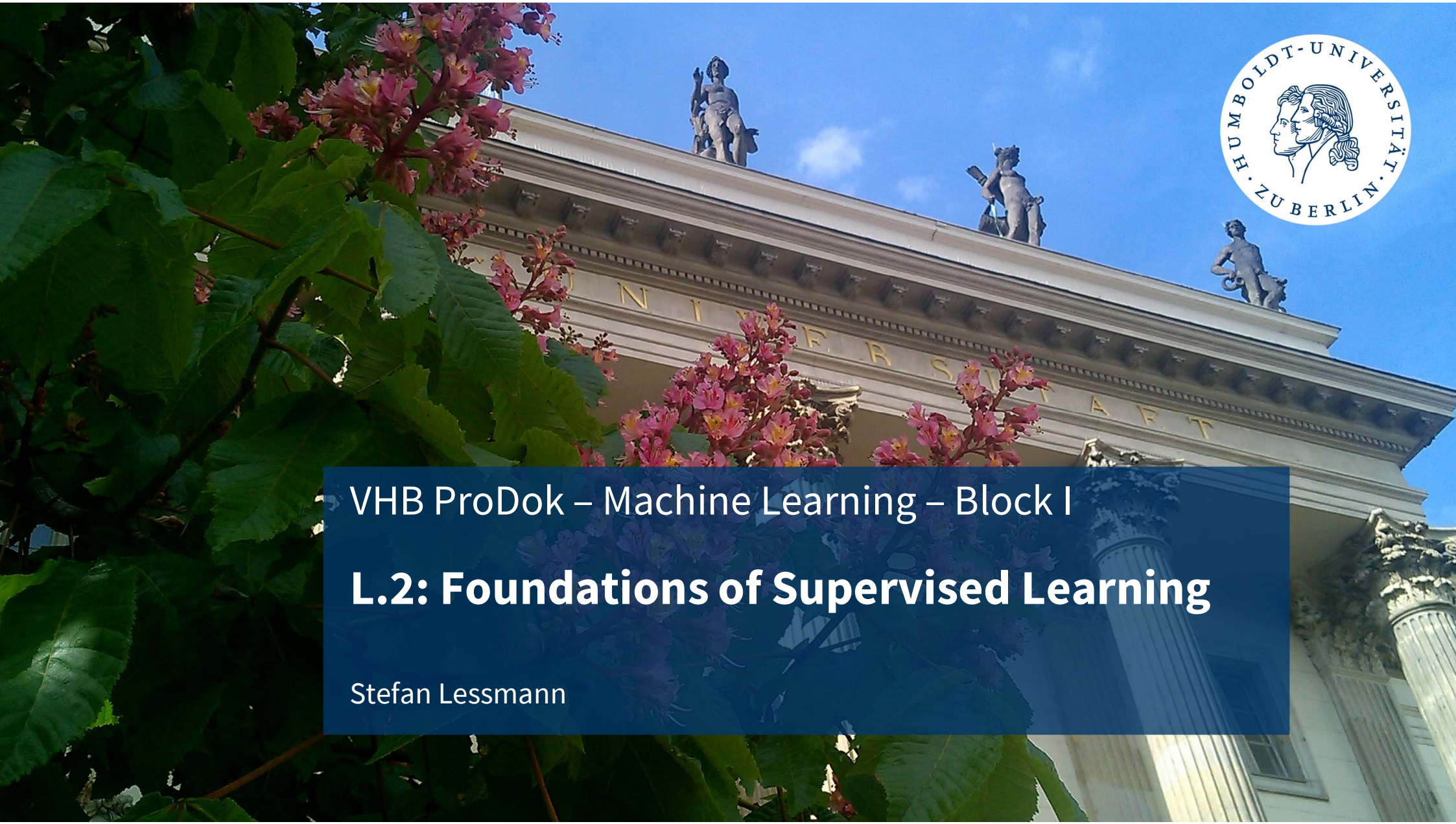
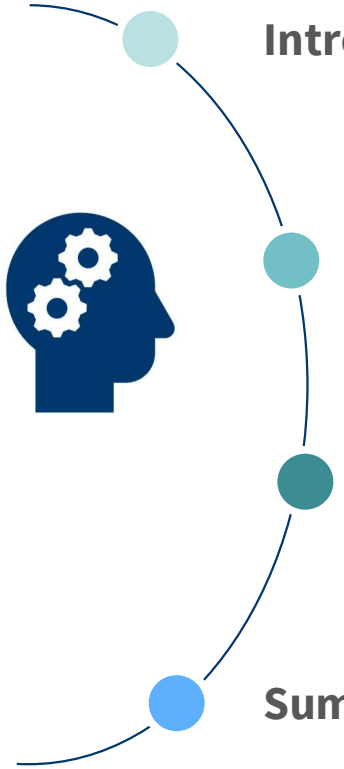


VHB ProDok – Machine Learning – Block I

L.2: Foundations of Supervised Learning

Stefan Lessmann

Agenda



Introduction

Supervised Learning Algorithms
Categorization, (semi-)parametric & nonparametric methods

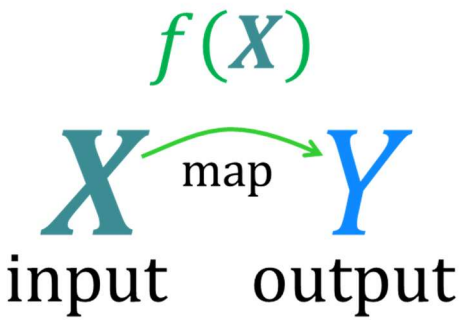
Bias-Variance Trade-Off
Bias, variance, and the fundamental problem of overfitting

Summary

Introduction

Supervised Machine Learning (SML)

Input to output mapping



- **The output variable defines a prediction target**
  - For an individual subject, this is the value we wish to know
  - We cannot observe the target (at the right time) or face a high cost to measure it (e.g., human labor)
- **The (input) features characterize each subject**
  - These values are easy/cheap to observe
  - We believe they determine the target value
- **SML algorithms approximate the hypothesized, unknown relationship between feature values and the target**
  - They receive data that exemplifies the relationship in the form of pairs of feature and target values
  - Using their internal logic, they then learn the mapping functions from these examples

Formalization of Supervised Machine Learning

Resale price forecasting example

- **We aim at forecasting resale prices (our target variable) denoted by y**
- **We assume that resale prices y depend on features x**
  - We do not know how exactly resale prices depend on feature values
  - But we have access to historical data $\mathcal{D} = \{y_i, x_i\}_{i=1}^{n}$ that exemplifies the relationship
- **At decision time, i.e., when need a forecast, we can observe x but not y**
- **We use algorithms to learn a model f that maps from features to target $f: x \mapsto y$**

PRODUCT	LIST PRICE [$]	AGE [month]	CLIENT INDUSTRY	…
Dell XPS 15’	2,500	36	Mining	…
Dell XPS 15’	2,500	24	Health	…
Dell XPS 17’	3,000	36	Manufacturing	…
Lenovo Yoga 11’	799	12	Office	…
Lenovo Yoga 13’	1,100	12	Office	…
…	…	…	…	…

$$x = (x_1, x_2, \dots, x_m) \in \mathbb{R}^m$$

$$f(x)$$

OBSERVED RESALE PRICE [$]
347
416
538
…
266
…

$$y \in \mathbb{R}$$

Two Flavors of Supervised Machine Learning

Regression & classification methods model numerical & discrete targets

- **Credit scoring example**
  - Financial institutions receive loan applications $x \in \mathbb{R}^m$
  - Loan approval decisions depend on an estimate of the applicant's probability to default $y \in \{0; 1\}$
  - When deciding on an application, we can observe x but not y
  - Historical data $\mathcal{D} = \{y_i, x_i\}_{i=1}^n$ illustrates which *granted* applications were repaid / defaulted
- **We use algorithms to learn a model f that maps from features to target $f: x \mapsto y$**

BUREAU SCORE	DEBT/ INCOME	COLLATERAL	YEARS AT ADDRESS	AGE	…
650	20%	Yes	2	<21	…
280	43%	No	0	21-29	…
575	33%	No	12	>50	…
715	24%	Yes	1	21-29	…
580	18%	No	6	40-50	…

$$x = (x_1, x_2, \dots, x_m) \in \mathbb{R}^m$$

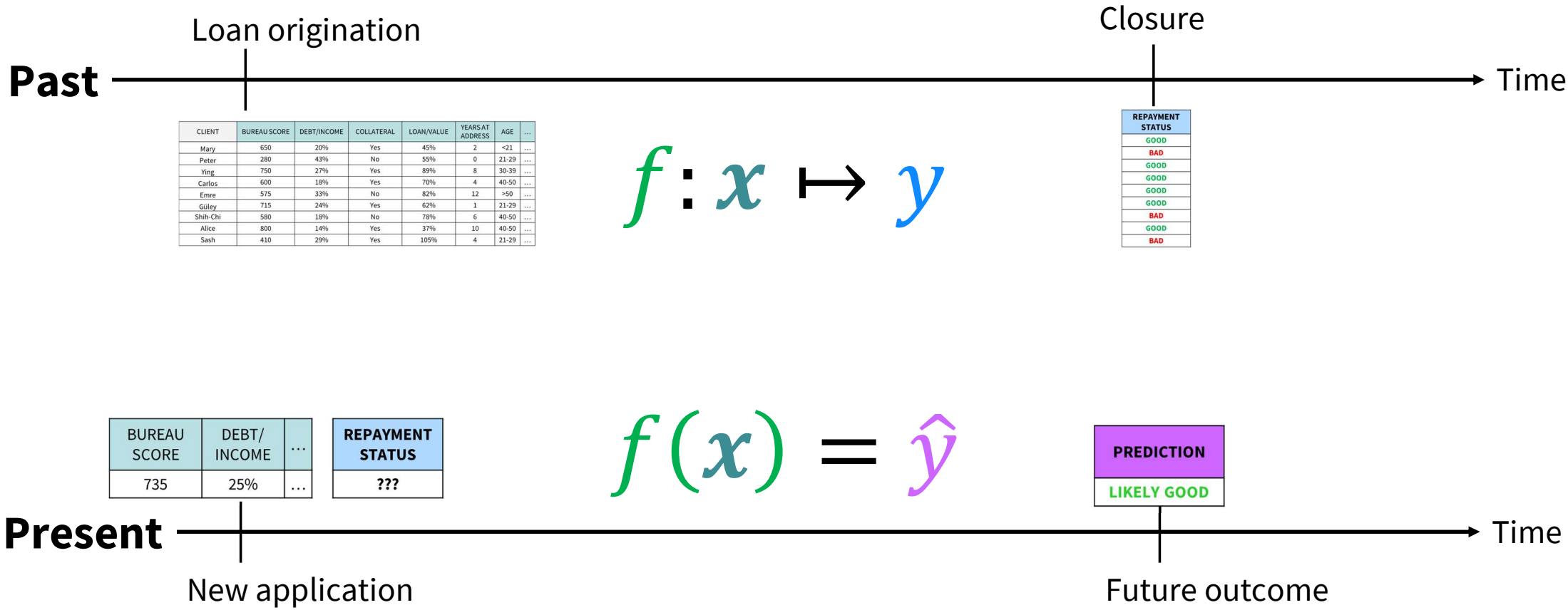
$$f(x)$$

$$y \in \{0; 1\}$$

REPAYMENT STATUS
Good (0)
Bad (1)
Good (0)
Good (0)
Bad(1)

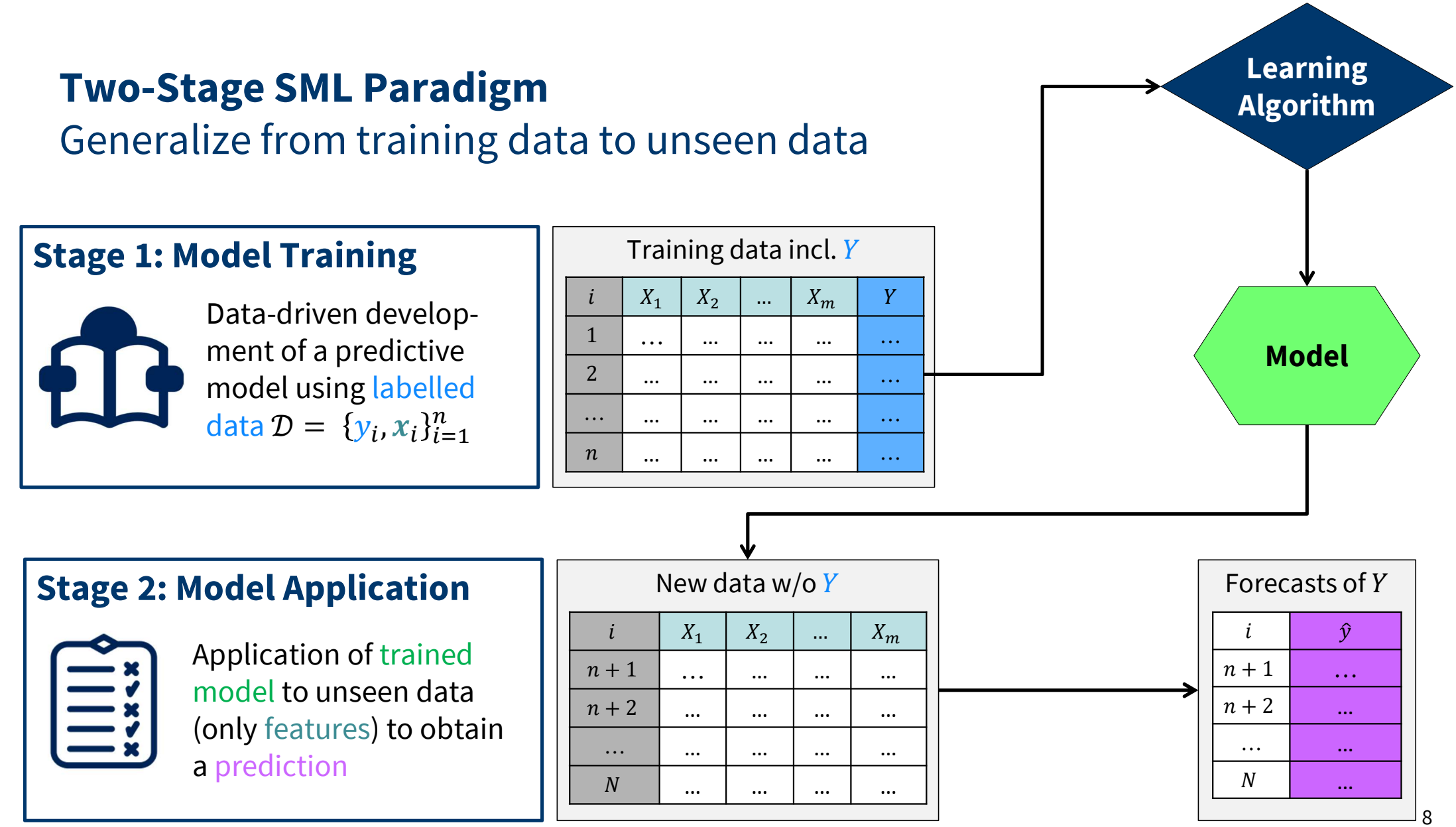
Credit Scoring: Predictive Analytics Perspective

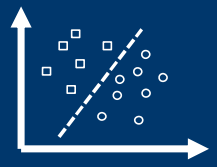
Train model on historical data to predict the risk of a new application



Two-Stage SML Paradigm

Generalize from training data to unseen data





Supervised Learning Algorithms

Categorization, (semi-)parametric & nonparametric methods

Algorithms for Supervised Learning (Selection)

Approaches split into (semi-)parametric and nonparametric models

- **Parametric models assume a functional form for the feature-target relationship**
  - The model is fully characterized by a fixed number of parameters $\boldsymbol{\theta}$
    - Leading to the stochastic model equation $y = f(\boldsymbol{x}; \boldsymbol{\theta}) + \epsilon$
    - With ϵ denoting the error term and $\boldsymbol{\theta} \in \mathbb{R}^k$, $k < \infty$
- **Nonparametric models learn the functional from data**
- **Semi-parametric models: parametric structure + nonparametric components**

Tree- and prototype-based algorithms (non-parametric)	Regression-type algorithms (semi-/parametric)
▪ CART, CHAID, C4.5 ▪ Bagging / Random Forest ▪ Gradient Boosting / XGB ▪ Nearest neighbors ▪ …	▪ Generalized linear models (GLM) ▪ Generalized additive models (GAM) ▪ Artificial neural networks (ANN) ▪ Support vector machines (SVM ▪ …

Linear Models for Supervised Machine Learning

Three well-known econometric approaches

- **Conditional mean structure**

$$y = \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}] + \epsilon$$

- **Additive predictor**

$$\eta(\boldsymbol{x}) = b + \sum_{j=1}^{m} f_j(x_j)$$

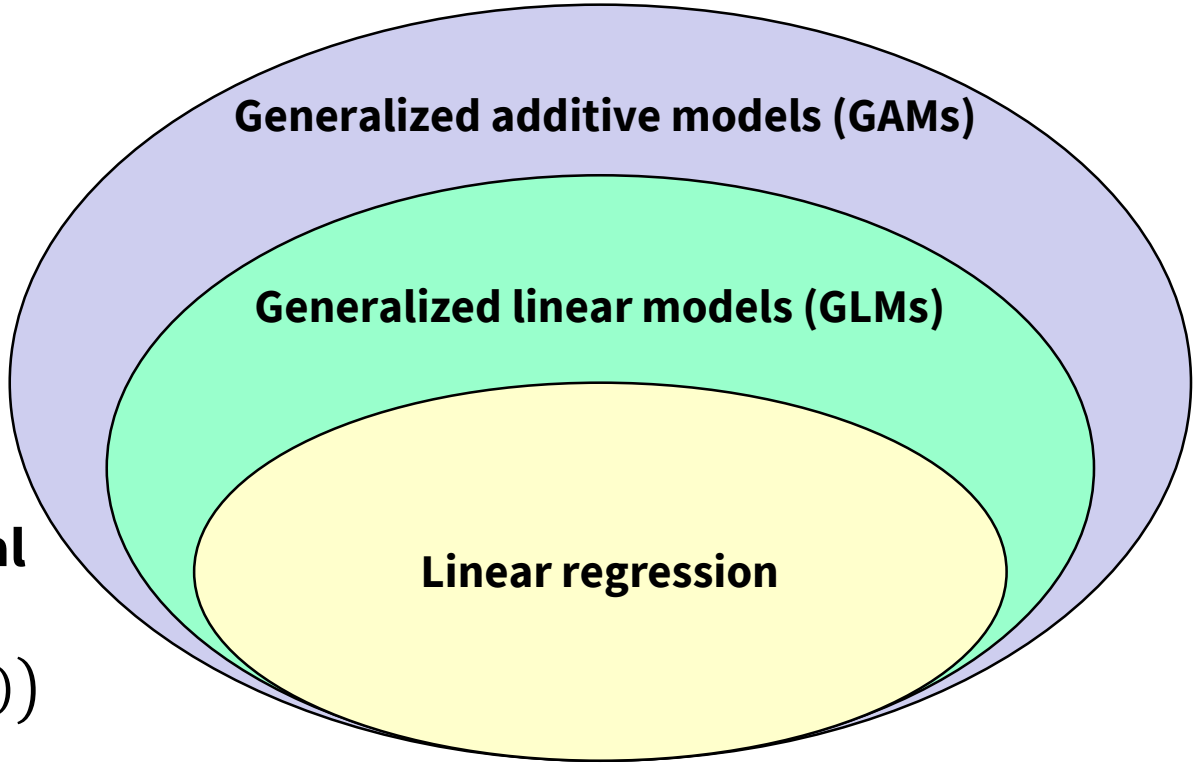
  - With f_j denoting some smooth function
  - For GLM $f_j(x_j) = w_j x_j$

- **Link function g connecting conditional expectation to additive predictor**

$$g(\mathbb{E}[\boldsymbol{y}|\boldsymbol{x}]) = \eta(\boldsymbol{x}) \Leftrightarrow \mathbb{E}[\boldsymbol{y}|\boldsymbol{x}] = g^{-1}(\eta(\boldsymbol{x}))$$

- **Prediction equation**

$$\hat{y} = g^{-1}(\eta(\boldsymbol{x}))$$



Linear Models for Supervised Machine Learning

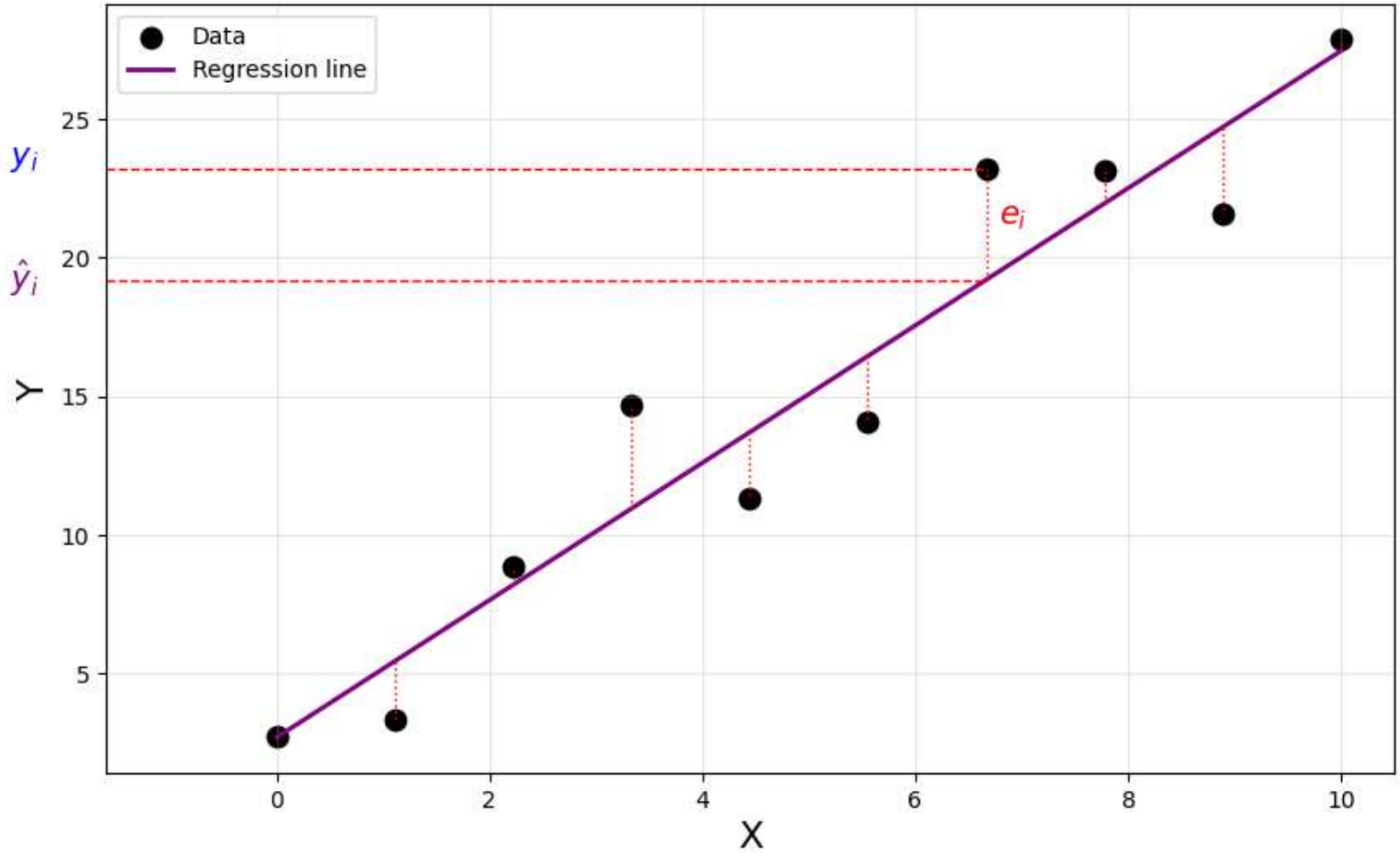
Three well-known econometric approaches

- **Linear regression: linearity in conditional mean and Gaussian errors**
  - Model equation: $\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}] = \eta(\boldsymbol{x}) = b + \sum_{j=1}^{m} w_j x_j$
  - and $g(\cdot) = \mathrm{id}$
- **Generalized linear models (GLMs) add a link function for non-Gaussian targets**
  - $g(\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]) = \eta(\boldsymbol{x}) = b + \sum_{j=1}^{m} w_j x_j$
  - Example: logistic regression for binary outcomes, $g := \log\left(\frac{p}{1-p}\right)$ with $p = \Pr(y = 1 \mid \boldsymbol{x})$
- **Generalized additive models (GAMs): replace linear effects by smooth nonlinear functions while keeping additivity and interpretability**
  - Model $g(\mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]) = b + \sum_{j=1}^{m} w_j f_j(x_j)$
  - Link function depends on the target's distribution (exactly as in GLM)
  - Binary → logit, count data → log, Gaussian → identity

Empirical Risk Minimization (ERM) Principle

Train (aka estimate) model through minimizing a loss function

- **Residuals** e
- **Loss function**
- **Loss minimization**



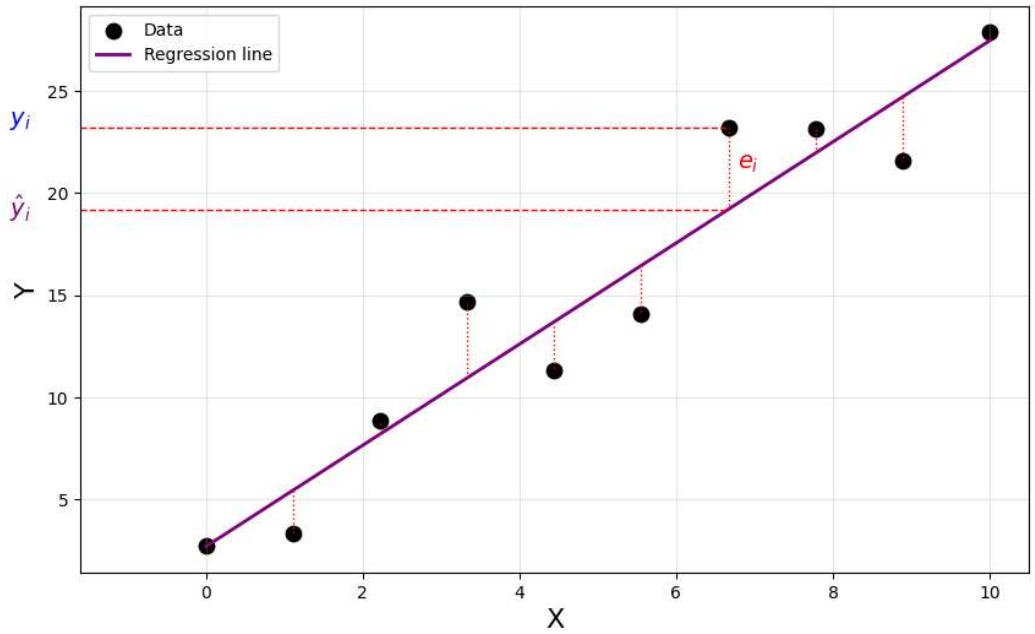
Empirical Risk Minimization (ERM) Principle

Train (aka estimate) model through minimizing a loss function

- **Residuals e measures deviation between actual target value and model prediction**

  $e = (y - \hat{y})$

- **Loss function assesses model fit**
  - Squared error loss function
    - Pointwise loss $J(y_i, \hat{y}_i) = e_i^2 = (y_i - \hat{y}_i)^2$
    - Dataset-level loss $J(y, \hat{y}) = \sum_{i=1}^{n} e_i^2$
  - Several other loss functions exist



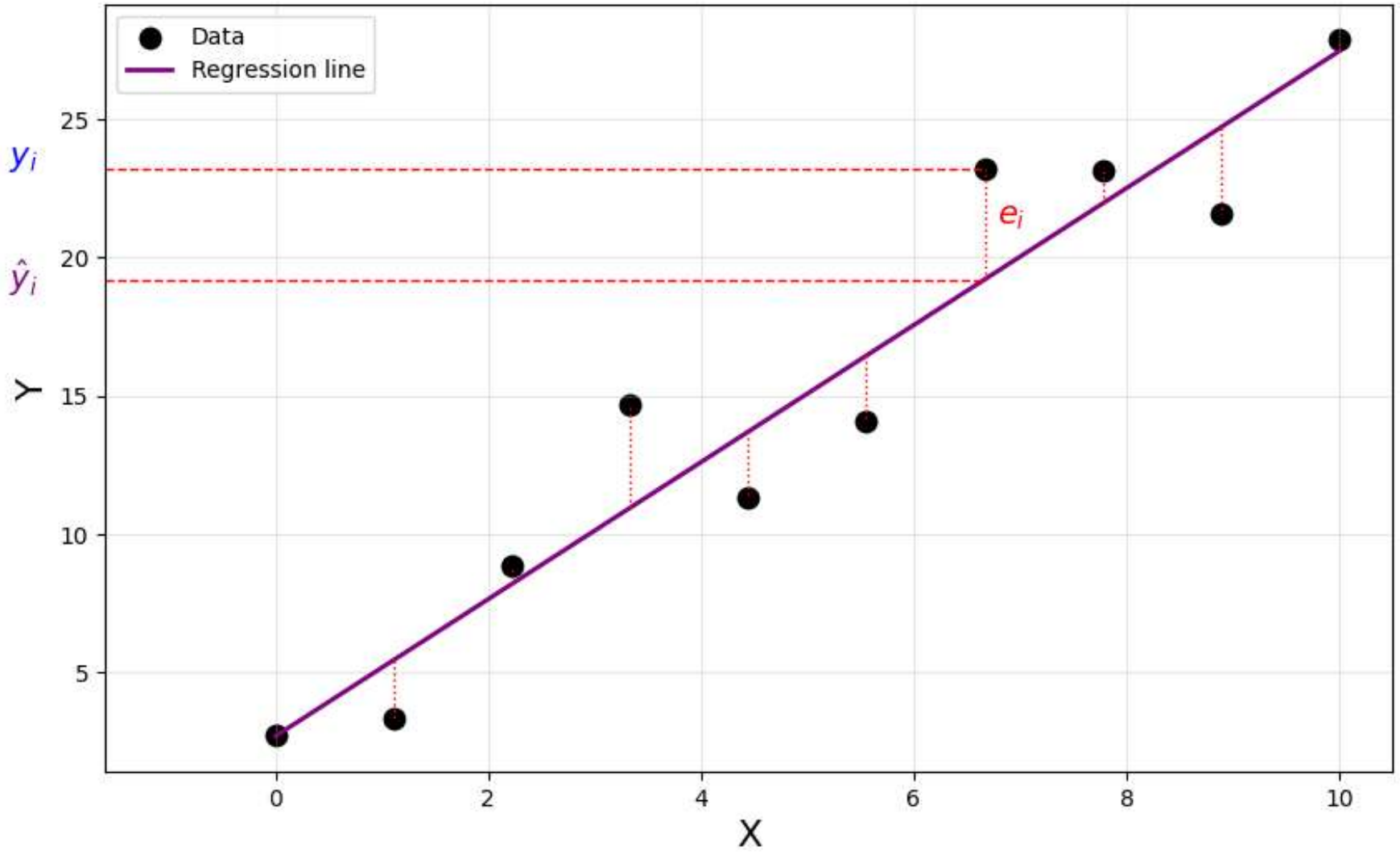
- **Loss minimization**
  - OLS estimator $\hat{b}, \hat{w} = \text{argmin}_{(b,w)} \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \left(b + \sum_{j=1}^{m} w_j x_j\right)\right)^2$
  - OLS estimator has an analytical solution (i.e., *normal equation*)
  - In general, we perform the minimization using iterative algorithms (e.g., gradient descent)

Empirical Risk Minimization (ERM) Principle

Estimating generalizable models by minimizing empirical risk

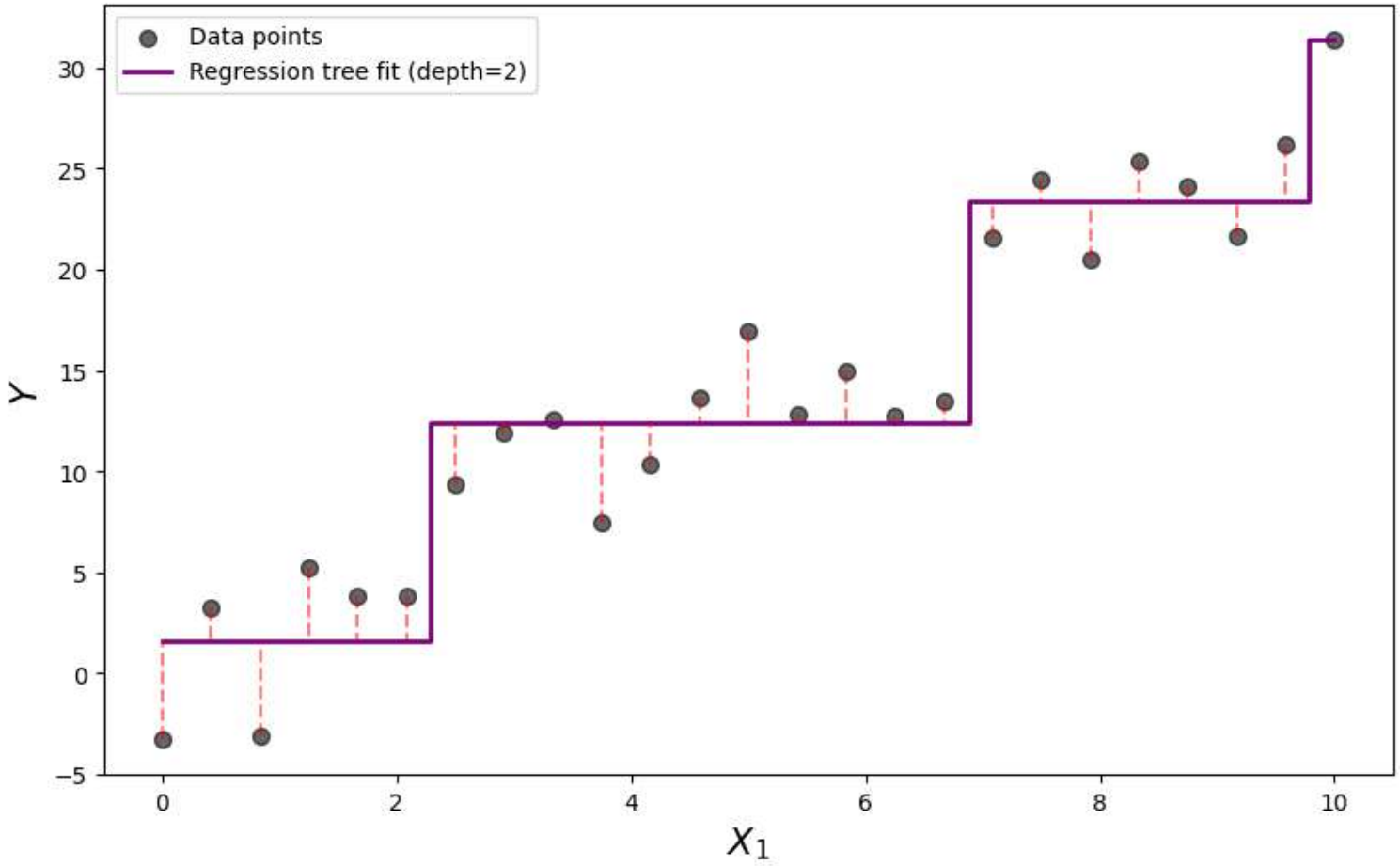
- **Minimize the expected loss of a model over the joint distribution of X and Y**
  - Population loss $J(f) = \mathbb{E}_{(X,Y)}\left[J\left(Y, f(X)\right)\right]$
  - ERM objective $f^* = \text{argmin}_{f \in \mathcal{F}}\ \mathbb{E}_{(X,Y)}\left[J\left(Y, f(X)\right)\right]$
- **Empirical risk**
  - Loss computed over a dataset $\mathcal{D} = \{y_i, x_i\}_{i=1}^n$
  - Empirical loss/risk $\hat{J}(f) = \frac{1}{n}\sum_{i=1}^n J(y_i, f(x_i))$
  - ERM estimator $\hat{f} = \text{argmin}_{f \in \mathcal{F}}\ \frac{1}{n}\sum_{i=1}^n J(y_i, f(x_i))$
  - with $\mathcal{F}$ denoting the class of functions the model can learn
- **Maximum likelihood estimation**
  - Idea is to maximize the probability of the model given the data
  - Specific instance of the ERM principle

Excursus: Loss Minimization and Probability Maximization



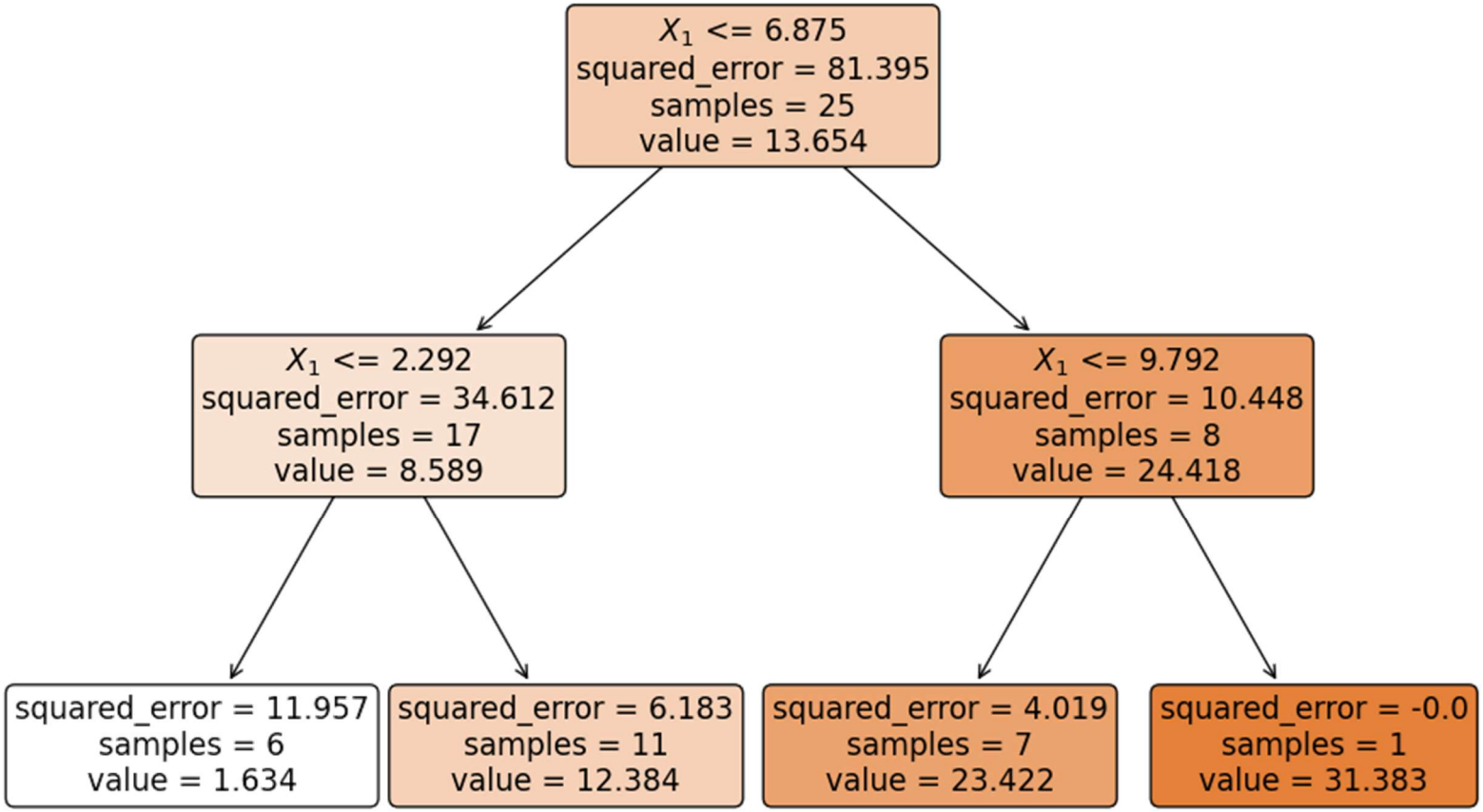
Beyond (Semi) Parametric Models: Classification & Regression Trees

Tree-based methods recursively partition the feature space



Beyond (Semi) Parametric Models: Classification & Regression Trees

Tree-based methods recursively partition the feature space



Root node

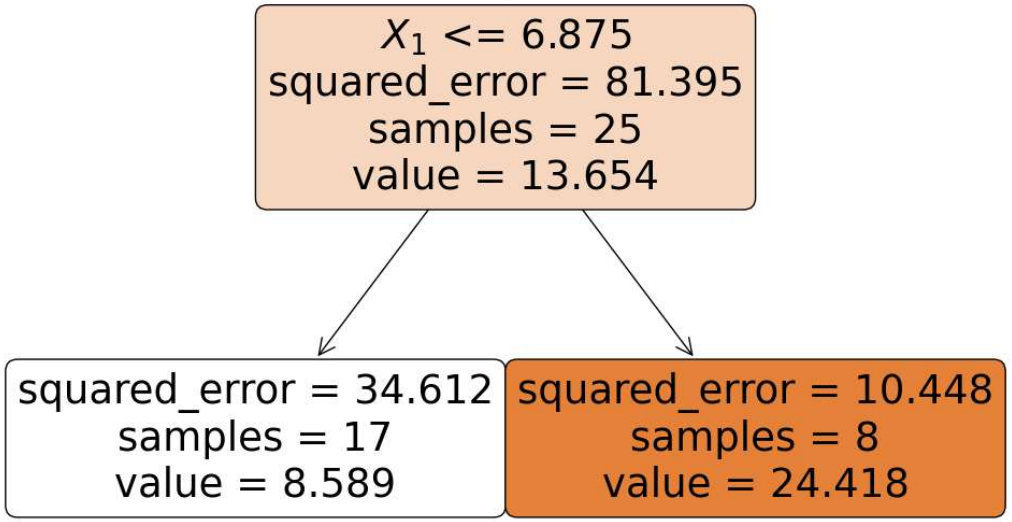
Splitter/internal nodes

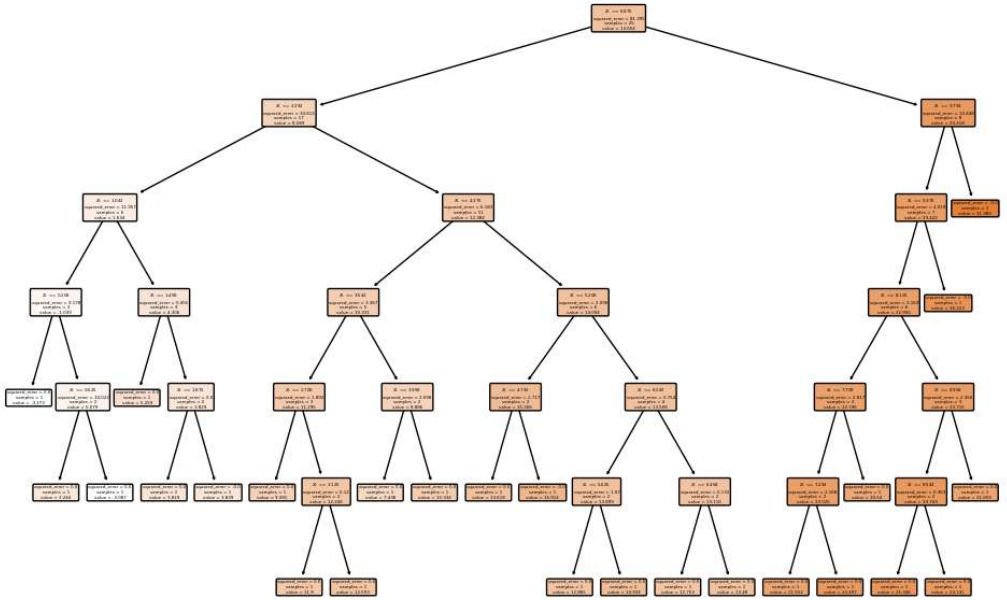
Leaf nodes

Simple and Complex Trees

Tree depth governs model complexity. Complexity determines residuals.

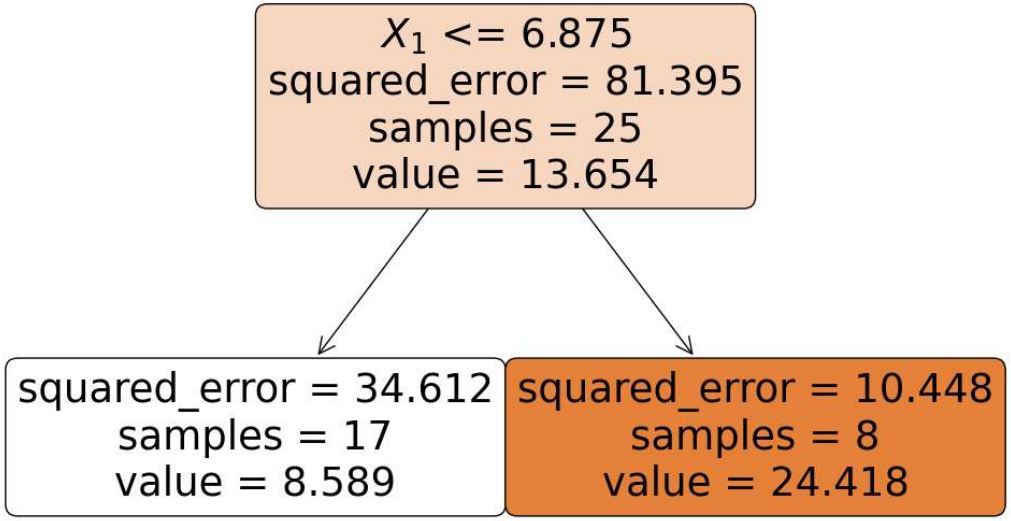
- **Hyperparameters govern the behavior of ML algorithms (e.g., depth of a tree)**
- **Many hyperparameters concern the complexity of the learnt model**
- **Model complexity is closely related to model residuals**

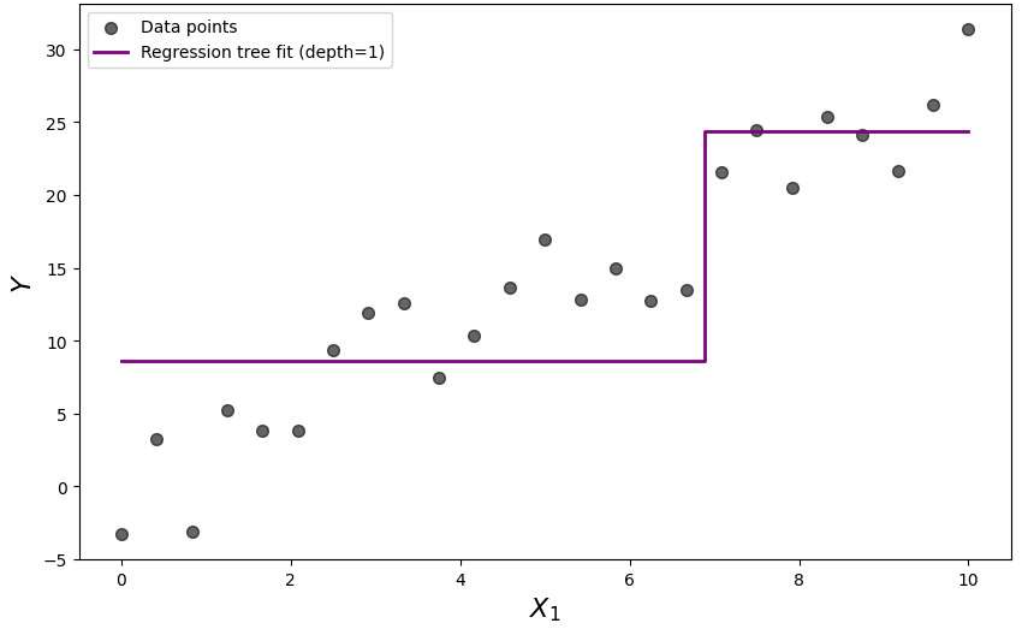




Simple and Complex Trees

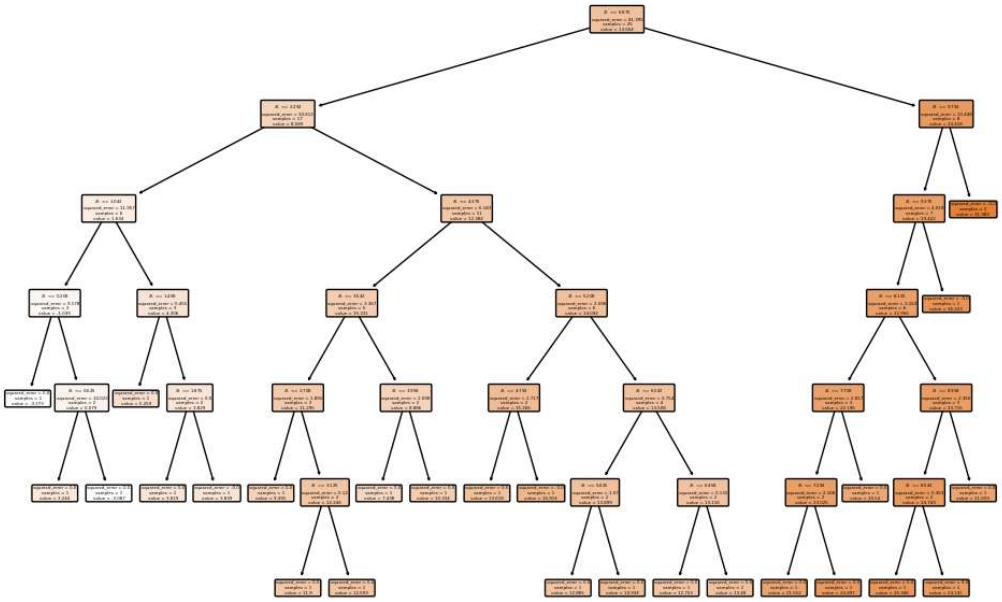
Total sum of (squared) residuals is large for the shallow tree

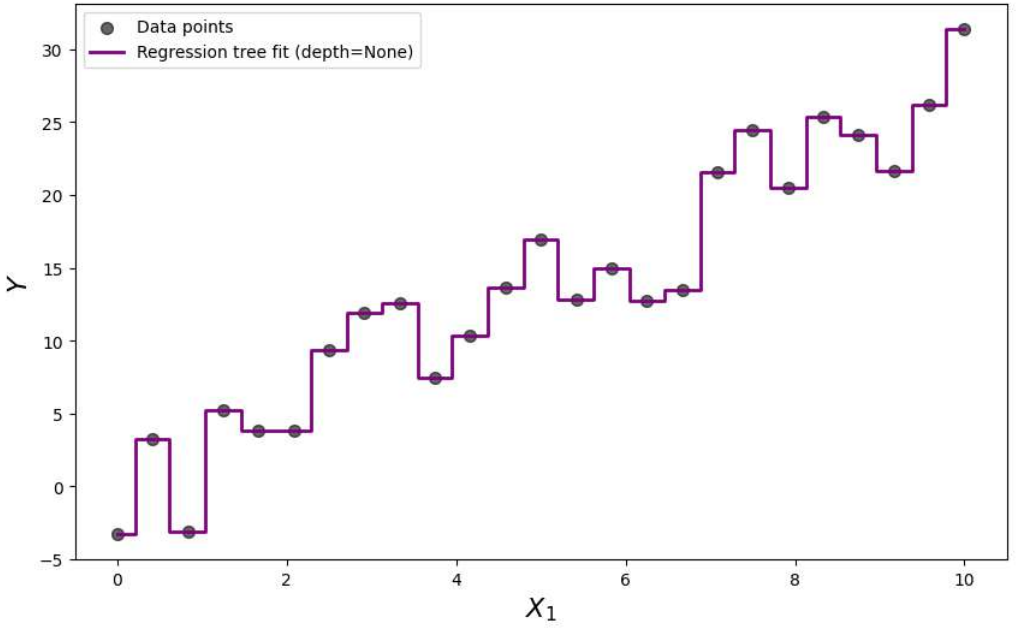




Simple and Complex Trees

Residuals are zero for the deep tree



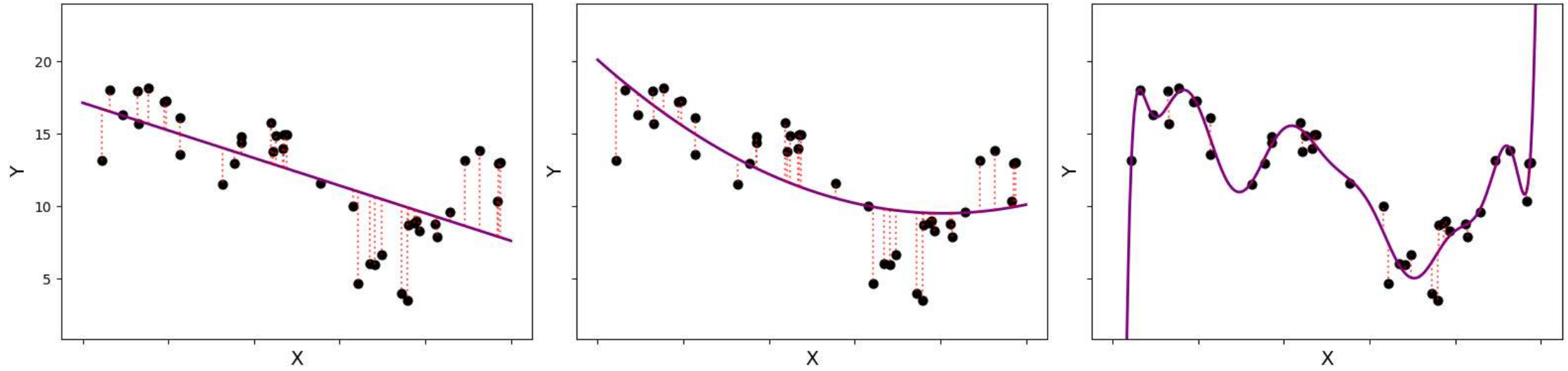


Bias-Variance Trade-Off

Bias, variance, and the fundamental problem of overfitting

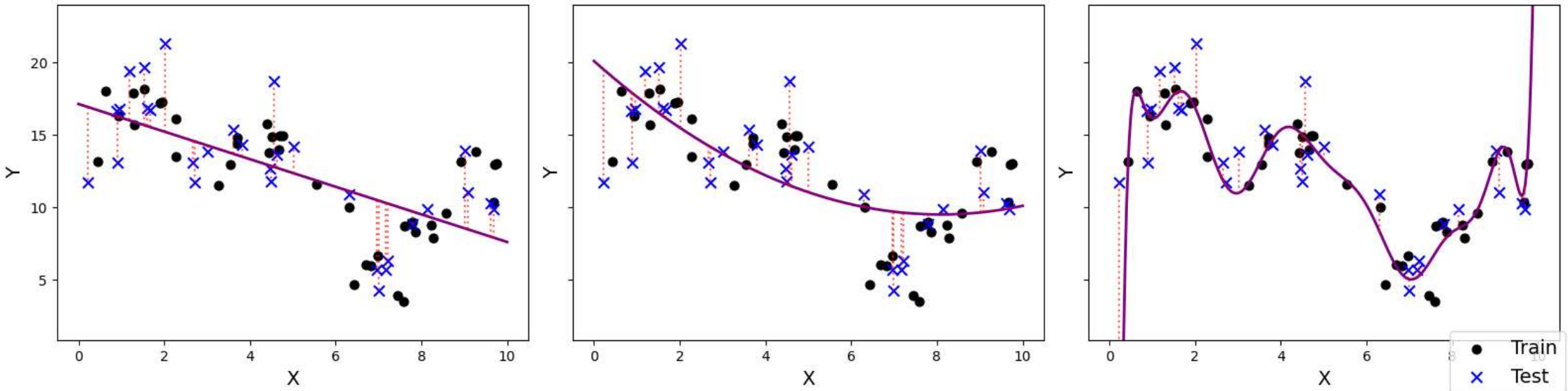
Three Candidate Regression Models for a Given Dataset

Which model do you prefer?



Three Candidate Regression Models for a Given Dataset

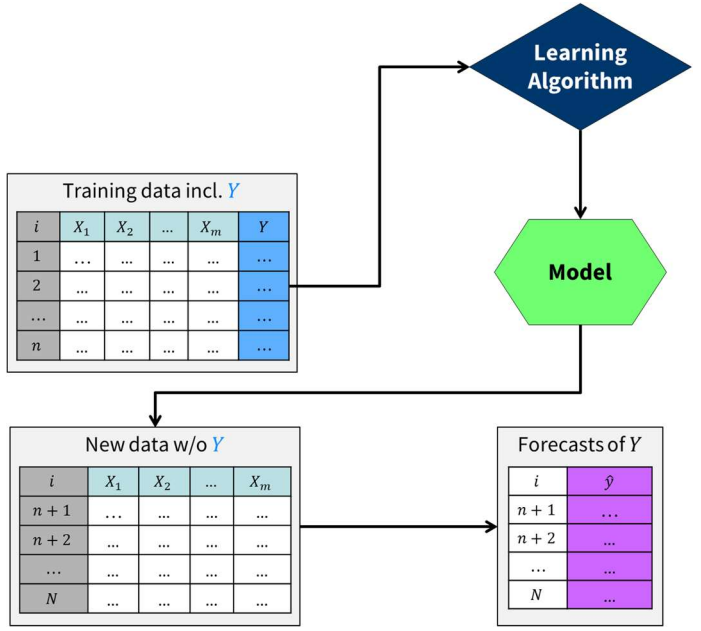
ML models can overfit the training data leading to poor generalization

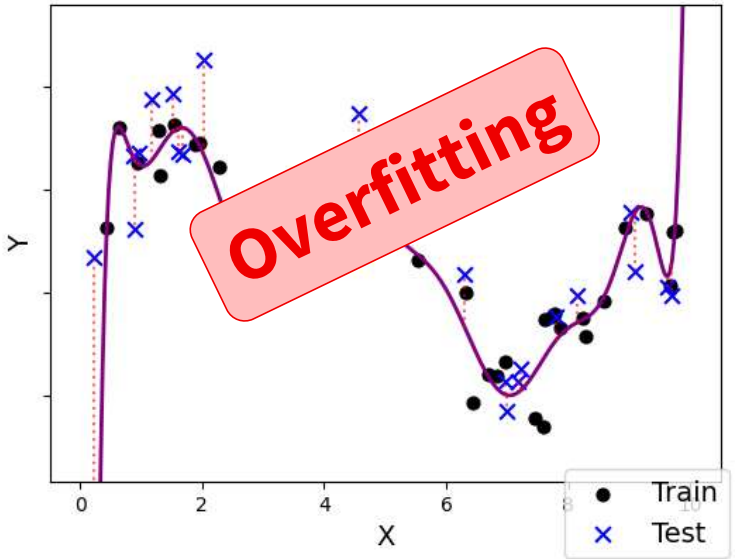


The Fundamental Problem of Overfitting

But why would a model overfit?

- **Recall how we organize model training**
  - Loss minimization → minimal loss is *mathematically* optimal
  - *Optimal* fit of the training sample does not imply a good model (i.e., capable to generalize to new data)
  - The model might be too specific and geared toward the training sample
- **The training data set is a sample**
  - We assume the sample represents the population well
  - But every sample exhibit random (sampling) variation
- **A complex ML model can capture both, true structure and idiosyncrasies of training sample**
- **Large error if applying such model to new data**

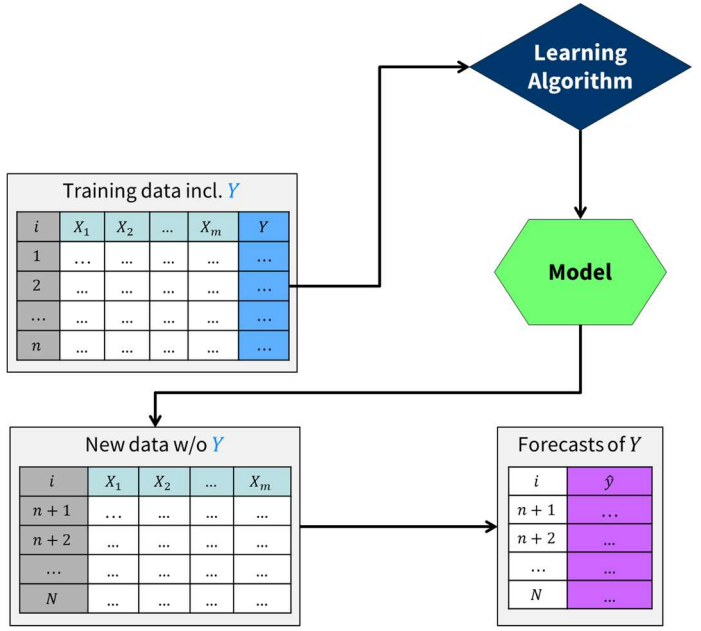




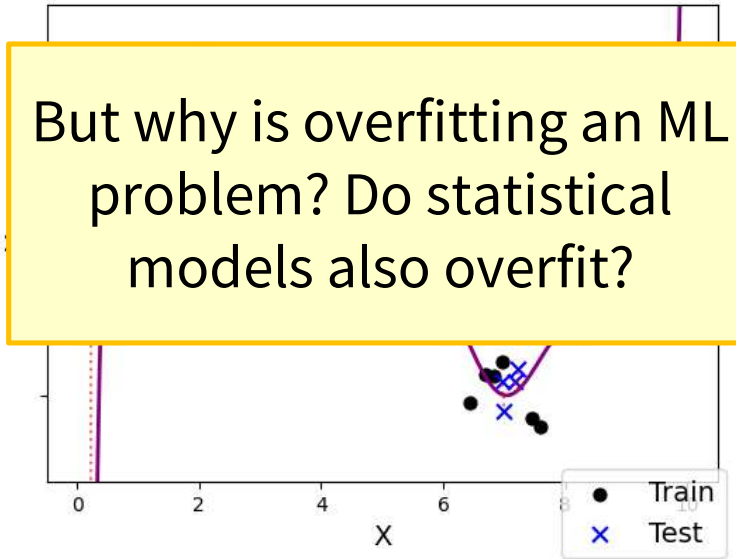
The Fundamental Problem of Overfitting

But why would a model overfit?

- **Recall how we organize model training**
  - Loss minimization → minimal loss is *mathematically* optimal
  - *Optimal* fit of the training sample does not imply a good model (i.e., capable to generalize to new data)
  - The model might be too specific and geared toward the training sample
- **The training data set is a sample**
  - We assume the sample represents the population well
  - But every sample exhibit random (sampling) variation
- **A complex ML model can capture both, true structure and idiosyncrasies of training sample**
- **Large error if applying such model to new data**



But why is overfitting an ML problem? Do statistical models also overfit?



The Trade-Off Between Bias and Variance

The generalization error depends on a model's bias and variance

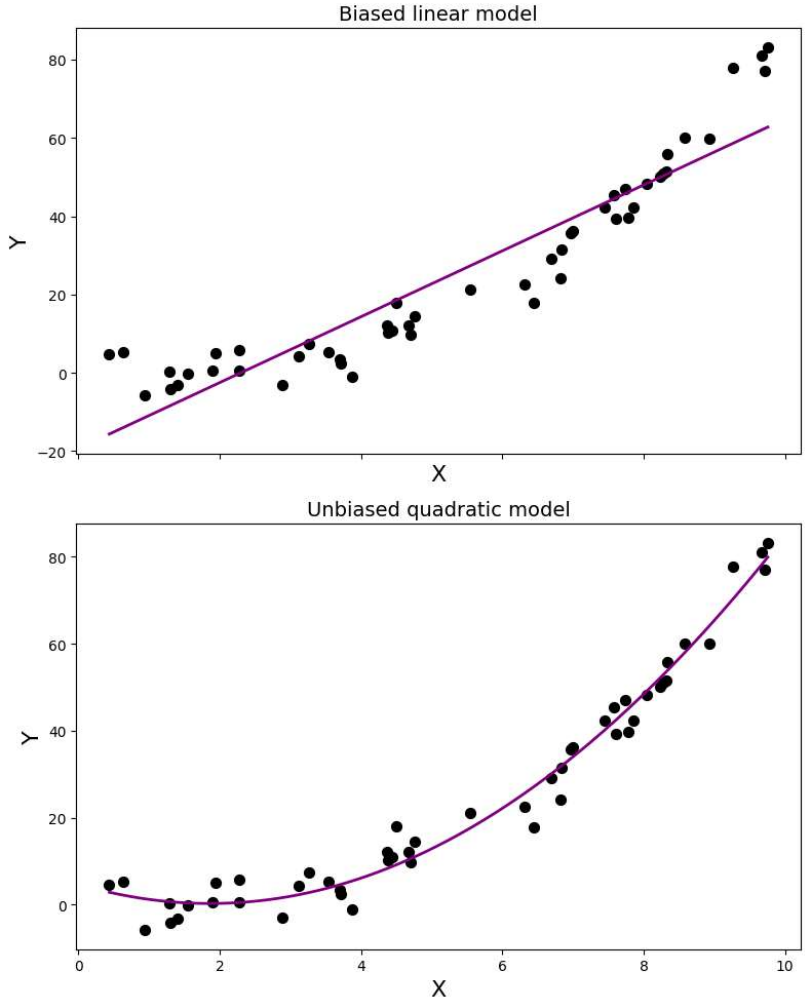
- **Generalization error: a model's error on *any* data from the population**
  - True target of supervised machine learning
  - Not observable in practice
  - Approximated by the empirical loss on a data sample
- **Bias: can the model learn the relationship between features and the target?**
  - Recall ERM objective

$$f^* = \text{argmin}_{f \in \mathcal{F}} \ \mathbb{E}_{(X,Y)}[J(Y, f(X))]$$

  - Bias asks whether the algorithm can learn

$$f^*(x) = \mathbb{E}[Y|X = x]$$

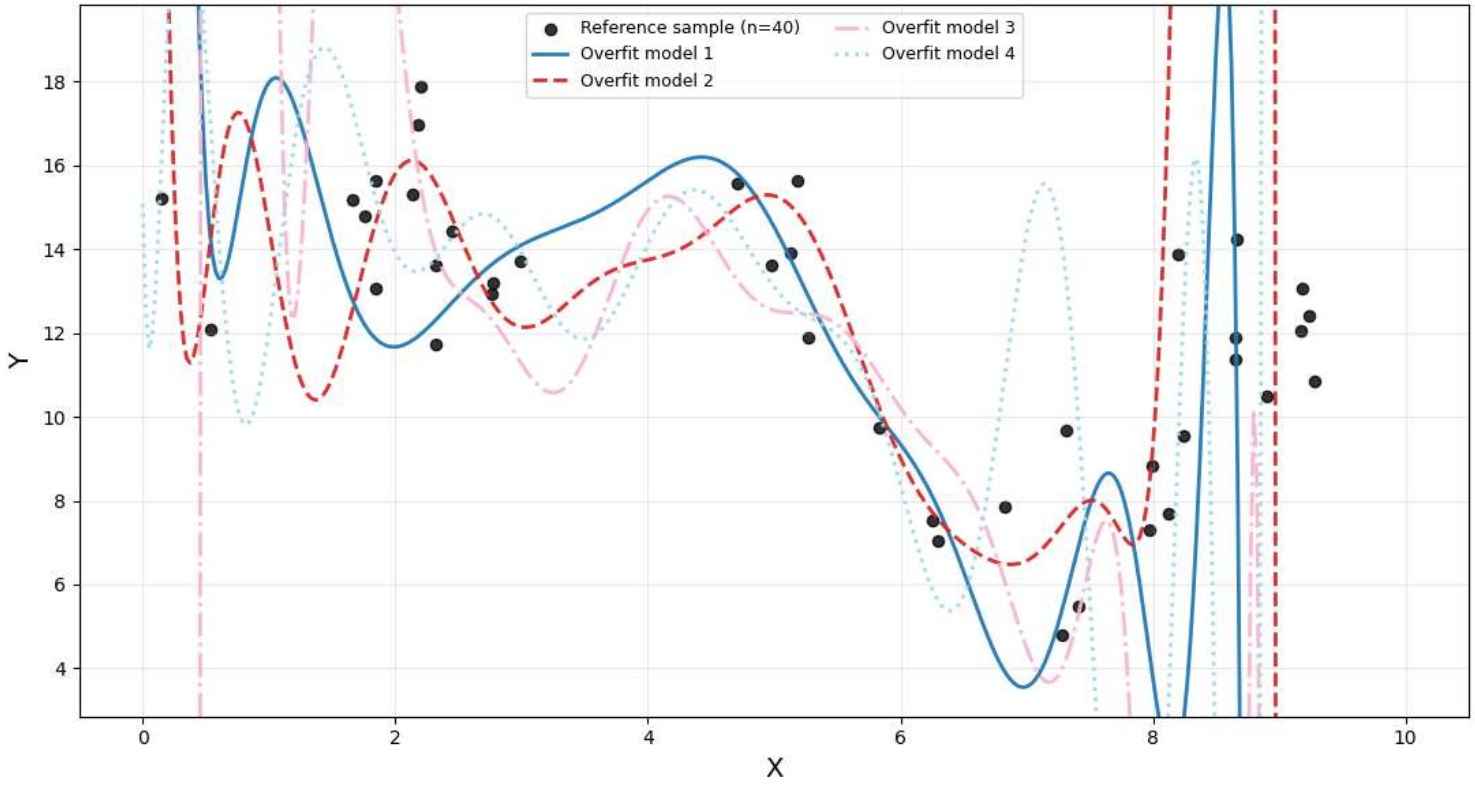
  - The more complex a model the lower its bias



The Trade-Off Between Bias and Variance

The generalization error depends on a model's bias and variance

- **Generalization error: a model's error on *any* data from the population**
- **Variance: how sensitive is the model to the training sample?**
  - Assume you estimate a linear regression model over many random samples drawn from one dataset
  - How much will coefficients (i.e., the model) vary across samples?
  - Changes in models imply changes in forecast,
  - So variance ≈ variation in model predictions



Bias-Variance Trade-Off and Overfitting

Overfitting is a problem of complex (low-bias) models that suffer high variance

- **Model complexity is about its expressive power**
- **Simple models**
  - High bias & low variance
  - E.g., linear regression
- **Complex models**
  - Low bias & high variance
  - E.g., deep regression tree
- **SML aims at a good compromise**
- **Common paradigm in ML**
  - Use an advanced, complex model
  - Manage / control complexity somehow

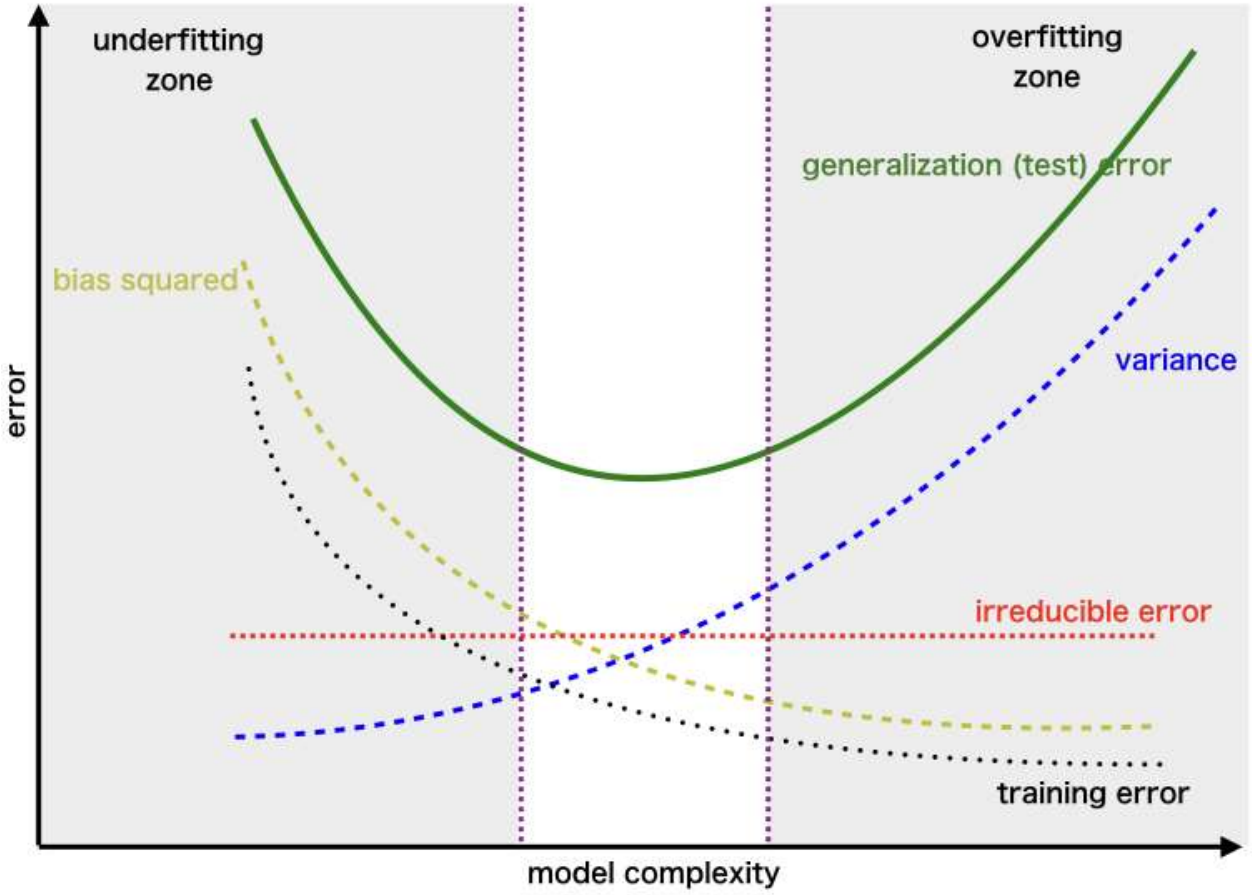
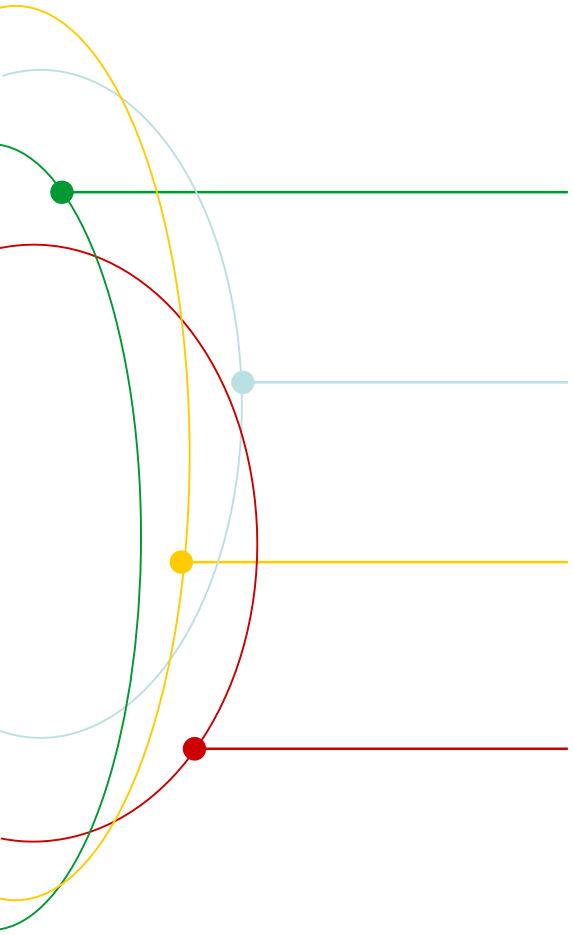


Image source: https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9

Summary

Summary



Learning goals

- Supervised machine learning paradigm
- Bias-variance trade-off and overfitting

Findings

- SML approximates a functional mapping from inputs to outputs
  - Parametric models assume a functional form a prior
  - Nonparametric models learn the mapping from data
- Training involves minimizing a loss function
- Complex models can overfit the training data leading to poor generalization
- Overfitting is a problem of low bias and high variance

What next

- Tree-based SML algorithm
- Practices for model evaluation

Thank you for your attention!

Stefan Lessmann

Chair of Information Systems
School of Business and Economics
Humboldt-University of Berlin, Germany

Tel. +49.30.2093.5742
Fax. +49.30.2093.5741

stefan.lessmann@hu-berlin.de
http://bit.ly/hu-wi

www.hu-berlin.de

Photo: Heike Zappe