IA|BE Data Science Certificate

Module 1 on Foundations of machine learning in actuarial sciences
Machine learning basic concepts

Katrien Antonio
LRisk - KU Leuven and ASE - University of Amsterdam

October 18, 2022

Acknowledgement

Some of the figures in this presentation are taken from *An Introduction to Statistical Learning, with applications in R* (Springer, 2021, 2nd edition) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Some of the figures in this presentation are taken from *The Elements of Statistical Learning: Data mining, Inference and Prediction* (Springer, 2009) with permission from the authors: T. Hastie, R. Tibshirani and J. Friedman.

Some of the figures in this presentation are taken from *Applied Predictive Modeling* (Springer, 2013) with permission from the authors: M. Kuhn and K. Johnson.

What is predictive modeling?

- Given a response (or outcome) Y and p different predictors $x_1,\ x_2, \ldots, x_p$, we assume

$$\begin{aligned} Y &= f(x_1, x_2, \ldots, x_p) + \epsilon \\ &= f(\boldsymbol{x}) + \epsilon, \end{aligned}$$

  with f some fixed, but unknown function of $x_1, \ldots, x_p$ (the predictors, independent variables, features or just variables) and ϵ a random error term.

- f represents the systematic information that $\boldsymbol{x}$ provides about Y.

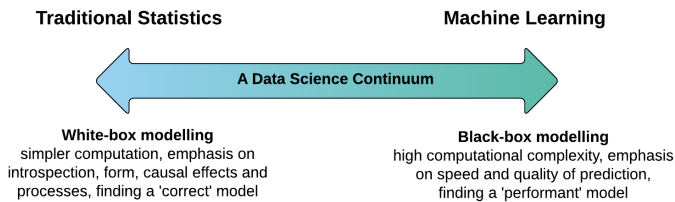
  [The Signal vs the Noise.]

- Predictive modeling refers to a set of approaches to construct $\hat{f}(.)$, the estimate for $f(.)$.

Why building predictive models?

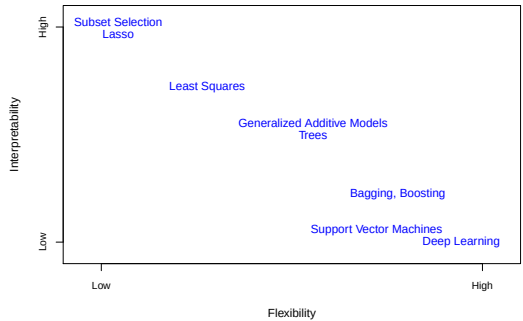
Why estimate f?

- for prediction
- for inference

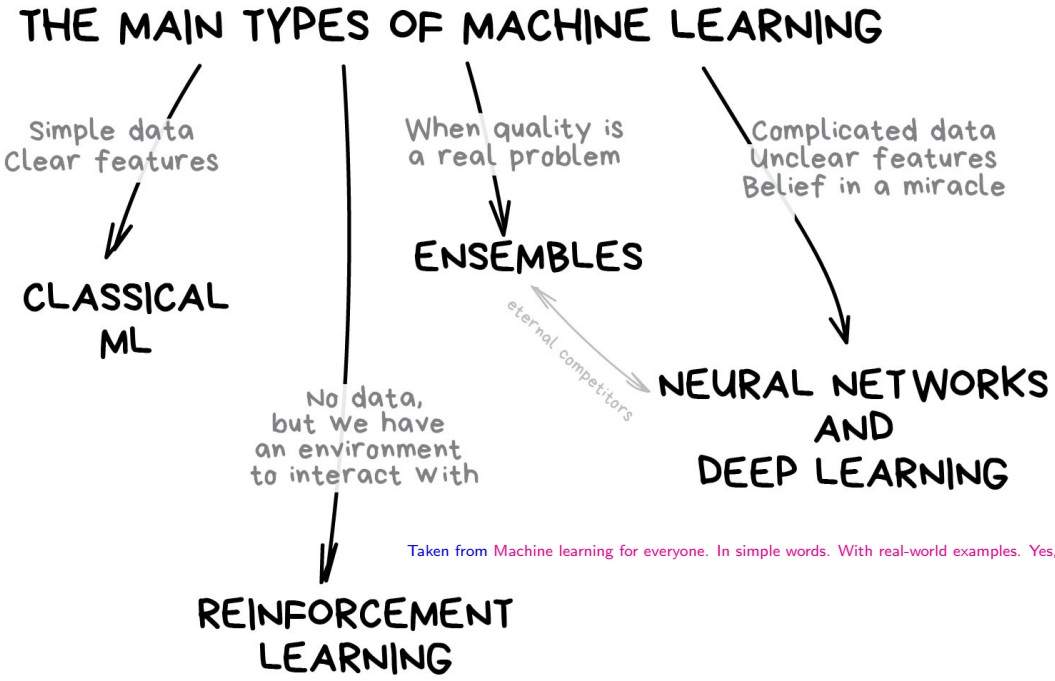


Flexibility vs interpretability

Trade off between flexibility of predictive models and model interpretability!



(Taken from James et al., 2021, An introduction to statistical learning.)



Taken from Machine learning for everyone. In simple words. With real-world examples. Yes, again.

Prediction error

- Consider $Y = f(\boldsymbol{x}) + \epsilon$ and fit a model $\hat{f}(.)$ from given (training) data $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$.
- Assume $\hat{f}(.)$ and $\boldsymbol{x}$ given, then

$$\begin{aligned} E\left[(Y - \hat{Y})^2\right] &= E\left[(f(\boldsymbol{x}) + \epsilon - \hat{f}(\boldsymbol{x}))^2\right]. \\ &= \underbrace{[f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x})]^2}_{\text{reducible}} + \underbrace{\text{Var}(\epsilon)}_{\text{irreducible}}. \end{aligned}$$

- Reducible error: $\hat{f}$ is not a perfect estimate for f, but potentially the accuracy of $\hat{f}$ can be improved (but do mind overfitting!).
- Irreducible error: Y is also a function of ϵ, which can not be predicted using $\boldsymbol{x}$.

Loss functions

- In statistics loss functions are metrics that compare fitted or predicted values to actual outcomes.
- These are minimized when fitting (or training) a model, e.g.
  - calibrate regression parameters in a (G)LM by minimizing the negative log likelihood
  - calibrate weights and bias terms in a neural net by minimizing the loss function.
- These metrics are also used to evaluate model or predictive accuracy on unseen, test data and to compare the performance of different models.

Loss functions

Some (classic) examples:

- mean squared error (MSE) in regression

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\boldsymbol{x}_i))^2$$

- cross-entropy or log loss in classification, with $\hat{p}_i = \frac{1}{1+e^{-\hat{f}(\boldsymbol{x}_i)}}$

$$-\frac{1}{n}\sum_{i=1}^{n}(y_i \cdot \log(\hat{p}_i) + (1-y_i)\cdot \log(1-\hat{p}_i)).$$

Many others exist (e.g. deviance in Poisson or gamma regression) $\sim$ tailor to predictive problem at hand!

Overfitting

- When a method yields a small training error, but a large error on new, unseen data

  overfitting results!

- Our learning procedure is working too hard to find patterns in the training data.
- The test or generalization error will then be (very) large, because the supposed patterns in the training data can not be generalized to unseen, test data!

Bias-variance trade off

- Take a closer look at the test MSE, with $(y_0, \boldsymbol{x}_0)$ a new observation:

$$E\left[(Y_0 - \hat{f}(x_0))^2\right] = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

- To minimize the expected test error, low variance and low bias are necessary.
- But these are two competing sources of error:
  - when bias $\searrow$ then typically variance $\nearrow$
  - when variance $\searrow$ then typically bias $\nearrow$.
- Both are a function of the model's complexity and flexibility!

Bias and variance of a learning method

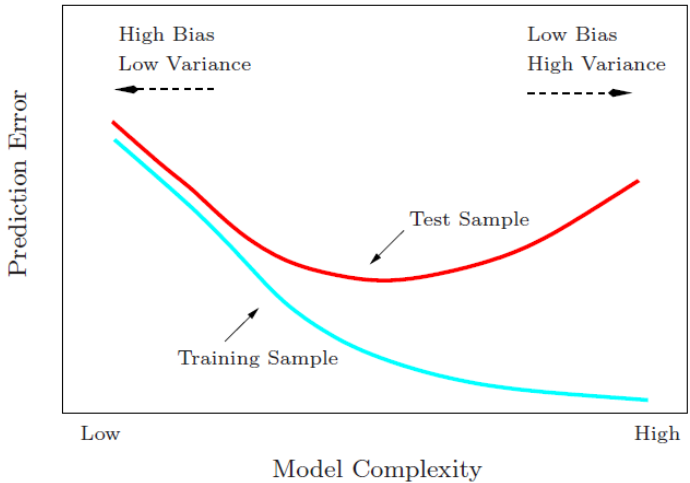
- Variance or (loosely) $\text{Var}[\hat{f}(.)]$
  - the amount by which $\hat{f}(.)$ would change when calibrated on a different training data set, say $\mathcal{D}$

$$\text{Var}_{\mathcal{D}}[\hat{f}(x;\mathcal{D})] = E_{\mathcal{D}}[(\text{E}_{\mathcal{D}}[\hat{f}(x;\mathcal{D})] - \hat{f}(x;\mathcal{D}))^2]$$

- Bias or (loosely) $E[\hat{f}(.)] - f(.)$
  - the error that is introduced by approximating a real-life problem by a (much simpler) model

$$\text{Bias}_{\mathcal{D}}[\hat{f}(x;\mathcal{D})] = \text{E}_{\mathcal{D}}[\hat{f}(x;\mathcal{D})] - f(x).$$

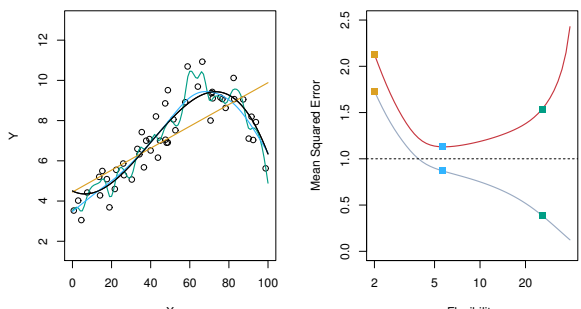
U-shape of test error



(Taken from James et al., 2021, An introduction to statistical learning.)

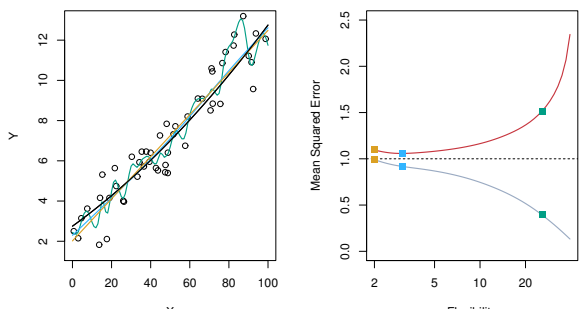
Let's practice: some illustrations

- We generate data from: $Y = f(\boldsymbol{x}) + \epsilon$, with black curve the true f.
- The orange (linear regression), blue (smoothing splines) and green (smoothing splines) curves are three estimates for f, with increasing level of complexity.



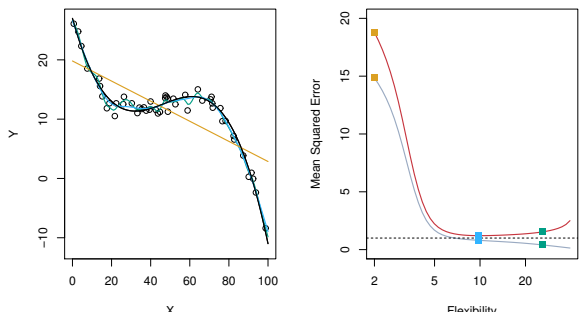
Let's practice: some illustrations (cont.)

- Now the true f is approximately linear.
- Training MSE decreases monotonically as model flexibility increases, and test set has U-shape curve.



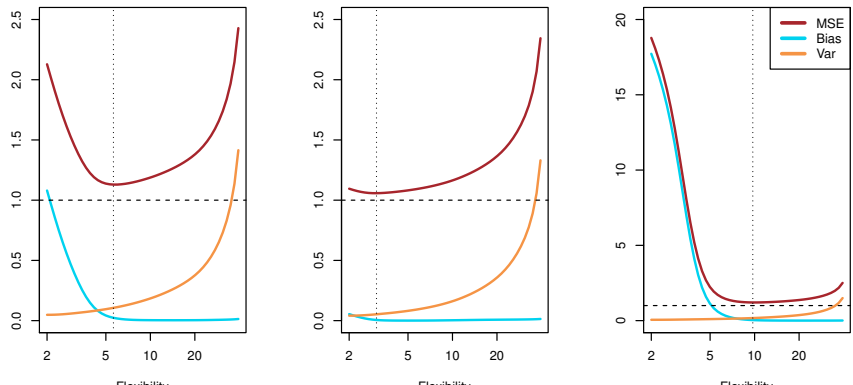
Let's practice: some illustrations (cont.)

- Finally, the true f is highly unlinear.
- Rapid decrease in both curves before test MSE starts to increase slowly.



Let's practice: some illustrations (cont.)

We visualize the bias-variance trade off for the three examples considered before.



Illustrations: the K–nearest neighbors (KNN) classifier

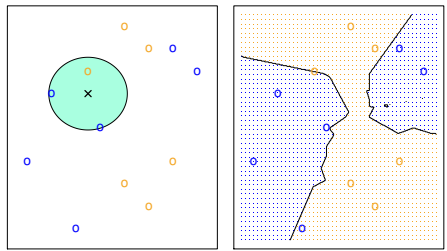
- Given a positive integer K and a test observation x_0, we identify the K points in the training data set that are "closest" to x_0.
- The set $\mathcal{N}_0$ results and we use

$$\Pr(Y = j | X = x_0) \quad = \quad \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j).$$

Then, KNN classifies the test observation x_0 to the class with the largest probability.

Illustrations: the K–nearest neighbors (KNN) classifier

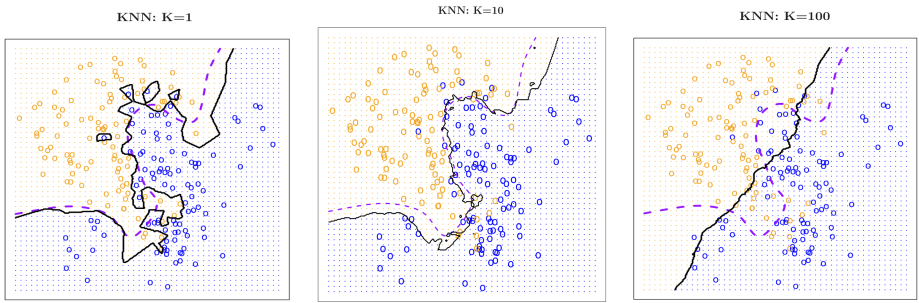
Illustrative example of KNN approach:



(Left) Suppose $K = 3$ and goal is to predict the point labeled by the black cross. (Right) Corresponding KNN decision boundary.

Illustrations: the K–nearest neighbors (KNN) classifier

Now compare KNN with K equals 1, 10 and 100.

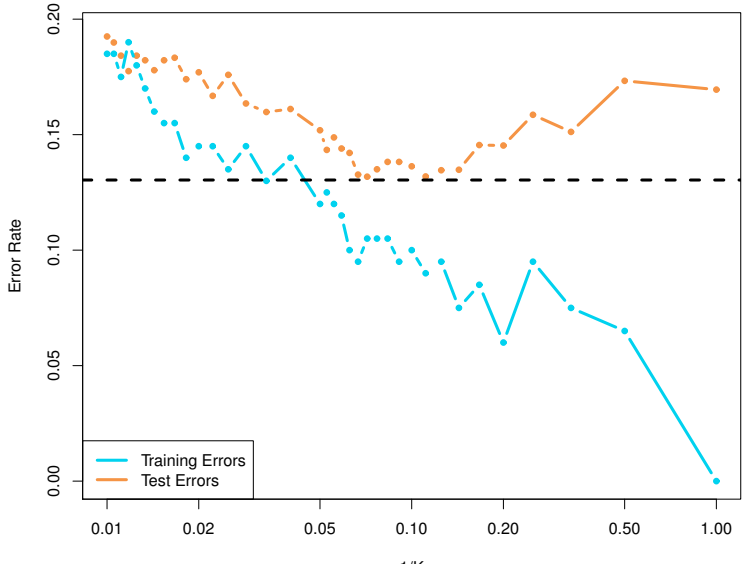


Which classifier do you prefer? Which one is overfitting, underfitting?

Illustrations: the K–nearest neighbors (KNN) classifier

- With $K = 1$, KNN training error rate is 0, but test error rate may be quite high.
- With more flexible classification methods, the training error rate will decline, but the test error rate may not.
- See the plot on the next sheet, where training and test errors are plotted as a function of $1/K$.
- In both regression and classification settings: (model or parameter tuning!)
  - choosing correct level of flexibility is critical
  - the bias-variance tradeoff and the resulting U-shape in the test error can make this a difficult task.

Illustrations: the K–nearest neighbors (KNN) classifier



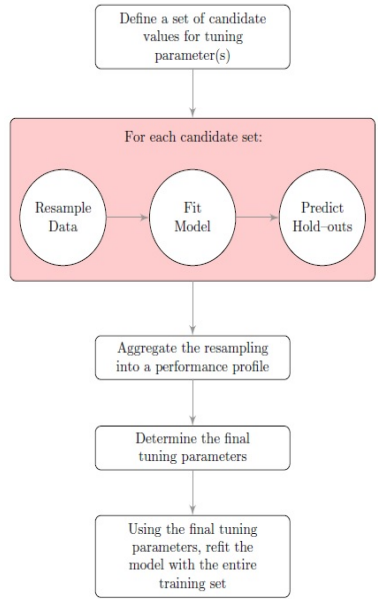
Tuning parameters

- K in the KNN classifier is a very first example of a tuning parameter.
- No analytical formula exists to calculate an appropriate value for such tuning parameters.
- Almost all machine learning methods have tuning parameters that enable the model to flex to find the structure in the data. Examples?
- Parameter tuning:
  - use the existing data to identify settings for the model's parameters
  - use the settings that yield the best and most realistic predictive performance.

Grid search and parameter tuning

General approach for searching for the best parameters:

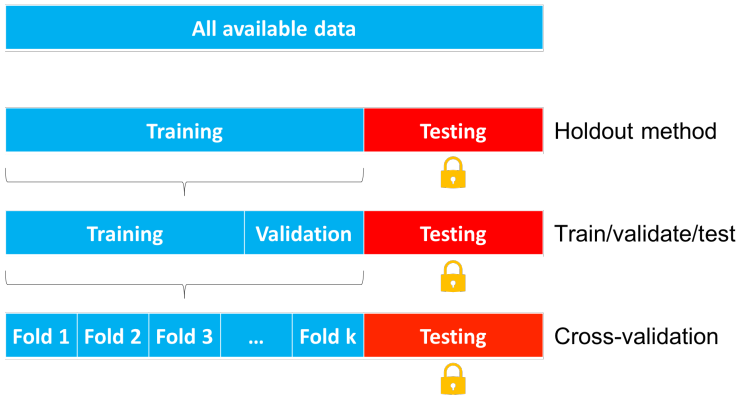
- define a set of candidate values (a grid)
- generate reliable estimates of model utility (i.e. generalization error) across the candidates
- choose the optimal settings
- refit on all available (training) data using optimal settings.



Data splitting

- What we want to assess: how well does the model generalize to new, unseen data?
- Split the data into:
  - training set: to develop, to train, to tune, to compare different settings, ...
  - test set: to obtain an unbiased estimate of the final model's generalization error, or: model's predictive performance.

Data splitting



(Picture taken from Introduction to machine learning in R.)

The validation set

We discussed training and test set. Let's take a deeper dive into resampling methods.

The validation set approach:

- hold out a subset of the training data (e.g. 30%) and evaluate the model on this held out validation set
- calculate the loss on this validation set, as an approximation of the true generalization error
- drawbacks: high variability + inefficient use of data.

Leave-one-out cross validation

- A single observation, say (x_1, y_1), is used for validation and $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ make up the training set.
- $\text{MSE}_1 = (y_1 - \hat{y}_1)^2$ is an unbiased but highly variable estimate for the test error.
- Repeat this procedure to obtain $\text{MSE}_1, \ldots, \text{MSE}_n$ and calculate the LOOCV estimate as:

$$\text{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{MSE}_i,$$

  which is then used to estimate the true generalization error.
- Obviously, computationally very demanding!

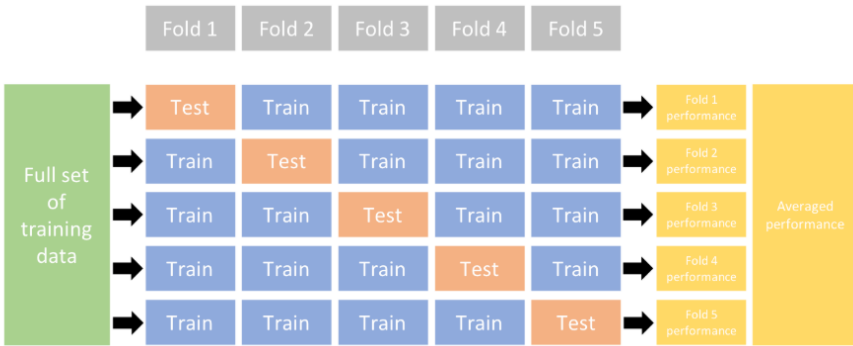
k-fold cross validation

- We randomly divide the set of observations into k groups of approximately equal size.
- We iterate over the k groups, treating each as validation set (and train on the other $k-1$ groups). This gives e.g. MSE_1 calculated on fold 1.
- We repeat this procedure k times and each time treat a different group of observations as a validation set.
- The k-fold CV estimate of the true generalization error is

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\text{MSE}_i.$$

- More stable and accurate compared to validation set approach, less computationally demanding compared to LOOCV.

k-fold cross validation



(Picture taken from Boehmke & Greenwell (2019). Hands-on machine learning with R.)

Stratified k-fold cross validation

- Let $\mathcal{D}_1, \ldots, \mathcal{D}_k$ be the k groups, i.e. disjoint random subsets of approximately the same size.
- Outliers may fall into the same fold $\mathcal{D}_\ell$ and this can substantially distort the k-fold cross-validation error!
- Stratified k-fold cross-validation aims for a more equal distribution of outliers across the folds.

Stratified k-fold cross validation

How does it work?

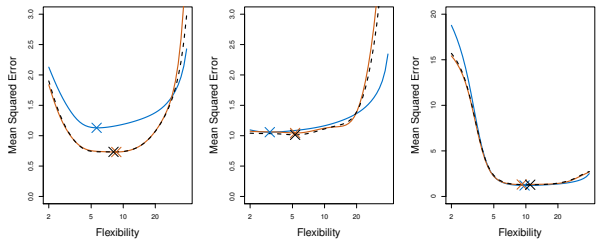
- order the outcomes $Y_{(1)} \geq Y_{(2)} \geq \ldots \geq Y_{(n)}$ (with a deterministic rule in case of ties)
- build urns $\mathcal{U}_\ell$ (with $\ell = 1, \ldots, \lceil n/k \rceil$) such that $\mathcal{U}_1$ contains the k largest observations, $\mathcal{U}_2$ the next k largest observations and so forth
- construct the k groups $\mathcal{D}_1, \ldots, \mathcal{D}_k$ as follows

$$\begin{aligned} \mathcal{D}_\ell \quad = \quad & \big\{\text{pick randomly from each urn } \mathcal{U}_1, \ldots, \mathcal{U}_{\lceil n/k \rceil} \\ & \text{one case (without replacement)}\big\}, \end{aligned}$$

with $\ell = 1, \ldots, k$.

Back to the illustrations

- In the examples discussed earlier the data were simulated, thus: we know the true test MSE.
- The plot shows the true test MSE (in blue), LOOCV (in black) and 10-fold CV (in orange).
- Interest lies in the location of the minimum point in the estimated test MSE curve.

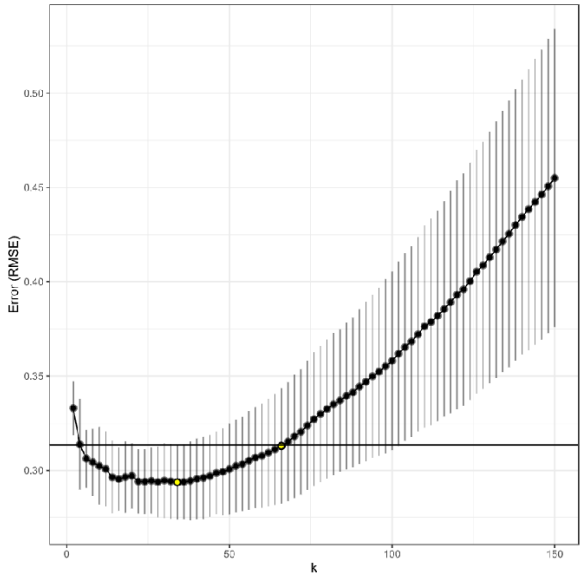


Choosing final tuning parameters

- We choose the final settings by:
  - quantifying model performance (i.e. generalization error) across sets of tuning parameters
  - pick the settings associated with the numerically best performance estimates.
- But, often, less is more:
  - opt for a simpler model that is within a certain tolerance of the numerically best performer
  - 'one-standard error' rule:

    find the numerically optimal value and its corresponding s.e. and seek the model whose performance is within a single s.e. of the numerically best value.

Choosing final tuning parameters



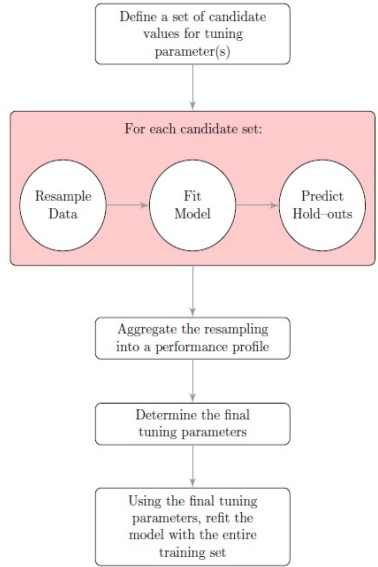
Putting it all together

Model training & validation phase

- choose the optimal settings for the tuning parameters via resampling
- refit the model on the entire training data with the final tuning parameters
- evaluate model utility on the test set.

Model selection

- repeat the above for different models
- compare generalization error via the test data.



Final thoughts

- A possible scheme for finalizing the type of model:
  1. start with several models that are the least interpretable and most flexible (e.g. boosted trees)

     among many domains these models have a high likelihood of producing empirically optimal results (='gold standard').
  2. investigate simpler models that are less opaque (e.g. not complete black boxes)
  3. consider using the simplest model that reasonably approximates the performance of the more complex models.
- Our (2022) paper develops a workflow to construct such global surrogate models.