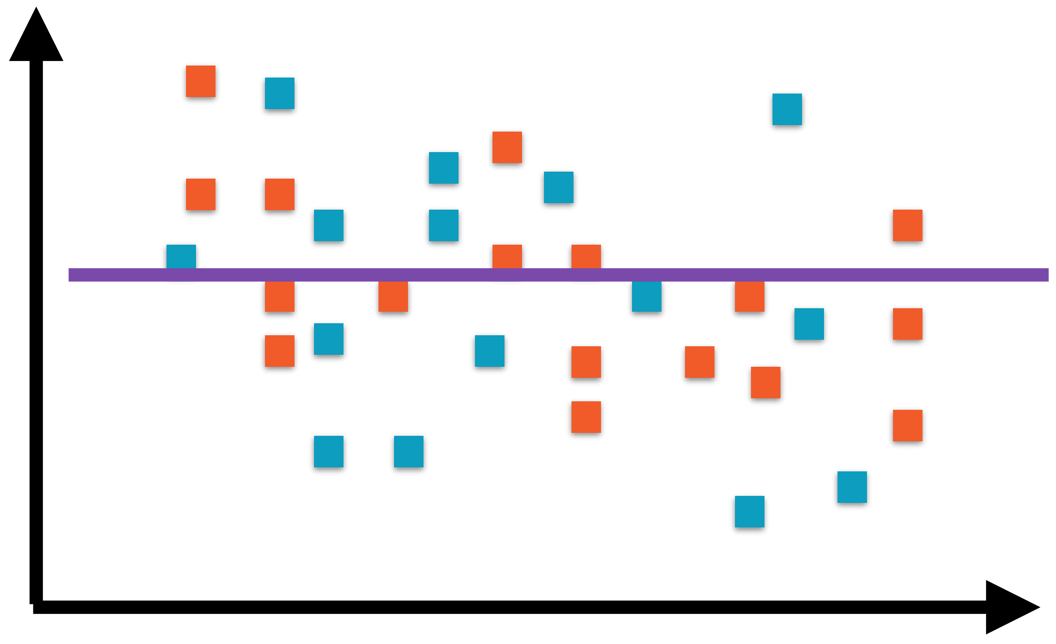
More assumptions about the underlying data

Bias



Model too complex

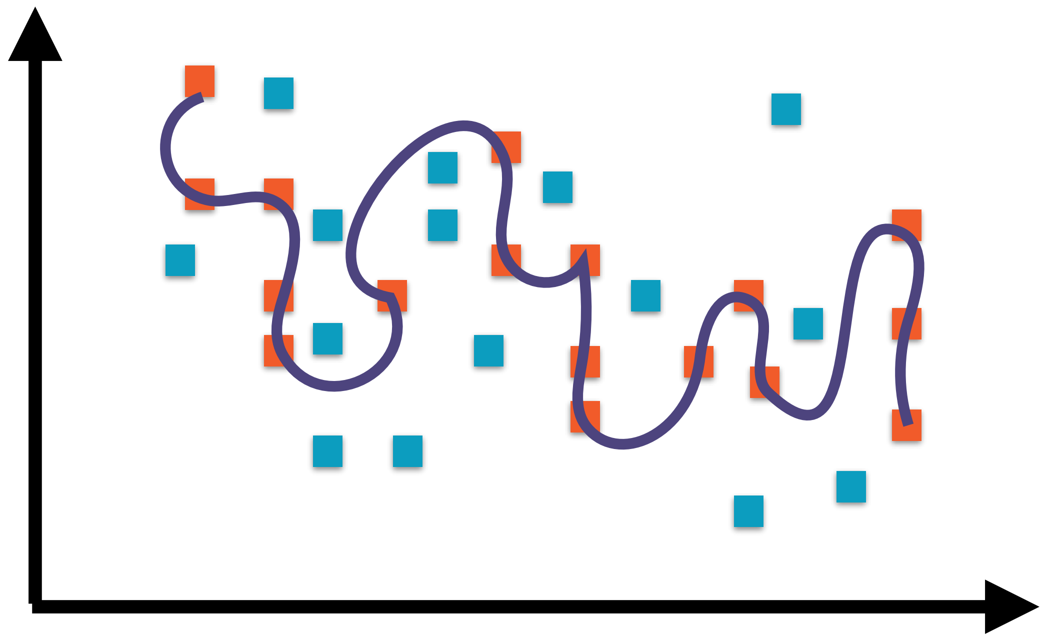
Training data all-important, model parameter counts for little



Model too simple

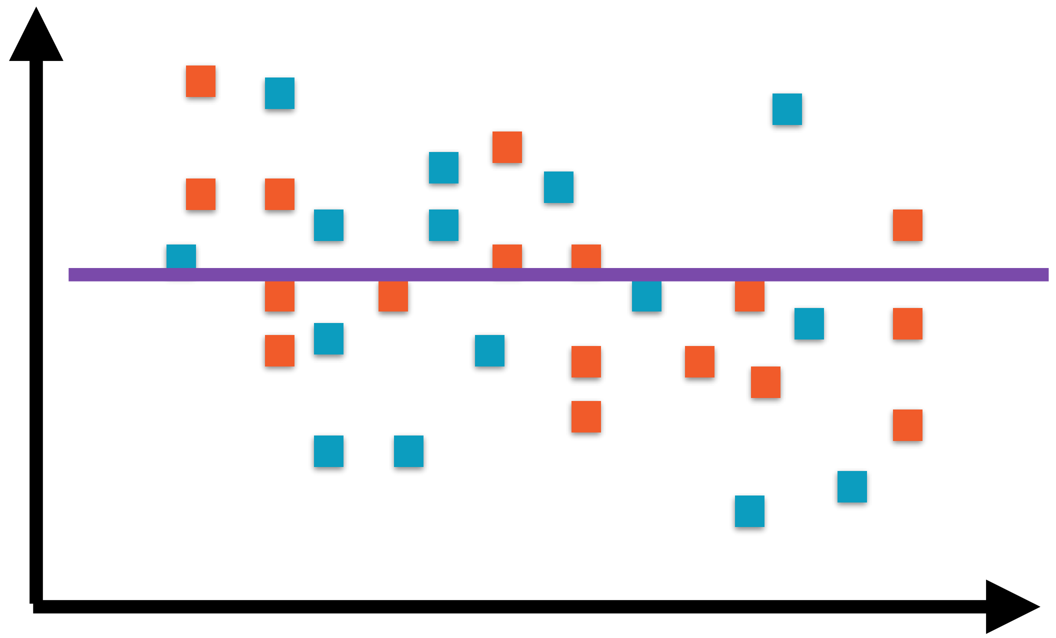
Model parameter all-important, training data counts for little

Variance



High variance

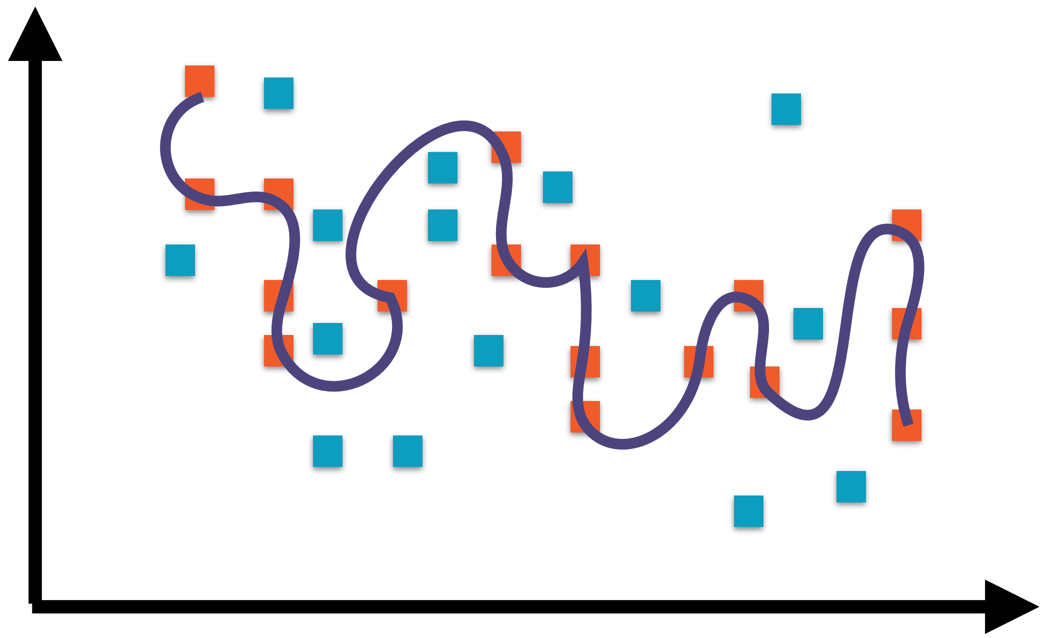
The model changes significantly when training data changes



Low variance

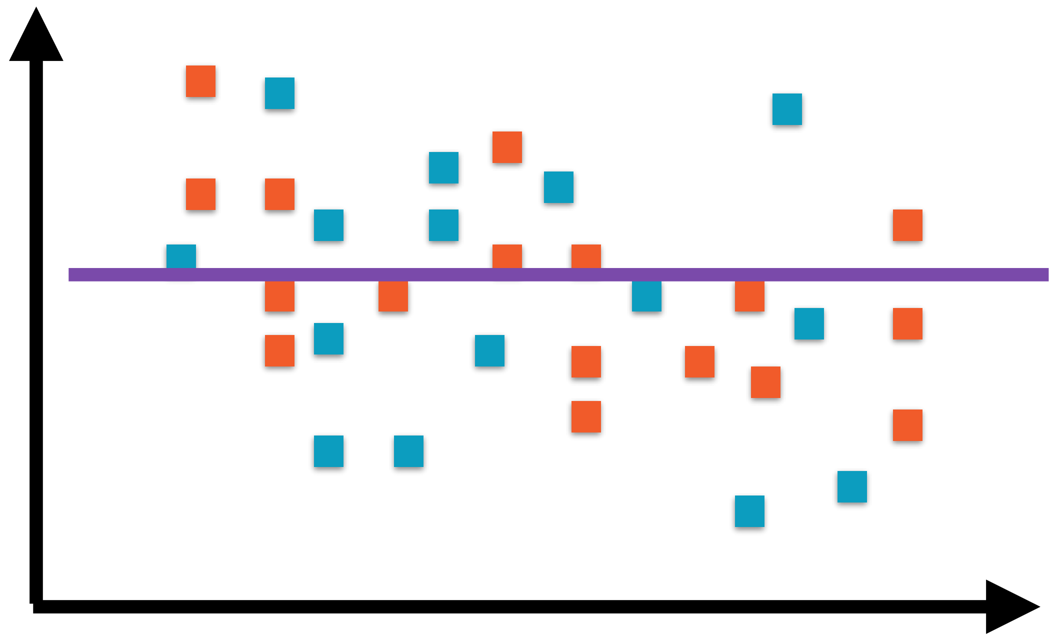
The model doesn't change much when the training data changes

Variance



Model too complex

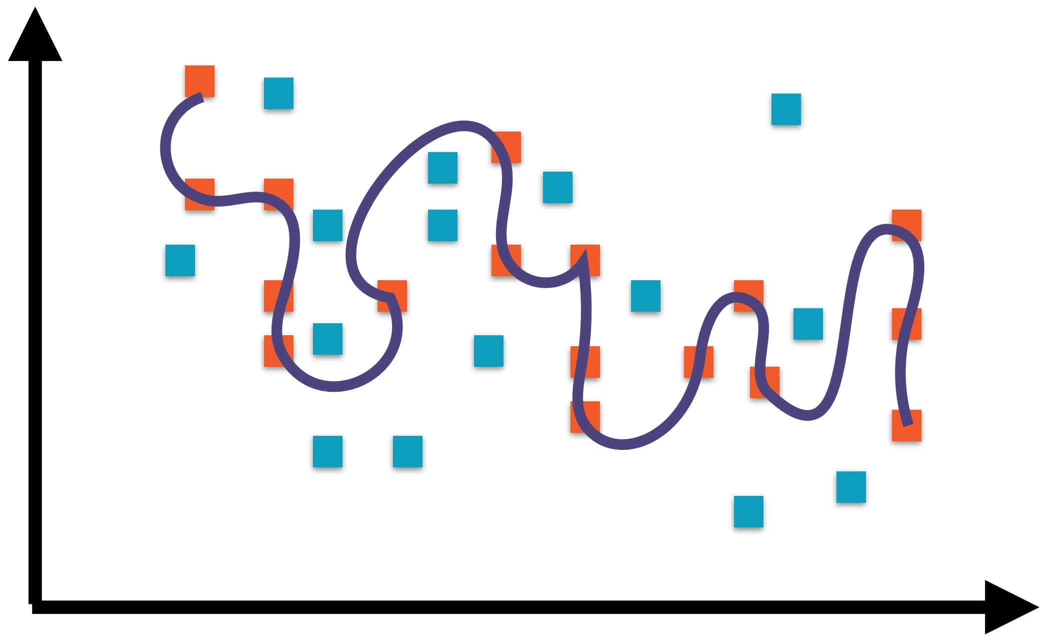
Model varies too much with changing training data



Model too simple

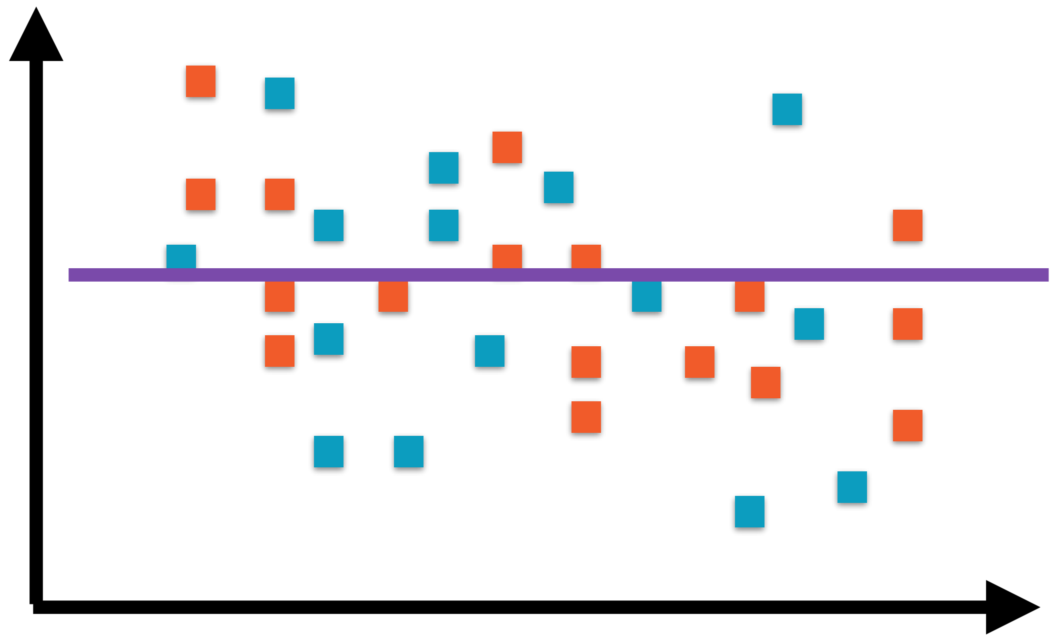
Model not very sensitive to training data

Bias-variance Trade-off



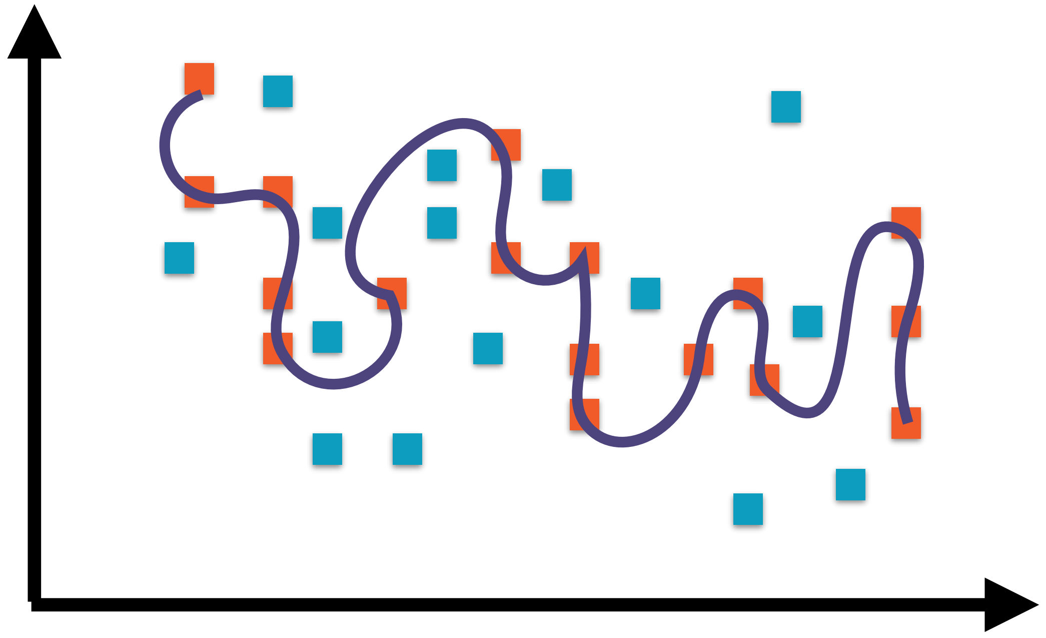
Model too complex

High variance error



Model too simple

High bias error



Bias-variance Trade-off

High-bias algorithms: simple parameters

- Regression

High-variance algorithms: complex parameters

- Decision trees
- Dense neural networks

Preventing Overfitting

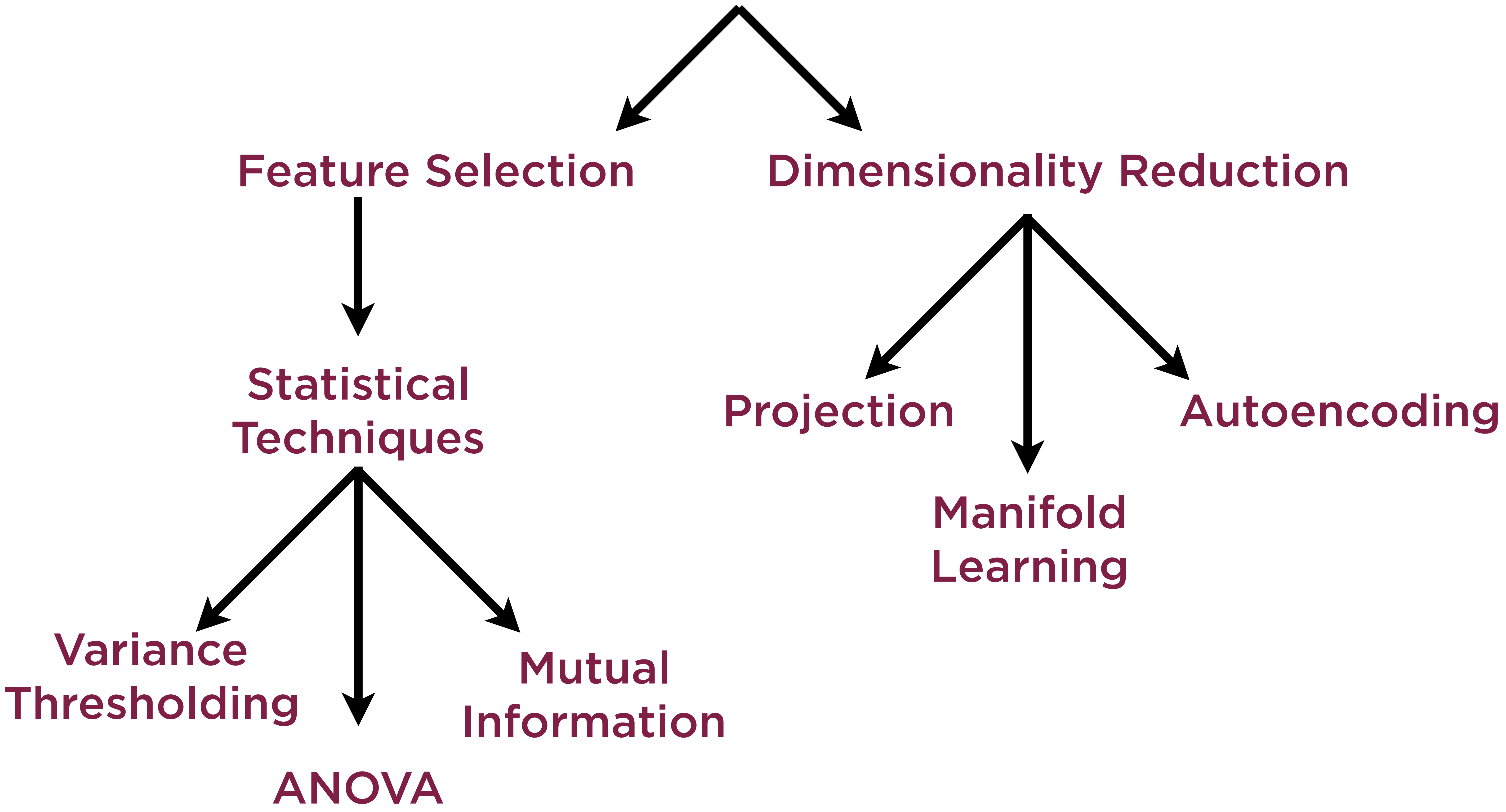
Regularization - Penalize complex models

Cross-validation - Distinct training and validation phases

Dimensionality Reduction - Reduce complexity of data

Solutions for Reducing Complexity

Reducing Complexity



Reducing Complexity

Feature Selection

Choose a subset of original X variables

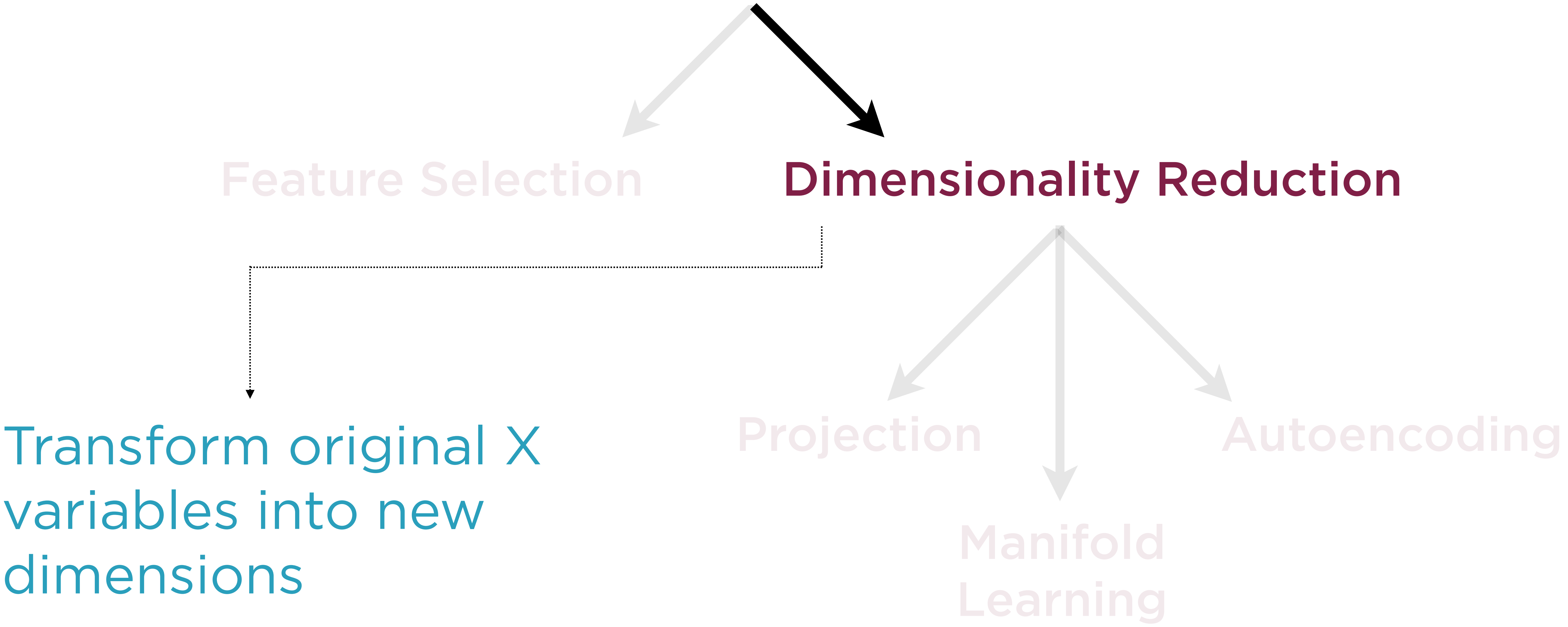
Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Reducing Complexity



Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Find new, better axes and re-orient data

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

e.g. PCA, Factor Analysis, LDA, QDA

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Works best with linear data (can use kernel trick to extend to non-linear data)

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Unroll the data so that twists and turns are smoothened out

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Autoencoding

Manifold Learning

Works best when data lies along a rolled-up surface such as a Swiss Roll or S-curve

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Autoencoding

Manifold Learning

e.g. MDS, Isomap, LLE, Kernel PCA

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Build neural networks to simplify the data

Reducing Complexity

Feature Selection

Dimensionality Reduction

Projection

Manifold Learning

Autoencoding

Extract efficient representations of complex data

Choosing Feature Selection

Use Case	Possible Solution
Many X-variables	
Most of which contain little information	
Some of which are very meaningful	Feature selection
Meaningful variables are independent of each other	

Choosing PCA and Factor Analysis

Use Case	Possible Solution
Large number of X-variables	Principal Components Analysis (PCA) or Factor Analysis
Most of which are meaningful	
Highly correlated to each other	
Linearly related to each other	
For use in regression	

Choosing PCA and Factor Analysis

Use Case	Possible Solution
Large number of X-variables	
Most of which are meaningful	
Highly correlated to each other	Linear Discriminant Analysis (LDA) or Dictionary Learning
Linearly related to each other	
For use in classification	

Choosing Manifold Learning

Use Case	Possible Solution
Y not linearly related to X	Manifold learning
Very high dimensionality of X (e.g. pixel counts in image data)	
Many constraints on allowable values of X-variables (sparse features)	
Three-dimensional plots of Y against pairs of X indicate manifold shape	

Choosing Autoencoders

Use Case	Possible Solution
Extremely complex feature vectors Images, video, documents Pre-processing before using in neural networks	Autoencoders

Drawbacks of Reducing Complexity

Drawbacks of Reducing Complexity

Loss of information

Performance degradation

Computational intensive

Complex pipelines

Transformed features hard to interpret

Demo

Explore the Diabetes dataset for classification

Perform classification with all features to establish a baseline

Demo

Explore the Boston Housing Prices dataset

Implement linear regression with all features i.e. kitchen sink regression

Summary

Need for dimensionality reduction in building ML models

Bias-variance trade-off

Overfitting and the curse of dimensionality

Drawbacks of excessively complex models

Choosing dimensionality reduction techniques based on use case