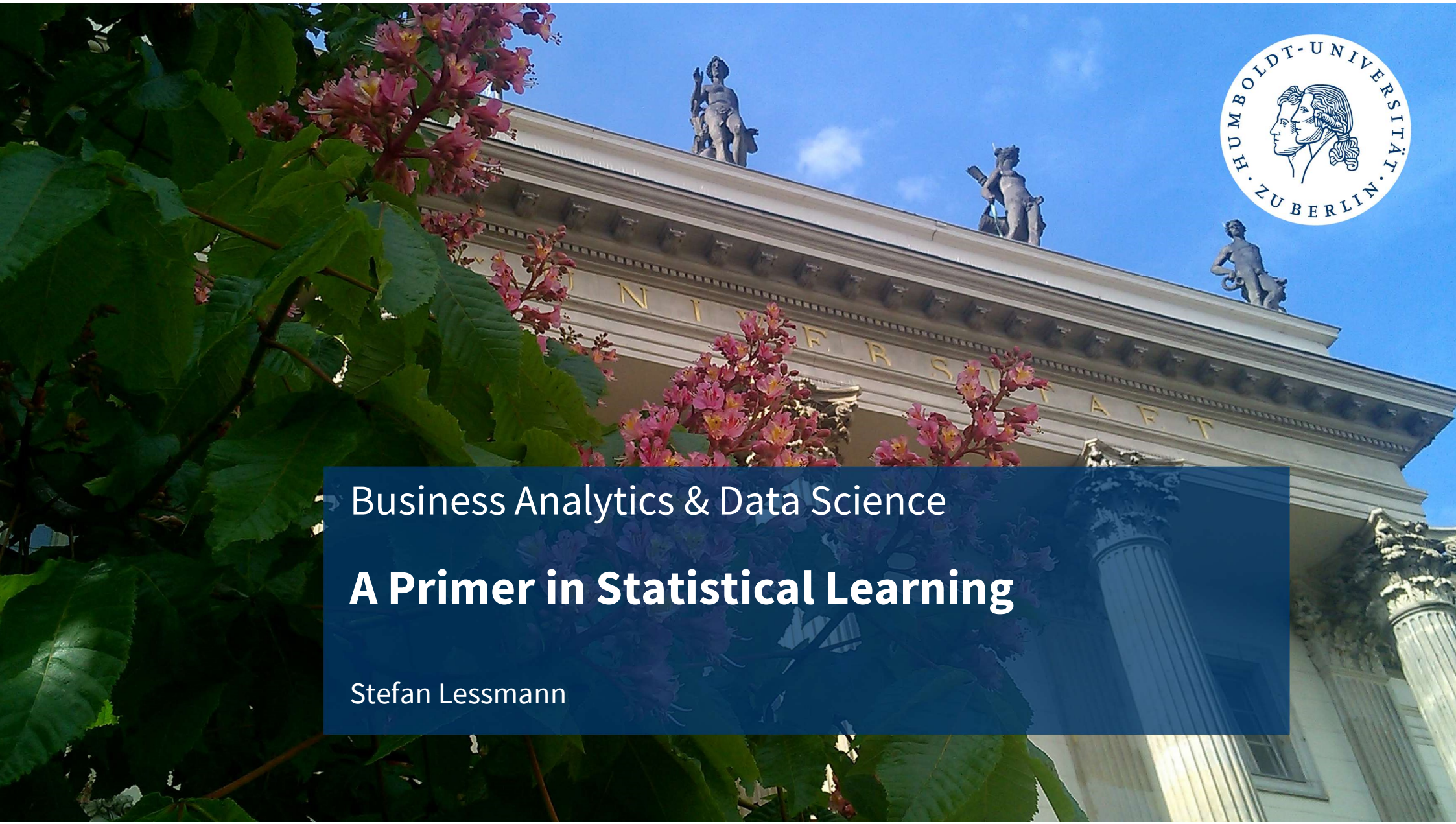
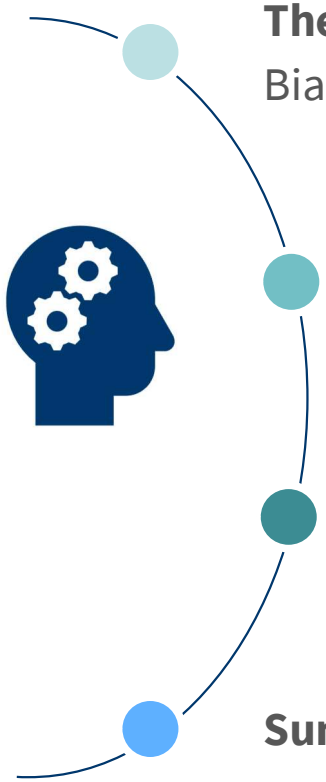


Business Analytics & Data Science

A Primer in Statistical Learning

Stefan Lessmann

Agenda



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

Regularization
Idea, connection to bias-variance trade-off, implementation

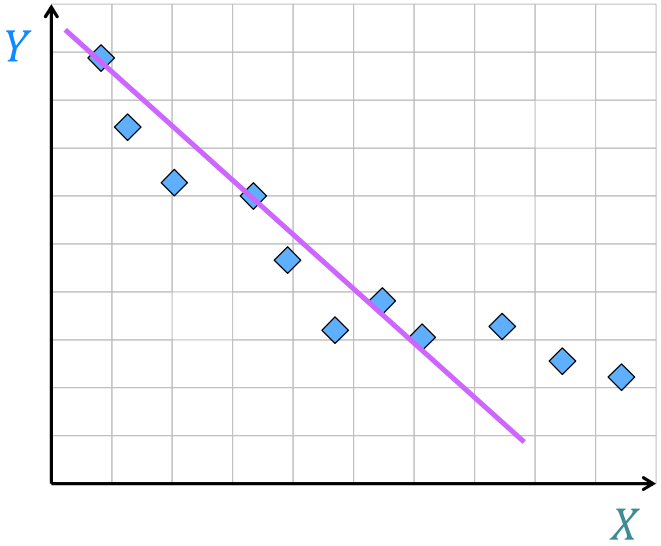
Model Selection
Search strategies and process perspective

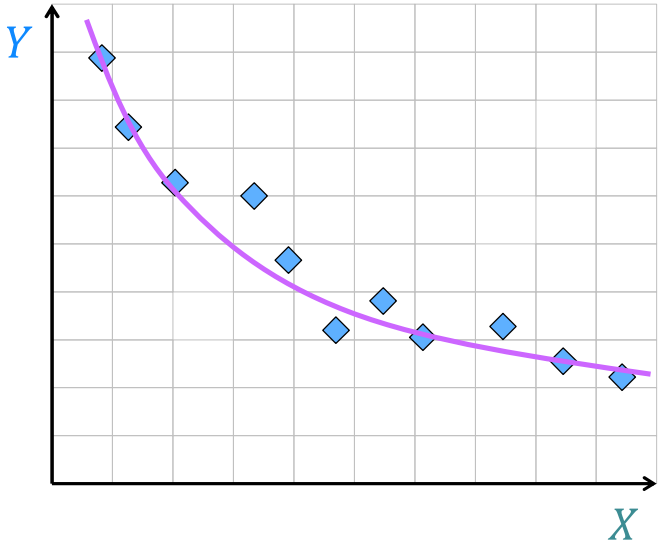
Summary

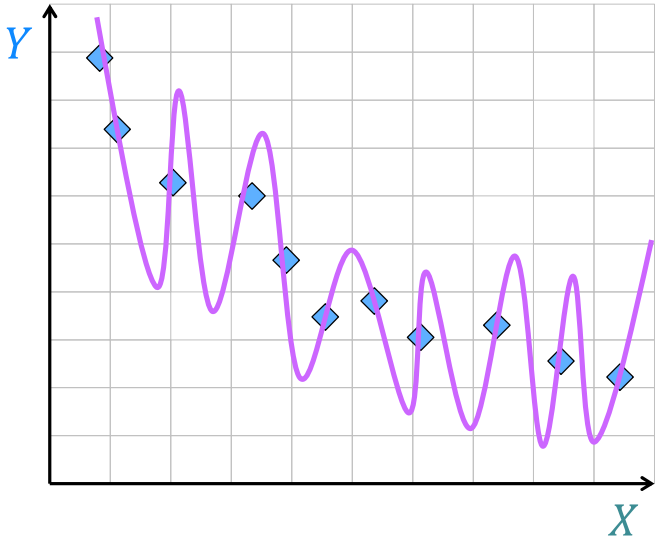
Bias-Variance Trade-Off

Bias, variance, and the fundamental problem of overfitting

Three Alternative Regression Models for same Data. Which one is best?



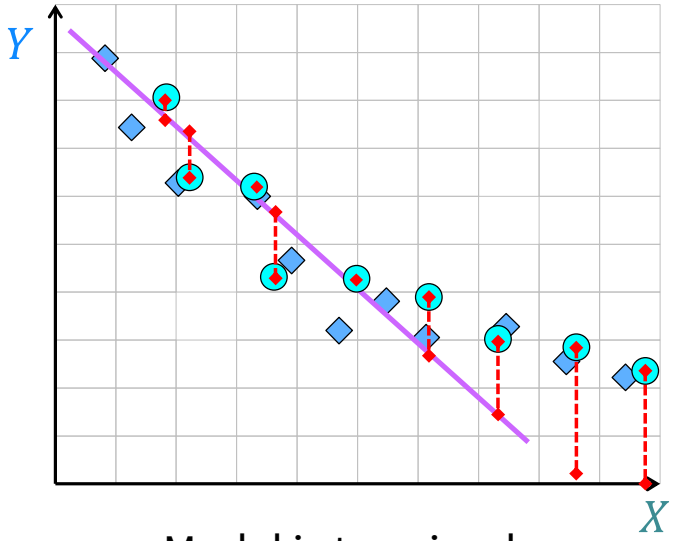




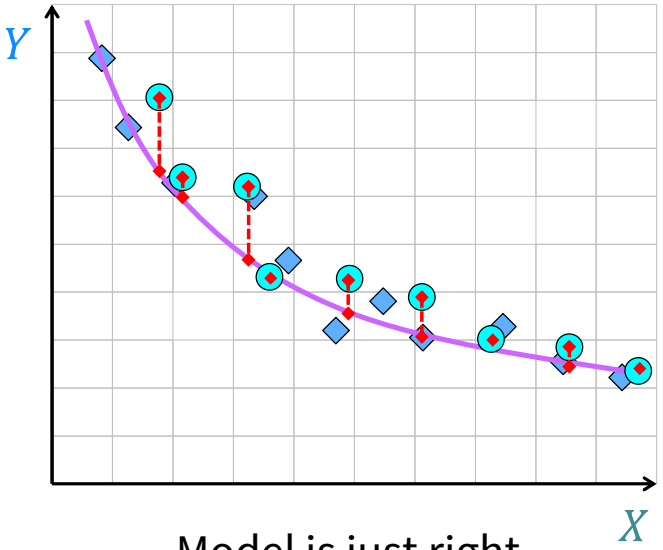
The Fundamental Problem of Overfitting

ML algorithms may overfit the training data

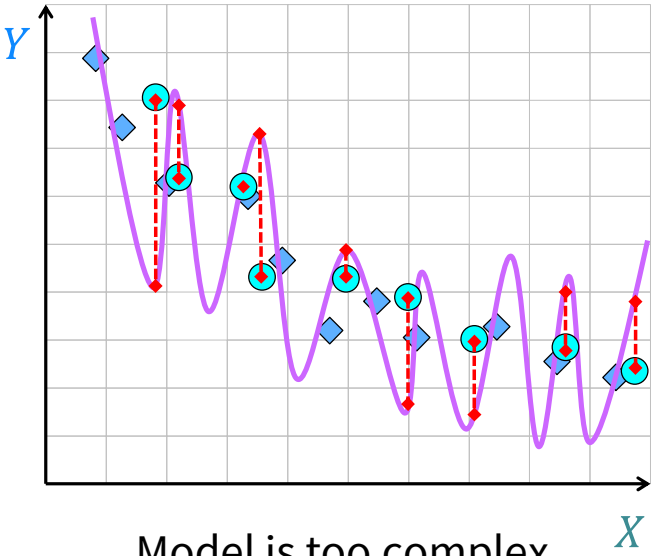
- **The model might be too specific, that is too much geared toward the training data**
- **An overfitting model will shows high forecast error on novel data**
- **An underfitting model has both high forecast errors on novel and high loss on training data**



Model is too simple
(**underfitting**)



Model is just right
(**good solution**)

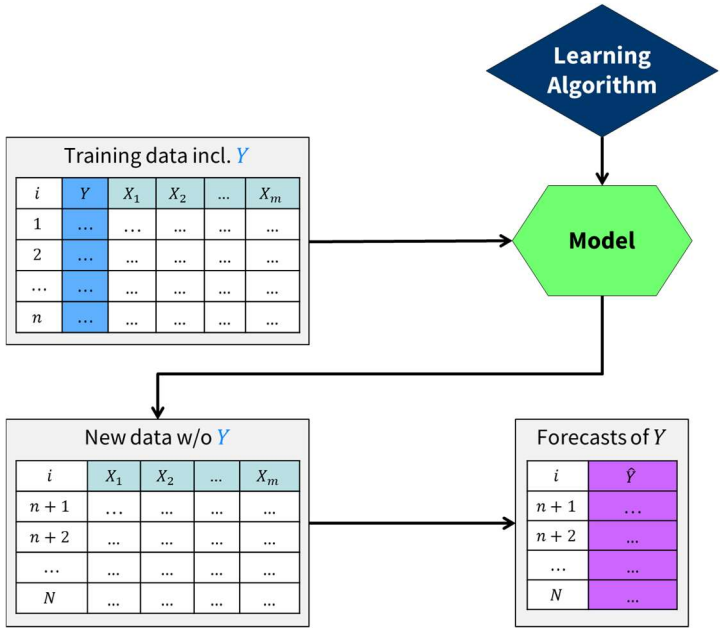


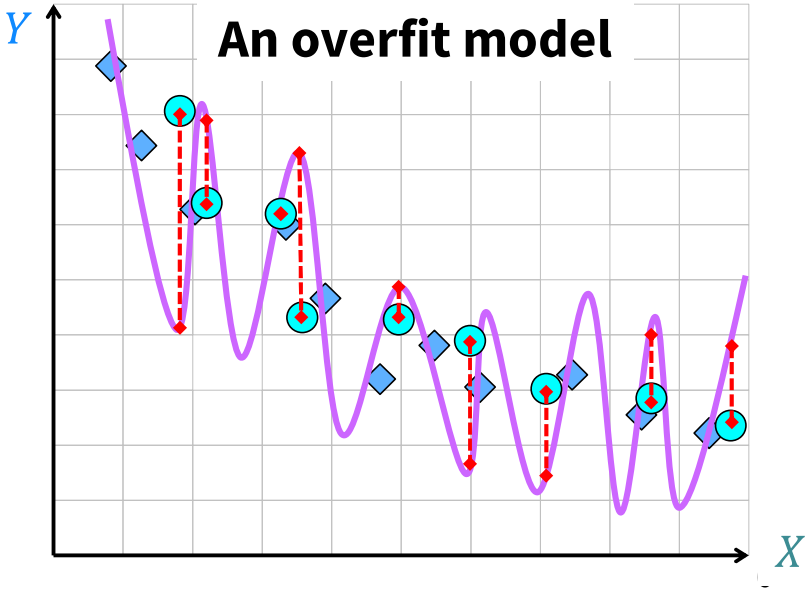
Model is too complex
(**overfitting**)

The Fundamental Problem of Overfitting

But why would a model overfit?

- **Recall how we organize model training**
  - Loss minimization → *mathematically optimal* solution has minimal loss
  - *Optimal* fit of the training data implies the model is very specific and geared toward the training data
- **The training data set is a sample**
  - We assume the sample represents the population well
  - And yet, any sample comprises actual structure…
    - How inputs and outputs related to another
    - Feature-to-target relationship, $E(Y|\boldsymbol{X})$
  - … and random variation (i.e., noise)
- **A model with very low loss might have picked up both, true structure but also the idiosyncratic randomness of the training sample**
- **Applying such a model to new data yields poor results**

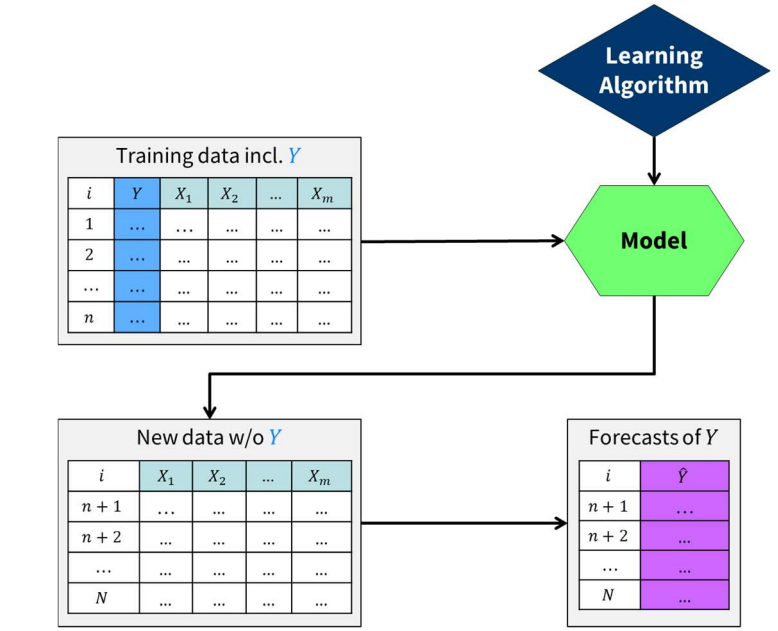


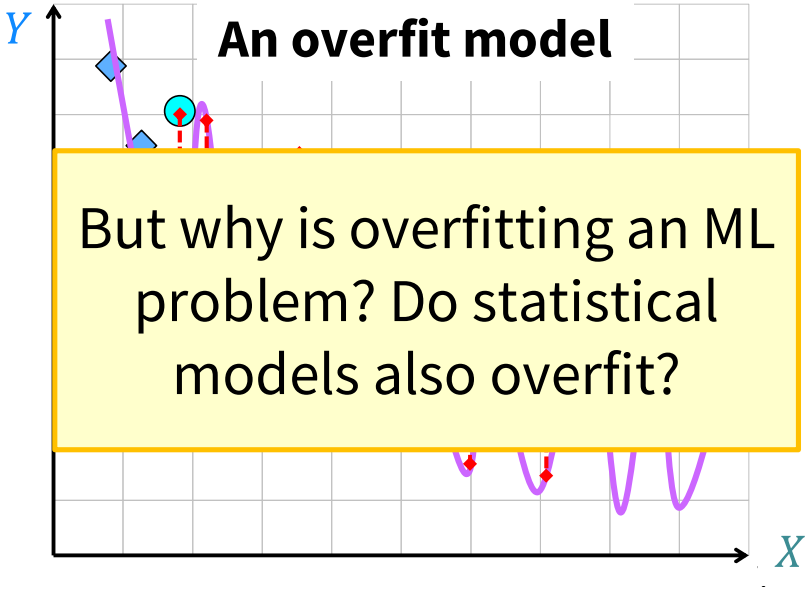


The Fundamental Problem of Overfitting

But why would a model overfit?

- **Recall how we organize model training**
  - Loss minimization → *mathematically optimal* solution has minimal loss
  - *Optimal* fit of the training data implies the model is very specific and geared toward the training data
- **The training data set is a sample**
  - We assume the sample represents the population well
  - And yet, any sample comprises actual structure…
    - How inputs and outputs related to another
    - Feature-to-target relationship, $E(Y|\boldsymbol{X})$
  - … and random variation (i.e., noise)
- **A model with very low loss might have picked up both, true structure but also the idiosyncratic randomness of the training sample**
- **Applying such a model to new data yields poor results**

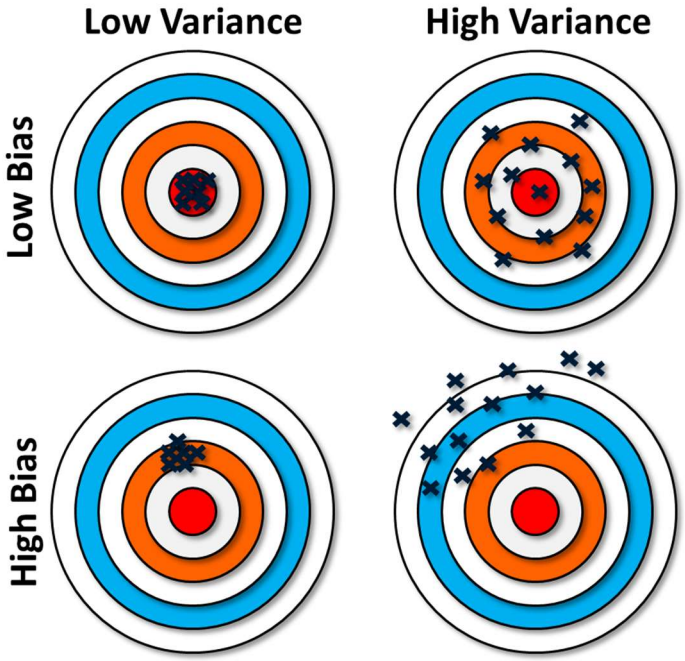




But why is overfitting an ML problem? Do statistical models also overfit?

The Trade-Off Between Bias and Variance

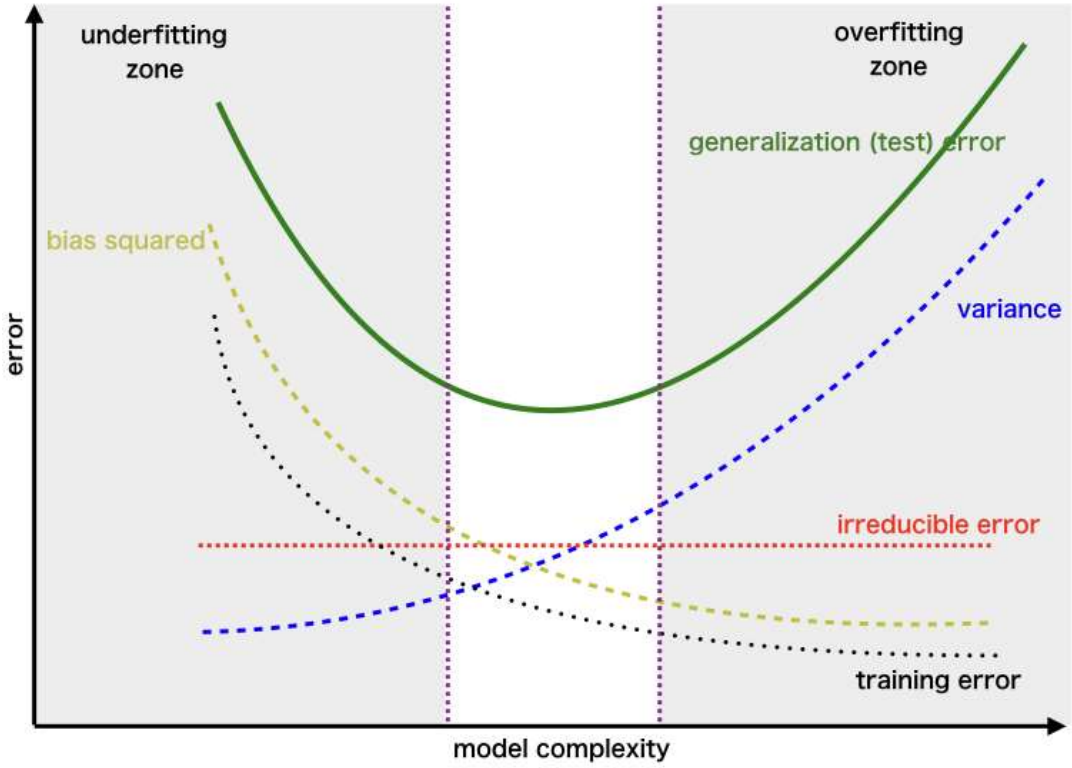
- **We can show that the generalization error of a model is a function to two 'evils'**
  - Generalization error refers to a model's error when predicting any data from the population
  - Not the loss you can measure on the training set
- **Bias**
  - Can the model approximate the true relationship between features and the target?
  - Refers to the expressive power of a learning algorithm
  - The more complex a model the lower its bias
- **Variance**
  - Think of it as sensitivity of a model to its training data
  - How much will the model change with changes in the training data?
  - How much will the model forecasts change?



The Trade-Off Between Bias and Variance

Generalization error is a function to two ‘evils’

- **Let $Y = f(X) + \epsilon$, with random variables**
  - $Y \in \mathbb{R}$, $X \in \mathbb{R}^m$
  - $\epsilon \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$
- **True (aka generalization) error of ML model $\hat{f}(X)$**
  - $E_{\mathcal{D},\epsilon}\left[\left(Y - \hat{f}_{\mathcal{D}}(X)\right)^2\right]$
  - Expectation is taken over different data sets $\mathcal{D} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ sampled from $P(X, Y)$
- Bias and Variance of the ML model
  - $\text{Bias}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X)\right] = E_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X) - f(X)\right]$
  - $Var_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X)\right] = E_{\mathcal{D}}\left[\left(\hat{f}_{\mathcal{D}}(X) - E_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X)\right]\right)^2\right]$
- **Bias-variance decomposition**



$$E_{\mathcal{D},\epsilon}\left[\left(Y - \hat{f}_{\mathcal{D}}(X)\right)^2\right] = \left(\text{Bias}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X)\right]\right)^2 + \text{Var}_{\mathcal{D}}\left[\hat{f}_{\mathcal{D}}(X)\right] + \sigma^2$$

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

Bias-Variance Trade-Off and Overfitting

- **Simple models**
  - High bias
  - Low variance
- **Complex models**
  - Low bias
  - High variance
- **Much of supervised ML is about finding a good compromise**
- **Common paradigm in ML**
  - Use an advanced, complex model
  - Manage / control complexity somehow

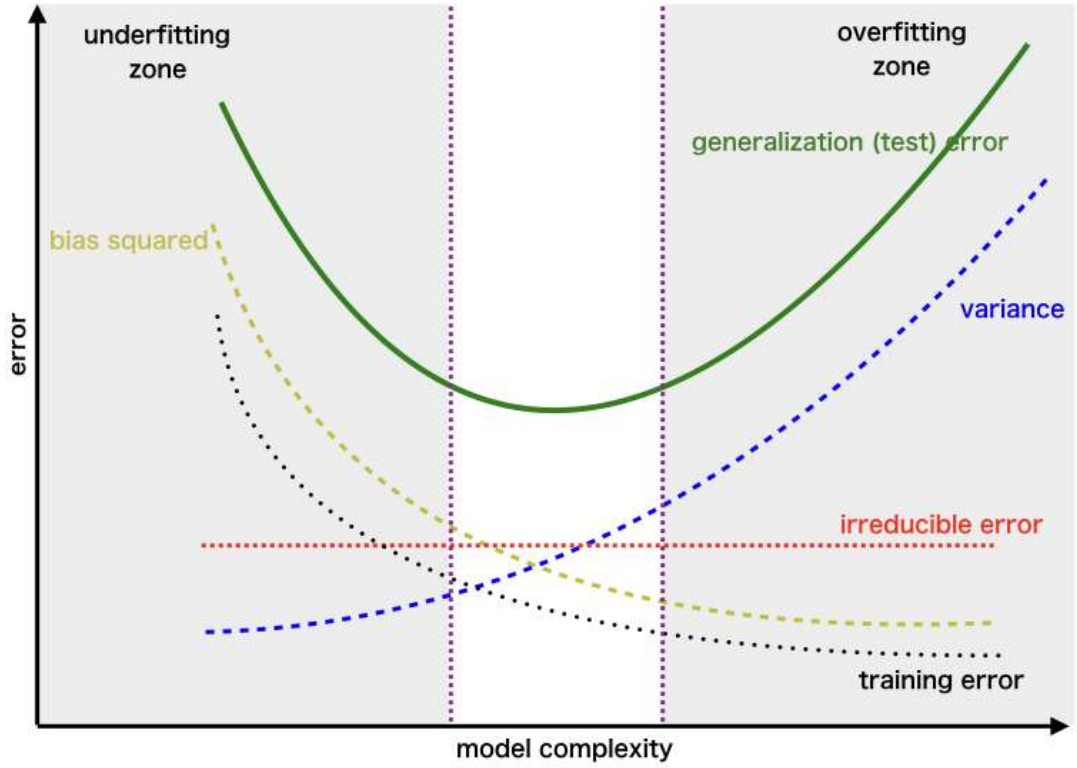
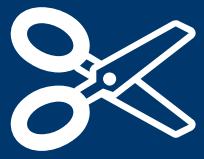


Image source: https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9



Regularization

Idea, connection to bias-variance trade-off, implementation

Regularization

- **Regularization revises practices to develop models**
  - Do not focus on training error alone rather balance between two conflicting objectives
  - Goal 1: **low training error (i.e., low bias)**
  - Goal 2: **low complexity (i.e., low variance)**
- **Complex prediction models …**
  - Display low bias but high variance
  - Are prone to overfit the training set
- **Introducing bias can, therefore, …**
  - Help prevent overfitting
  - Reduce generalization error
- **Regularization involves …**
  - Penalizing model complexity
  - Introducing bias to decrease variance, and the generalization error

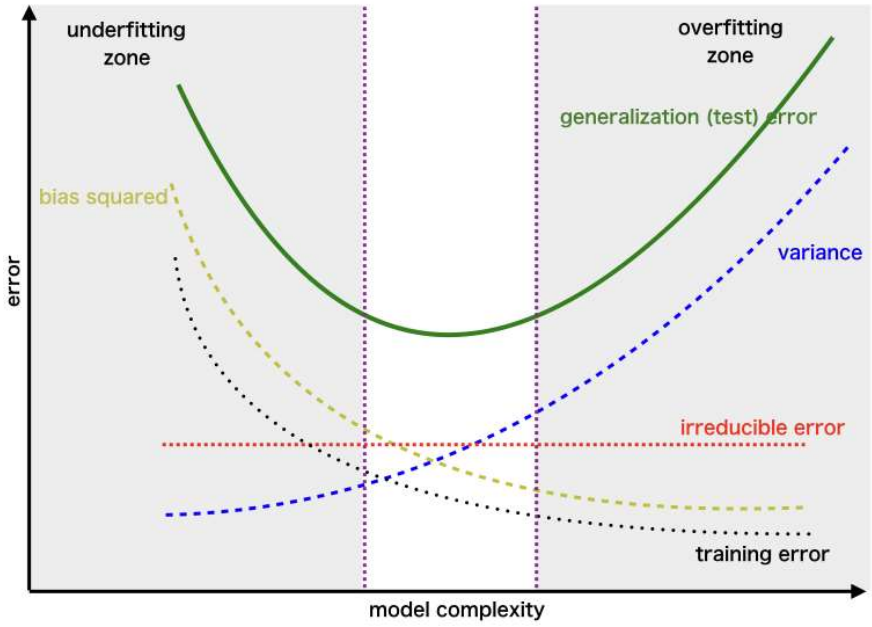
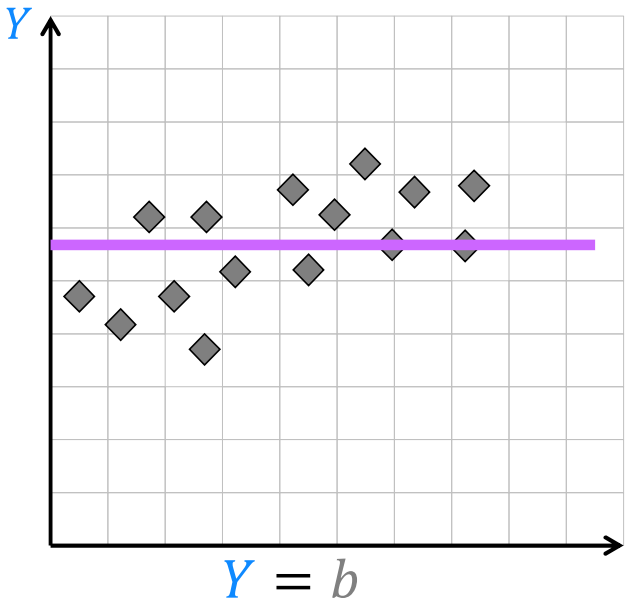


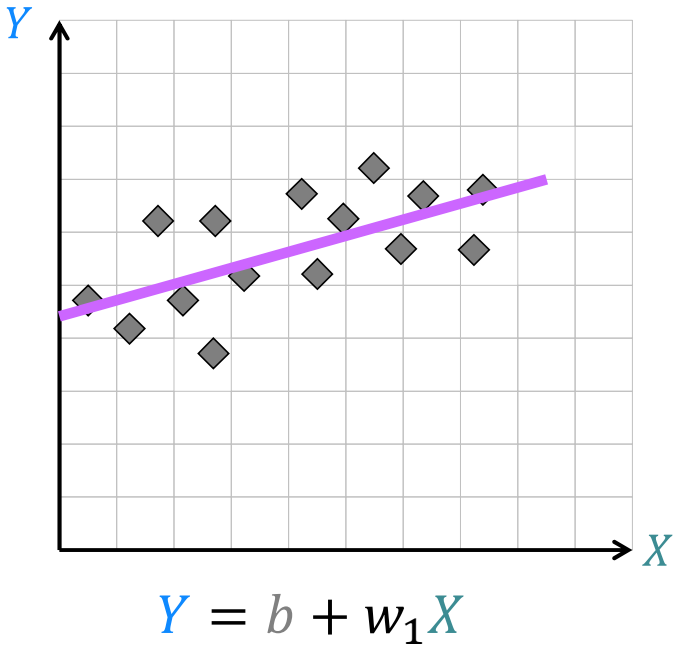
Image source: G. Papachristoudis (2019): https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9

Measuring Model Complexity

Motivating example for regression models

- **Approaches toward measuring complexity vary across learning algorithms**
- **Consider for example univariate linear regression**





Which model is simpler?

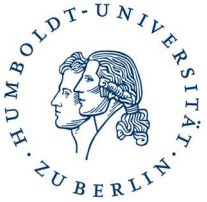
Implementing Regularization

Common complexity penalties for regression-type models

- **LASSO penalty** $L_1(\boldsymbol{w}) = \sum_{j=1}^{m} |w_j|$
- **Ridge penalty** $L_2(\boldsymbol{w}) = \sum_{j=1}^{m} w_j^2$
- **Considerations on penalty choice**
  - LASSO complicates model estimation but gives sparser models
  - Ridge imposes stronger penalty on (very) large coefficients
- **Elastic net penalty** $L_{enet}(\boldsymbol{w}) = \frac{1-\alpha}{2} \sum_{j=1}^{m} w_j^2 + \alpha \sum_{j=1}^{m} |w_j|$
  - With α a mixing parameter between ridge ($\alpha = 0$) and LASSO ($\alpha = 1$)
  - Needs additional tuning by the data scientist

Implementing Regularization

Linear regression revisited



- **Model formulation: the target is a linear, additive function of the features**

$$Y = b + \sum_{j=1}^{m} w_j X_j + \epsilon$$

- **Loss function: least squares loss**

$$J(b, \boldsymbol{w}) = \sum_{i=1}^{n} (\epsilon_i)^2 = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n} \left(Y_i - \left(b + \sum_{j=1}^{m} w_j X_{ij} \right) \right)^2$$

- **Model estimation: minimize the loss function with respect to free model parameters**

$$\hat{\boldsymbol{w}} \leftarrow \operatorname{argmin}_{\boldsymbol{w}} J(b, \boldsymbol{w})$$

Implementing Regularization

Logistic regression revisited

- **Model formulation: model log-odds ratio as linear function of the features**

$$\log\left(\frac{p(Y=1|\boldsymbol{X})}{1-p(Y=1|\boldsymbol{X})}\right) = b + \sum_{j=1}^{m} w_j X_j$$

- **Loss function: negative of the log-likelihood function (aka binary cross-entropy)**

$$J(b,\boldsymbol{w}) = -\left(\sum_{i=1}^{n} Y_i \log\big(p(Y_i=1|\boldsymbol{X}_i)\big) + (1-Y_i)\log\big(1-p(Y_i=1|\boldsymbol{X}_i)\big)\right)$$

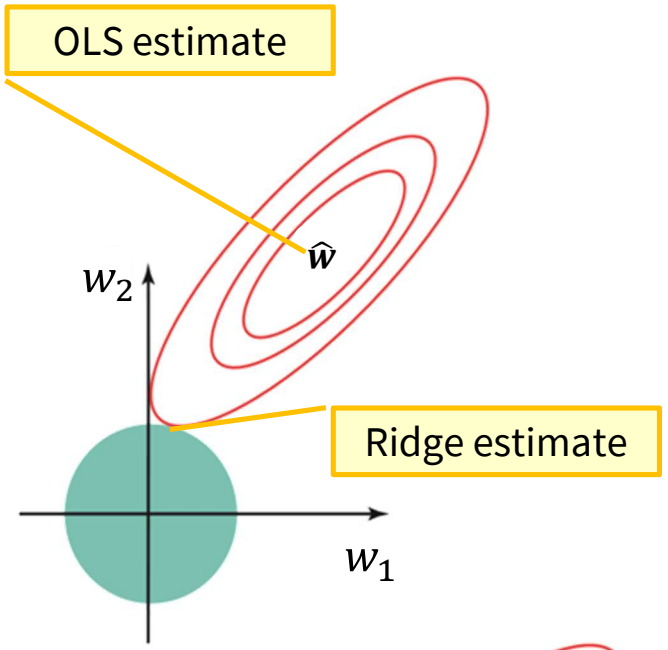
- **Model estimation: minimize the loss function with respect to coefficients**

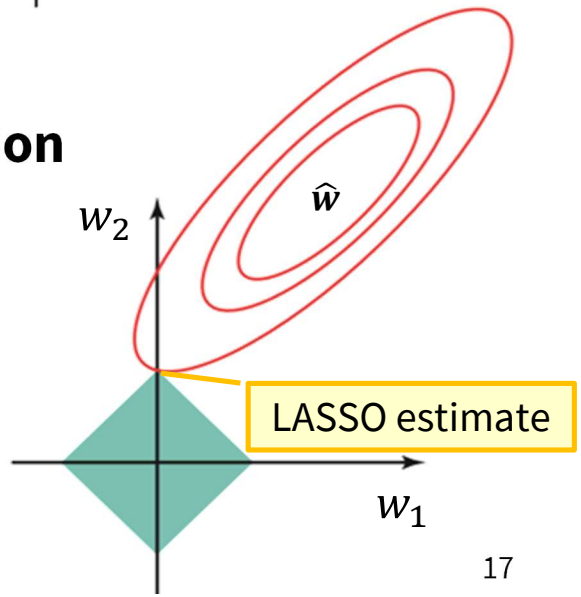
$$\hat{\boldsymbol{w}} \leftarrow \min_{\boldsymbol{w}} J(b,\boldsymbol{w})$$

Regularized Regression

Extent loss function by adding a complexity penalty

- **Ridge penalty**
  - L2 norm of the features $L_2(\boldsymbol{w}) = \|\boldsymbol{w}\|_2 = \sum_{j=1}^{m} w_j^2$
  - Loss function with ridge penalty $\mathcal{L}^{ridge}(\boldsymbol{w}) = J(b, \boldsymbol{w}) + \lambda\, L_2(\boldsymbol{w})$
- **LASSO penalty**
  - L2 norm of the features $L_1(\boldsymbol{w}) = \|\boldsymbol{w}\|_1 = \sum_{j=1}^{m} |w_j|$
  - Loss function with LASSO penalty $\mathcal{L}^{lasso}(\boldsymbol{w}) = J(b, \boldsymbol{w}) + \lambda\, L_1(\boldsymbol{w})$
- **Additional (meta-)parameter λ controls degree of regularization**
  - $\lambda \to \infty$ ➔ elements of $\boldsymbol{w}$ shrink toward zero
  - $\lambda \to 0$ ➔ recovers original, non-regularized model
  - Finding suitable settings for λ requires tuning





Regularization Introduces Bias to Reduce Variance

- **Consider a regression model with many features**
- **Due to the many featuers, the model could have low bias but, for example due to multicollinearity, high variance**
- **Using regularization, we shrink the effect of the feature**
  - Thereby, we reduce the variance of the model,
  - at the cost of increasing its bias
  - This might achieve a more favorable trade-off and a model with lower generalization error

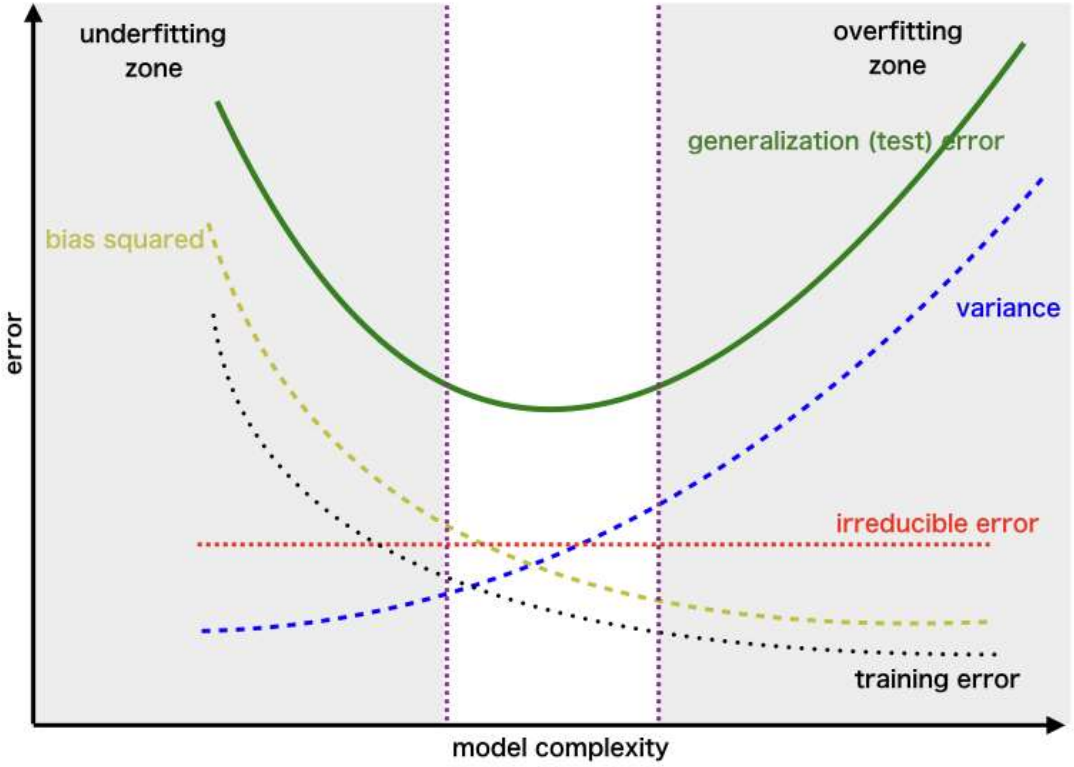
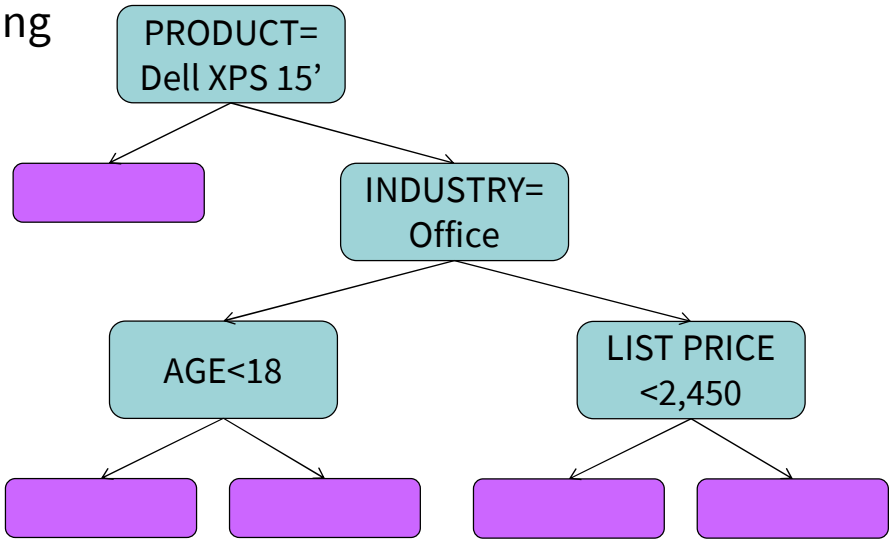


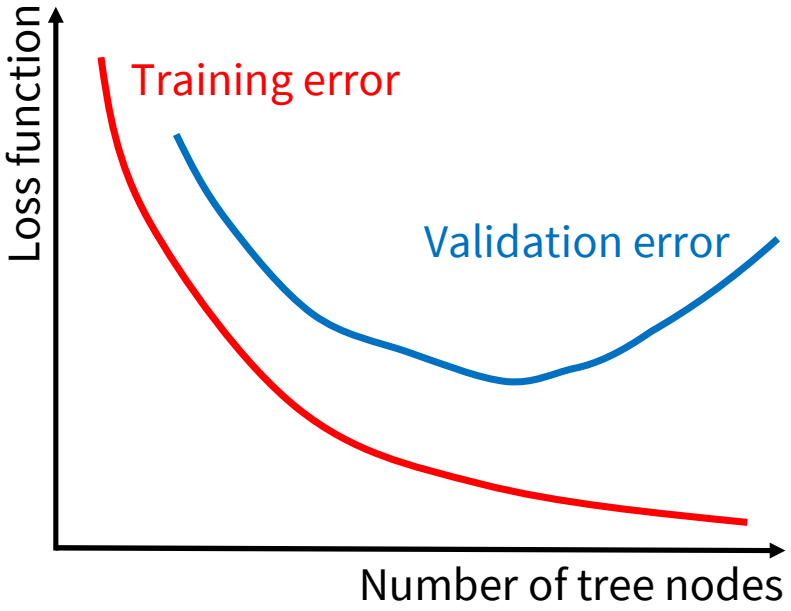
Image source: https://towardsdatascience.com/the-bias-variance-tradeoff-8818f41e39e9

Decision Tree Pruning Revisited

Pruning as a form of regularization

- **Complexity of a tree-based model often measured as number of terminal nodes**
- **Tree pruning is, therefore, also a form of regularization**
  - Pre-Pruning (e.g., max tree depth, min no samples to branch a node, min loss reduction)
  - Post Pruning (grow full tree, then merge leaf nodes ex post)
  - Early stopping



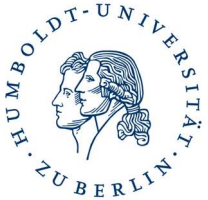


Model Selection

Search strategies and process perspective

Model Selection

Tuning of algorithmic hyperparameters

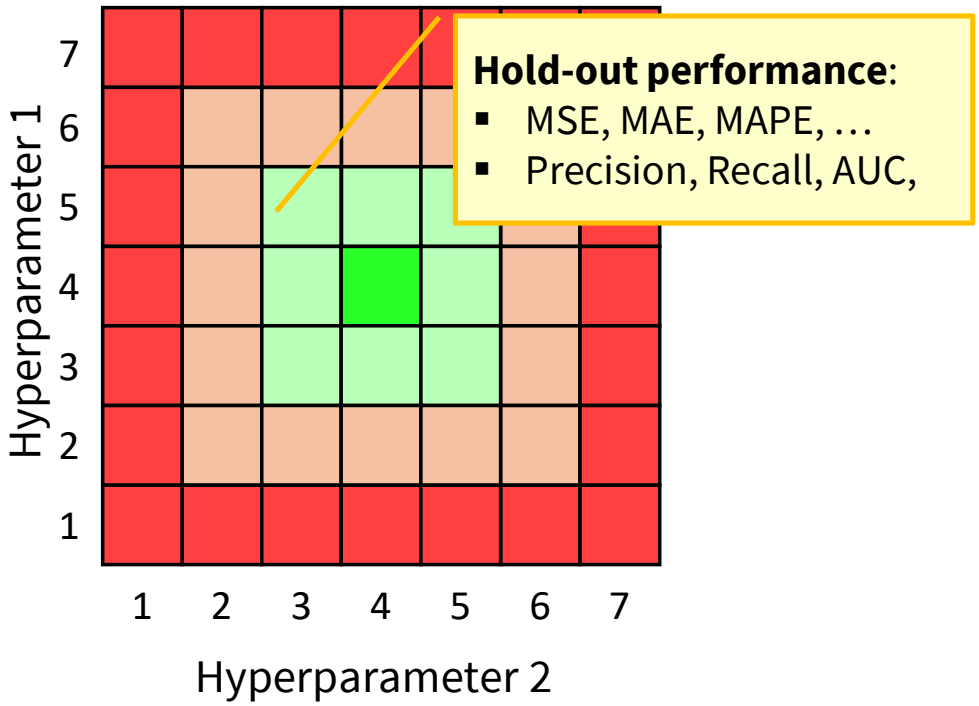


- **Learning algorithms offer hyperparameters (aka meta-parameters)**
  - Facilitate adapting the model to a given data set
  - Facilitate controlling over-fitting / managing the trade-off between bias and variance
  - Need to be set by the data scientist
  - Regularization parameter λ , max depth of a decision tree, etc.
- **Similar to feature selection (in regression modeling)**
  - Manually decide which features to use in a model
  - Try out candidate settings using heuristic search (forward/backward, stagewise regression)
- **How to take corresponding decisions?**
  - Default settings / rules of thumb (not a good idea!)
  - Experience (may work, may fail as well)
  - Empirically, in a model selection process (common practice)

Grid Search

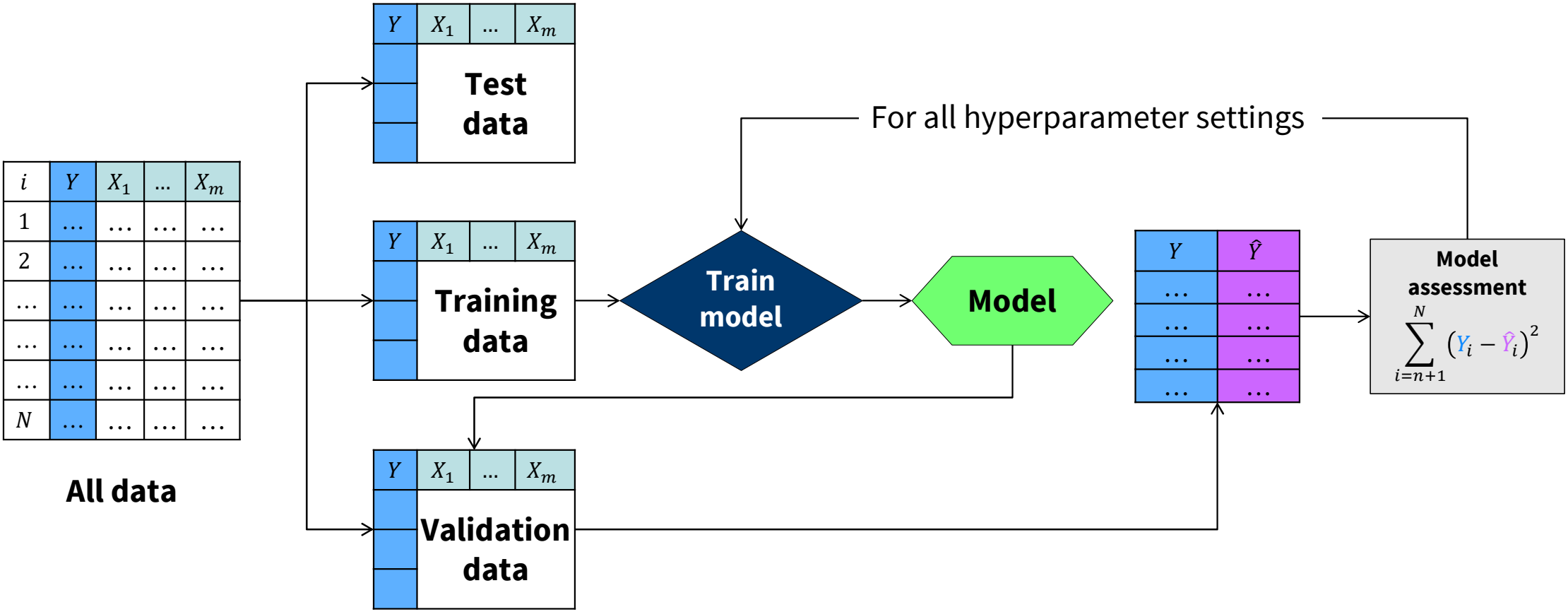
A versatile approach toward model selection

- **Fully enumerative search through all possible combinations of candidate hyperparameter settings**
- **Algorithm**
  - Define candidate range for each hyperparameter
  - Enumerate combinations of candidate values
  - Train model with given configuration
  - Assess model performance on hold-out data
  - Repeat with next configuration
- **Magnify grid resolution in promising regions of the search space**



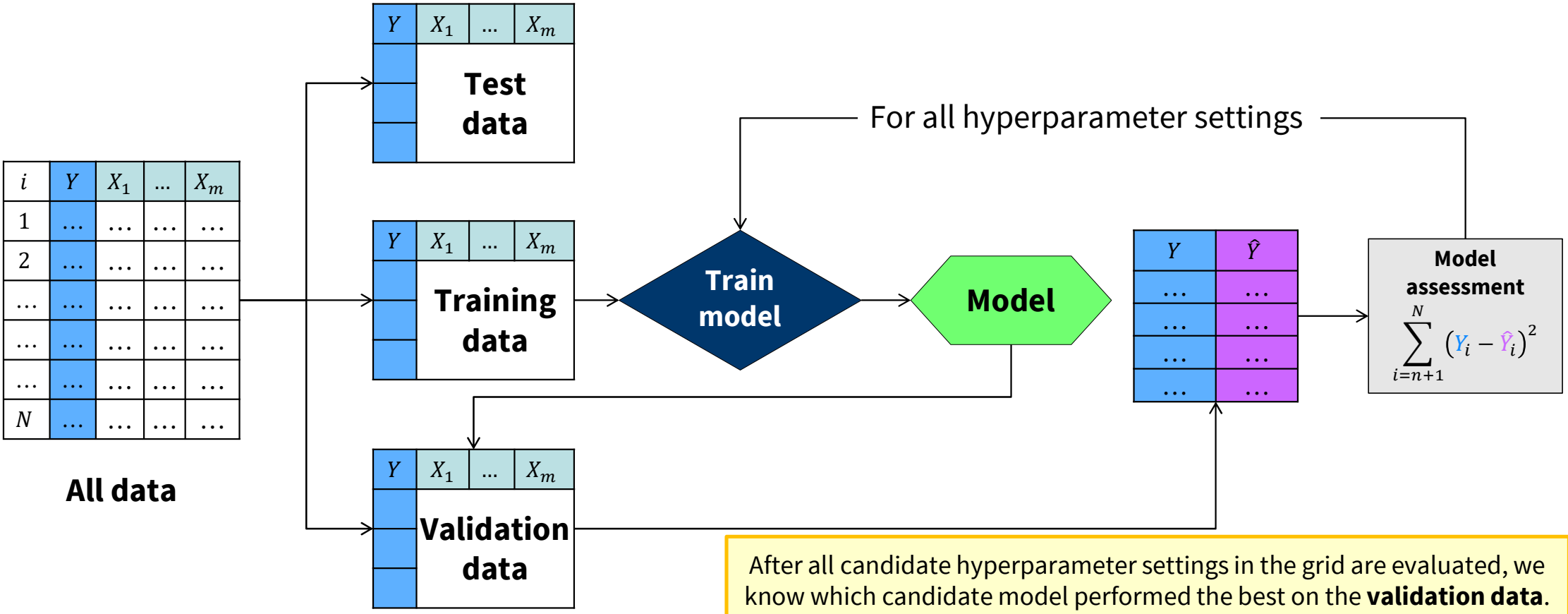
Generalized Holdout Method

Measuring hyperparameter suitability as forecast accuracy needs ‘fresh’ data



Generalized Holdout Method

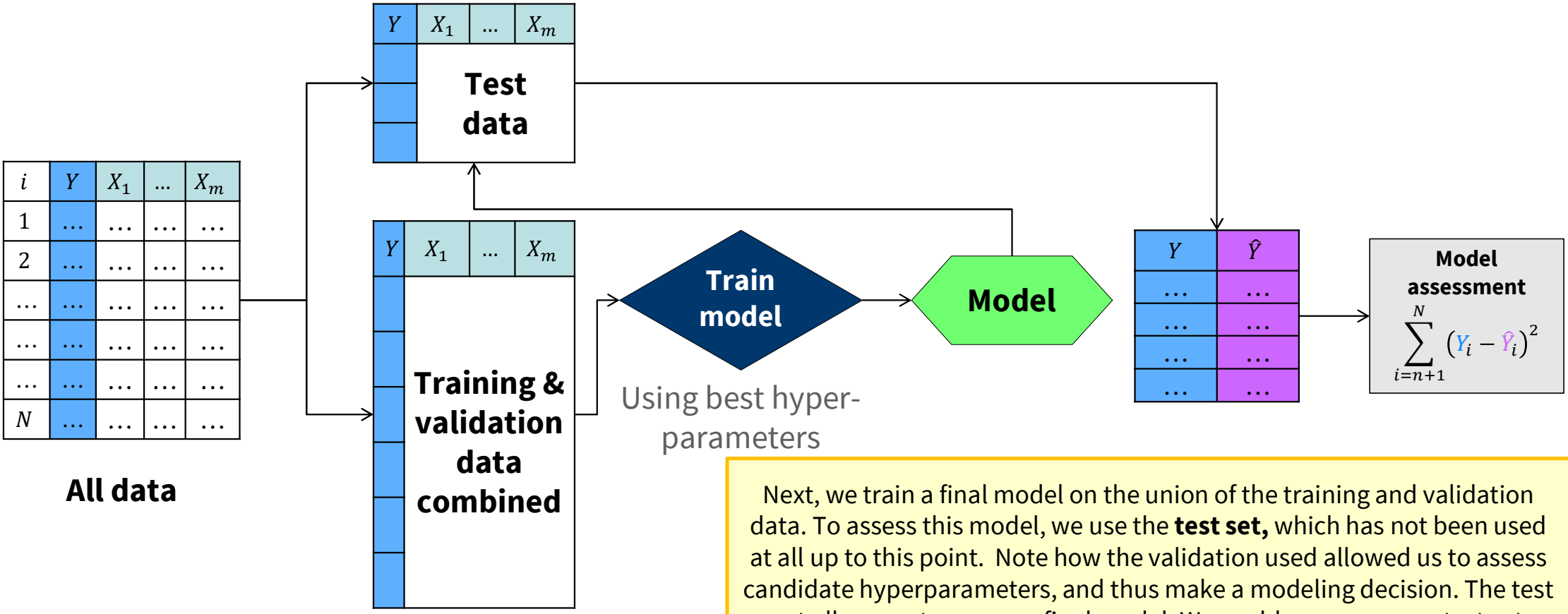
Measuring hyperparameter suitability as forecast accuracy needs ‘fresh’ data



After all candidate hyperparameter settings in the grid are evaluated, we know which candidate model performed the best on the **validation data**. We use the corresponding hyperparameter value(s) for our final model. The hyperparameter tuning problem has been solved.

Generalized Holdout Method

Once we found suitable hyperparameters, we train & assess our final model

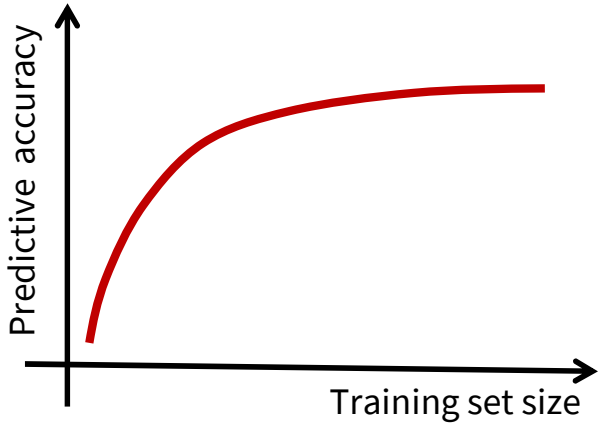


Next, we train a final model on the union of the training and validation data. To assess this model, we use the **test set,** which has not been used at all up to this point. Note how the validation used allowed us to assess candidate hyperparameters, and thus make a modeling decision. The test set allows us to assess a final model. We could now compare test set accuracy to our expectations or alternative forecasting models.

Model Selection Summarized

Tuning of algorithmic hyperparameters using grid search and holdout data

- **Adjust ML algorithm to the given data set and control overfitting**
- **Model selection is costly**
  - Iterative training and evaluation of different candidate models
    - As many as candidate hyperparameter combinations in the grid
    - Even more if using cross-validation instead of train, validation, test split
    - Some ML algorithms exhibit many hyperparameters (e.g., XGBoost, see later)
  - Careful exploration of parameter space computationally challenging
- **Practical recommendation**
  - Check whether you can reduce the amount of data during model selection
  - Does hyperparameter performance depend on the size of the training sample?
  - If not (too much) you can down-sample the training set, determine best hyperparameters, and build a model with best hyperparameters on the full training set can give a major speed-up
  - Can start from a learning curve analysis (Perlich et al., 2003) to determine how much down-sampling is possible



A Cautionary Closing

Mind the Shortcut

Fallacies of the training/test set approach

- **Shortcut solutions in an ML context**
  - Model relies on simple characteristics of the data
  - Model does not learn the true essence of the relationship between features and the target
- **Problem with shortcut solutions**
  - Shortcut features facilitate accurate predictions for a specific data set
  - Splitting the data into training and test set does not help, as test set performance is still high
  - On novel data, however, the shortcut might no longer be accessible
  - This would break the model (i.e., poor generalization)

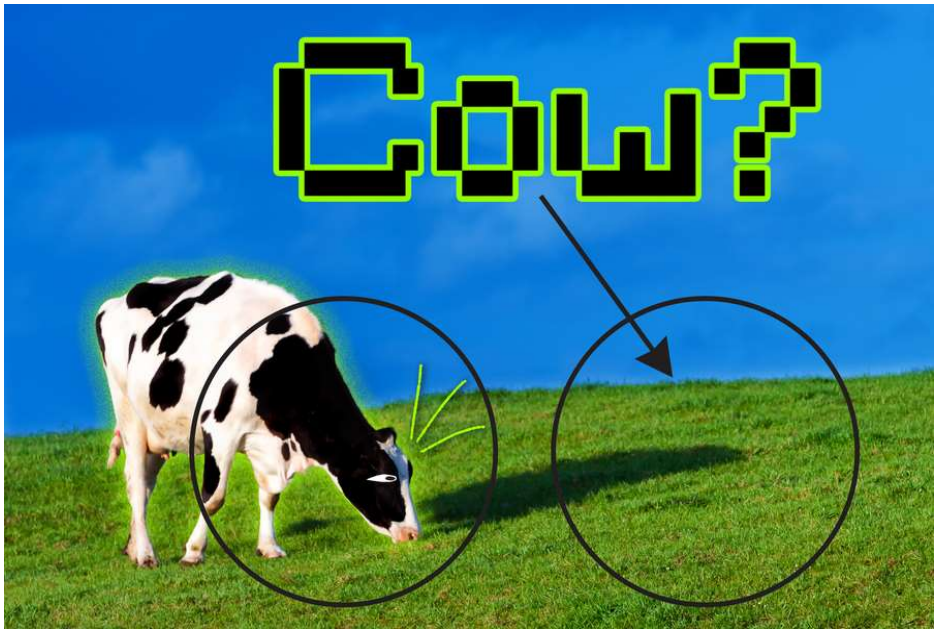


Image source:
MIT News https://news.mit.edu/2021/shortcut-artificial-intelligence-1102)

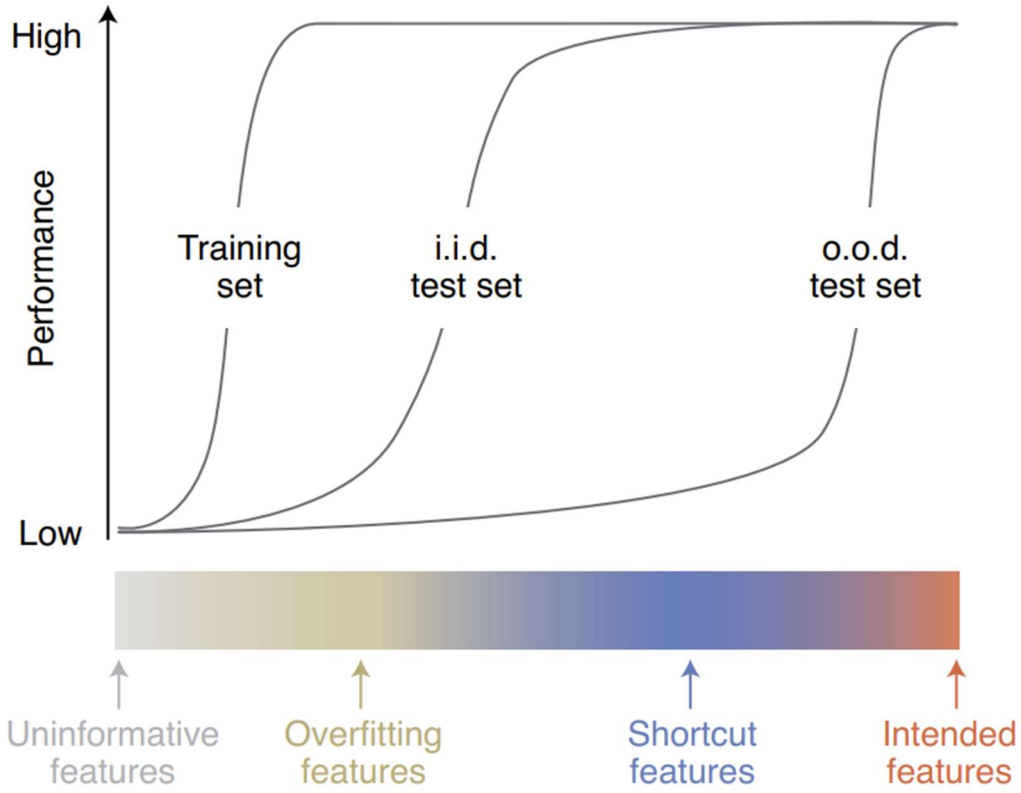
Geirhos, et al. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence, 2(11), 665-673.* https://doi.org/10.1038/s42256-020-00257-z

Note that **i.i.d.** stands for independently and identically distributed. This is the kind of data we obtain from a random train/test set split. Conversely, **o.o.d.** stands for out-of-distribution data.

Mind the Shortcut

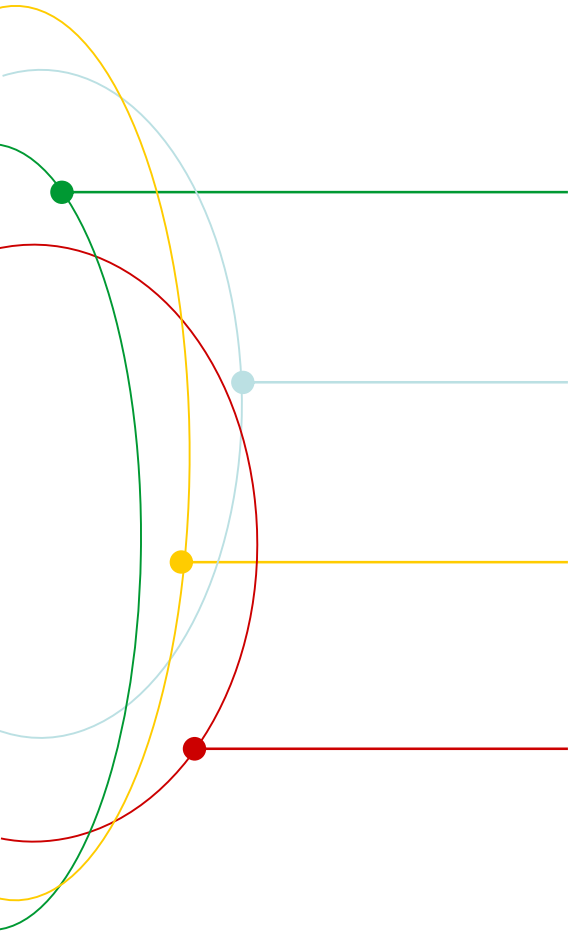
Fallacies of the training/test set approach

- **Shortcut solutions in an ML context**
  - Model relies on simple characteristics of the data
  - Model does not learn the true essence of the relationship between features and the target
- **Problem with shortcut solutions**
  - Shortcut features facilitate accurate predictions for a specific data set
  - Splitting the data into training and test set does not help, as test set performance is still high
  - On novel data, however, the shortcut might no longer be accessible
  - This would break the model (i.e., poor generalization)



Geirhos, et al. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence, 2(11), 665-673.* https://doi.org/10.1038/s42256-020-00257-z

Summary



Learning goals

- Understand overfitting problem
- and its connection to bias and variance

Findings

- Detect overfitting by comparing training to test error
- Complex models display low bias and high variance
- Regularization introduces bias to decrease variance
- Implementing regularization through penalties
- Model selection for tuning algorithmic hyperparameters including regularization penalty

What next

- Python tutorial
- XMAS Break

Literature

- **Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization Journal of Machine Learning Research, 13, 281-305.**
- **Perlich, C., Provost, F., Simonoff, J. S., & Cohen, W. W. (2003). Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research, 4(2), 211-255.**

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

Appendix

Further considerations related to model selection

Model Selection

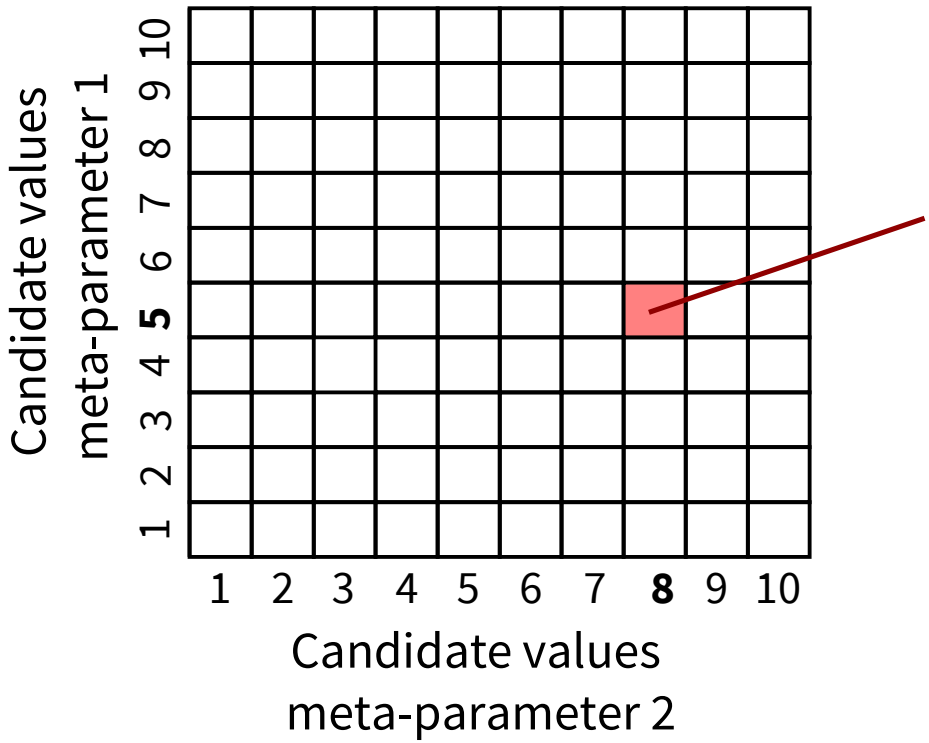
Learning algorithms may exhibit many hyperparameters

- **Regularized logistic regression**
  - Regularization coefficient
  - Two coefficients for elastic net penalty
- **Decision trees**
  - Splitting criterion
  - Max depth
  - Min observations per leaf
  - Magnitude of IG to continue splitting
  - Post-pruning
- **Tree based ensembles**
  - Hyperparameters of the base learner
  - Size of the ensemble
  - …
- **Neural networks**
  - Regularization coefficient, dropout rate
  - No. hidden layers
  - No. hidden nodes
  - Activation function
  - Learning rate, decay schedule
  - Solver
- **Support vector machines**
  - Regularization coefficient
  - Kernel function
  - Parameters of kernel function

Model Selection

Grid search: a versatile approach toward model selection

- **Three step approach**
  - For each meta-parameter,
  - define candidate settings
  - test combinations empirically
- **Example**
  - Two parameters
  - 10 candidate settings each
  - Grid search explores 10*10=100 value combinations



Model Selection

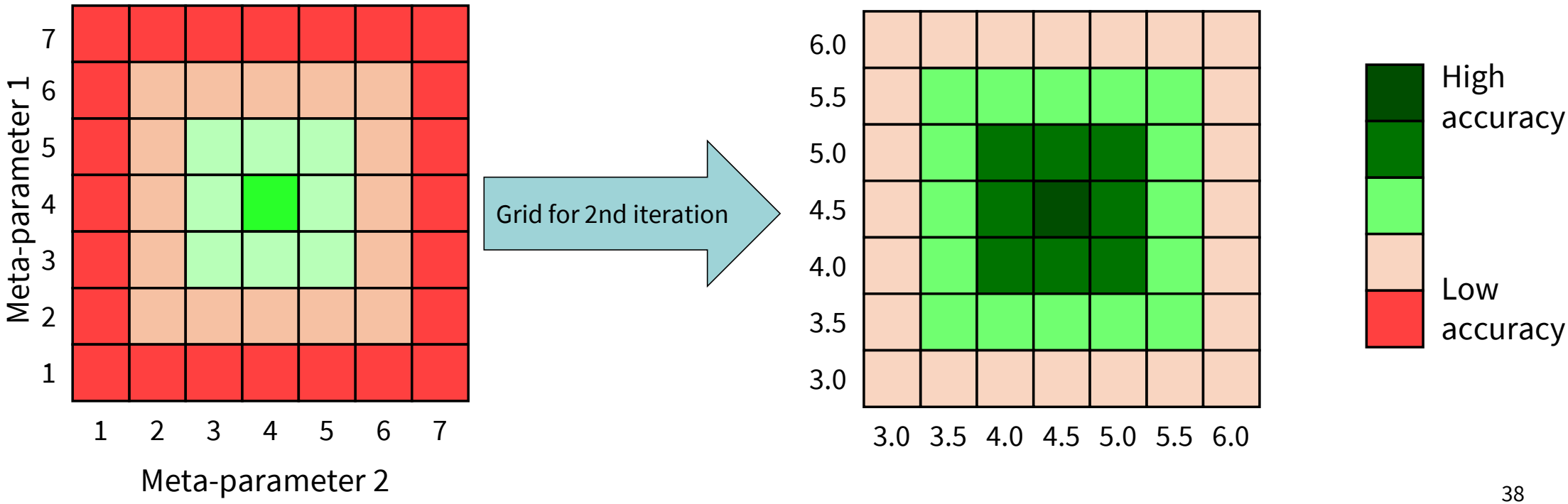
Some paper to learn about candidate parameter settings

- Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble Selection from Libraries of Models. In C. E. Brodley (Ed.), *Proc. of the 21st Intern. Conf. on Machine Learning* (pp. 18-25). Banff, Alberta, Canada: ACM.
- Caruana, R., Munson, A., & Niculescu-Mizil, A. (2006). Getting the Most Out of Ensemble Selection. In *Proc. of the 6th Intern. Conf. on Data Mining* (pp. 828-833). Hong Kong, China: IEEE Computer Society.
- Dejaeger, K., Verbeke, W., Martens, D., & Baesens, B. (2011). Data mining techniques for software effort estimation: A comparative study. *IEEE Transactions on Software Engineering, 38*, 375-397.
- Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). A Practical Guide to Support Vector Classification. In. Taiwan: Department of Computer Science and Information Engineering, National Taiwan University.
- S. Lessmann, M.C. Sung, J.E. Johnson, T. Ma, A new methodology for generating and combining statistical forecasting models to enhance competitive event prediction, *European Journal of Operational Research*, 218(1) (2012) 163-174.
- S. Lessmann, S. Voss, Customer-centric decision support: A benchmarking study of novel versus established classification models, *Business and Information System Engineering*, 2(2) (2010) 79-93.
- S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification models for software defect prediction: A proposed framework and novel findings, *IEEE Transactions on Software Engineering*, 34(4) (2008) 485-496.
- Lessmann, S., Baesens, B., Seow, H.-V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research, 247*, 124-136.
- Lessmann, S., Haupt, J., Coussement, K., & De Bock, K. W. (2019). Targeting customers for profit: An ensemble learning framework to support marketing decision-making. Information Sciences, (doi:10.1016/j.ins.2019.05.027).
- Loterman, G., Brown, I., Martens, D., Mues, C., & Baesens, B. (2012). Benchmarking regression algorithms for loss given default modeling. *International Journal of Forecasting, 28*, 161-170.
- Partalas, I., Tsoumakas, G., & Vlahavas, I. (2010). An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning, 81*, 257-282.
- Van Gestel, T., Suykens, J. A. K., Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., De Moor, B., & Vandewalle, J. (2004). Benchmarking least squares support vector machine classifiers. *Machine Learning, 54*, 5-32
- Verbeke, W., Dejaeger, K., Martens, D., Hur, J., & Baesens, B. (2012). New insights into churn prediction in the telecommunication sector: A profit driven data mining approach. *European Journal of Operational Research, 218*, 211-229.

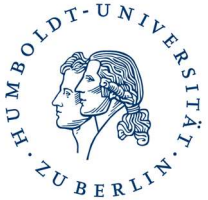
Repeated Grid Search

Repeat grid-search zooming in on promising search areas

- **Magnify resolution of candidate settings in promising areas**
- **Consider two to three iterations and/or trace degree of improvement**

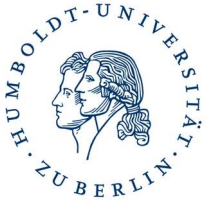


Model Selection Approaches Beyond Grid Search



- **Other search strategies**
  - Random search (popular for deep neural networks, see Bergstra & Bengio, 2012)
  - Meta-heuristics and evolutionary algorithms (genetic algorithms, evolution strategies, particle swarm optimization, harmony search, etc.)
- **Promise autonomous, self-adaptive hyperparameter tuning**
  - Do not require candidate settings for prediction model hyperparameters to be defined
  - But what about the parameters of the search strategy ???
- **Practical recommendation**
  - Using an advanced search strategy, you trade one tuning problem for another
  - Availability in software packages might also be an issue
  - In most cases, grid search will work well

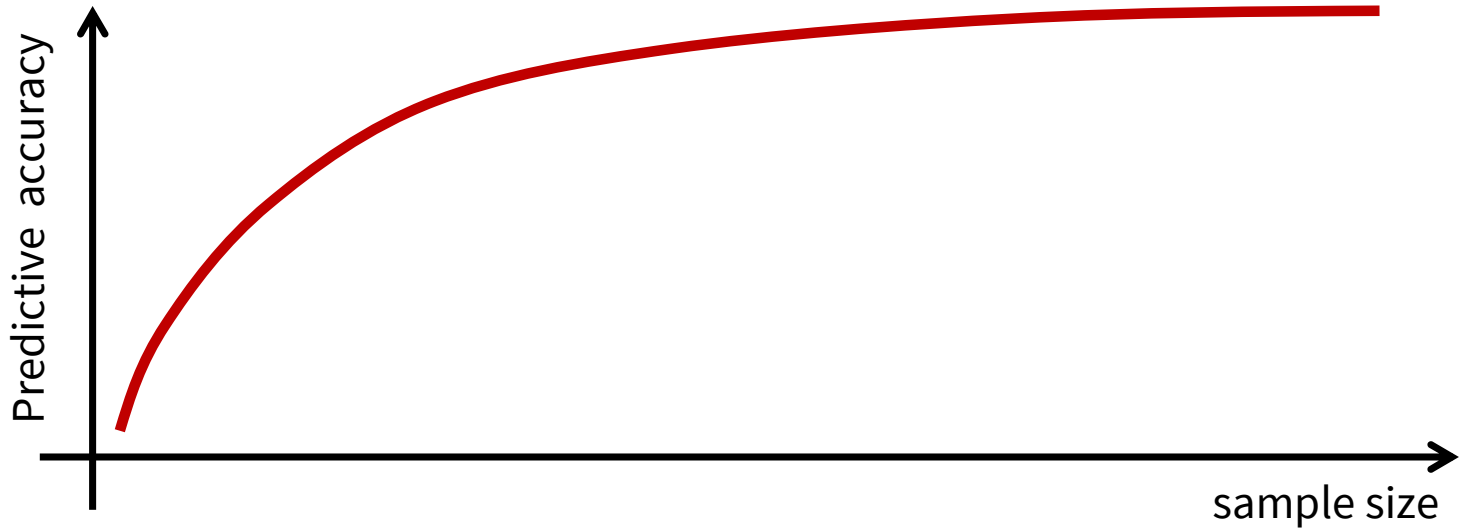
Model Selection Efficiency



- **Model selection is costly**
  - Iterative estimation of different candidate models
  - As many as candidate hyperparameter values in grid-search
  - Careful exploration of parameter space challenging
- **Approaches to increase efficiency**
  - Algorithmic specific strategies (much work on support vector machines; Lessmann & Voß, 2009)
  - Generic approaches
- **Practical recommendation**
  - Consider learning curve-based heuristic
  - Learning curve analysis tells you how much data is needed
  - Carry out model selection with this amount of data might give substantial speed-up
  - Following slides detail this idea

Learning Curve Analysis

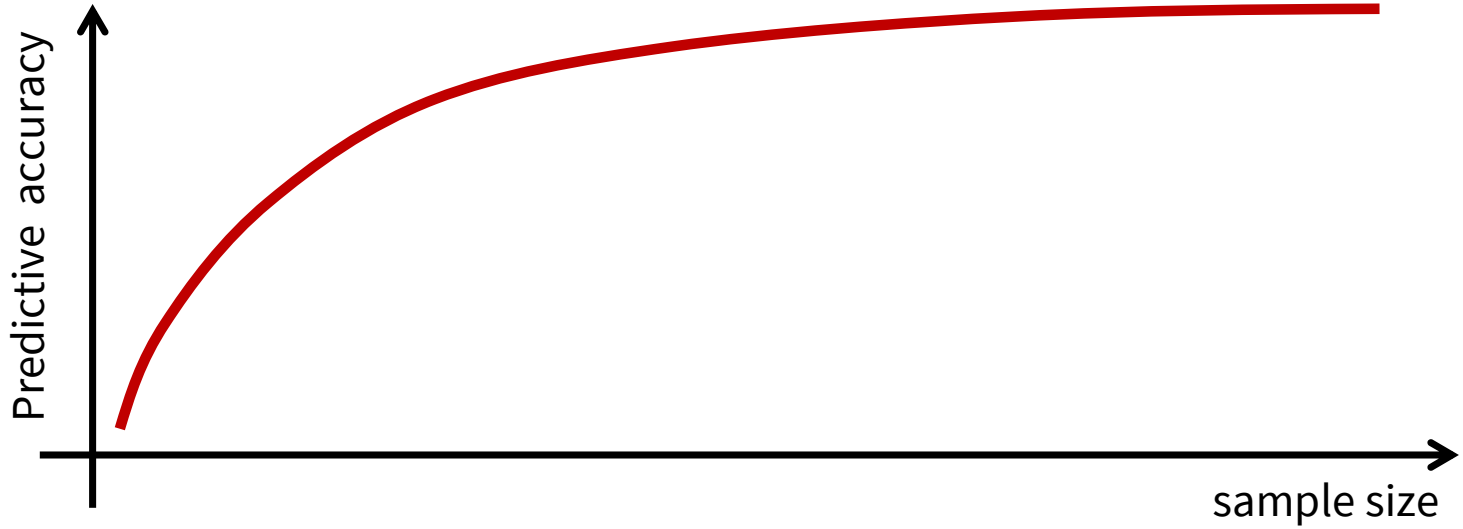
Examines the sensitivity of a model regarding training data size



Learning Curve Analysis

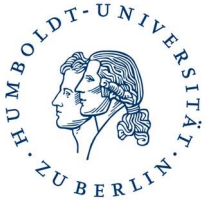
Examines the sensitivity of a model regarding training data size

- **How much data is needed or what is the marginal value of more data**
- **Three-step approach**
  - Draw small sample from your data
  - Estimate and assess model (e.g., split-sample method)
  - Increase samples size and repeat
- **Perlich et al. (2003)**
- **Learning curve will often display a degressive trend**
  - Marginal value of data diminishes
  - Curve offers insight when training has stabilized



Learning Curve Analysis & Model Selection

A heuristic to increase the efficiency of model selection



- **Assumption**
  - Hyperparameter efficacy does not depend on sample size
  - Relaxation: Moderate dependence is still ok
- **Perform learning curve analysis with default hyperparameter values**
- **Find sample size where classifier training stabilizes**
- **Perform model selection using that sample size**
- **Can give substantial speed-up**
  - For many classifiers, training time increases exponentially with sample size
  - Small reduction in sample size facilitates notable speed-up

Learning Curve Analysis & Model Selection

A heuristic to increase the efficiency of model selection