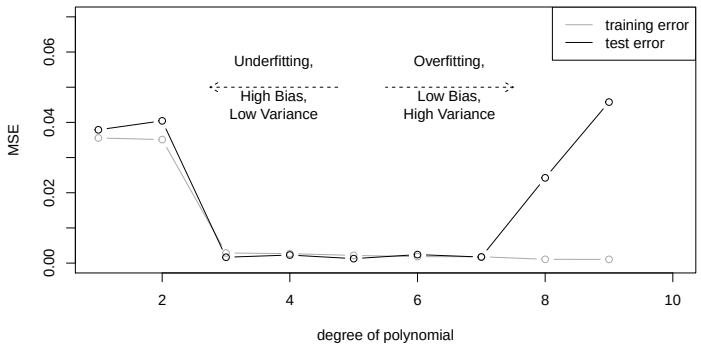
Introduction to Machine Learning

Evaluation: Test Error



Learning goals

- Understand the definition of test error
- Understand how overfitting can be seen in the test error

TEST ERROR AND HOLD-OUT SPLITTING

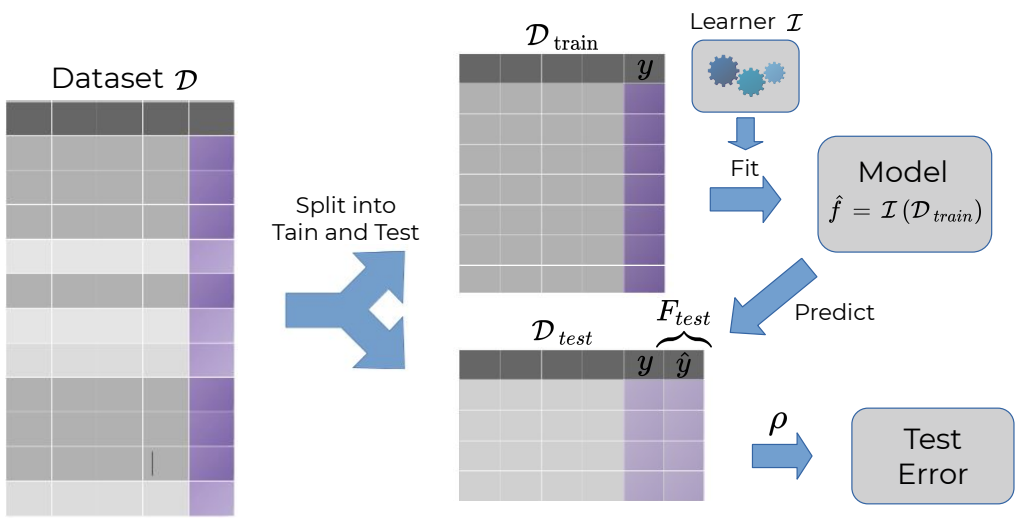
- In order to avoid optimistic bias we will use the **test error**, i.e.,

$$\rho(\mathbf{y}_{\text{test}}, F_{\text{test}}), \text{ where } F_{\text{test}} = \left[\hat{f}_{\mathcal{D}_{\text{train}}}(\mathbf{x}_{\text{test}}^{(1)})^\top \quad \dots \quad \hat{f}_{\mathcal{D}_{\text{train}}}(\mathbf{x}_{\text{test}}^{(m)})^\top\right]^\top,$$

  to simulate how our model performs on new, unseen data.

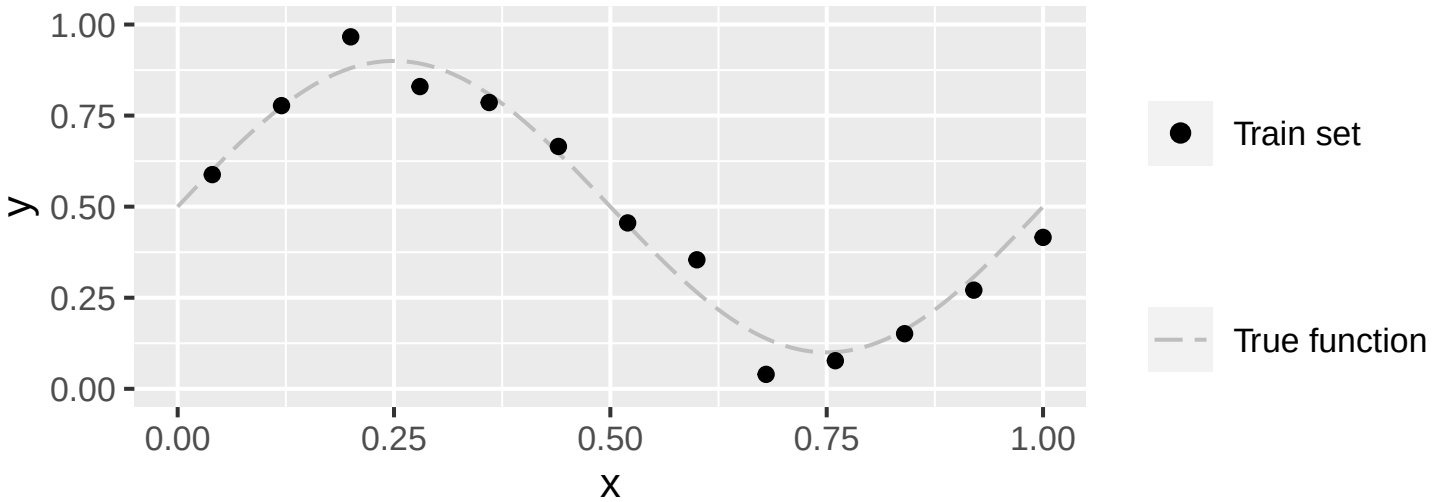
- Evaluating a given model therefore means predicting only on the test data and measuring the resulting performance.

- This implies splitting the data into disjoint sets (e.g., 2/3 for training and 1/3 for testing).



EXAMPLE: POLYNOMIAL REGRESSION

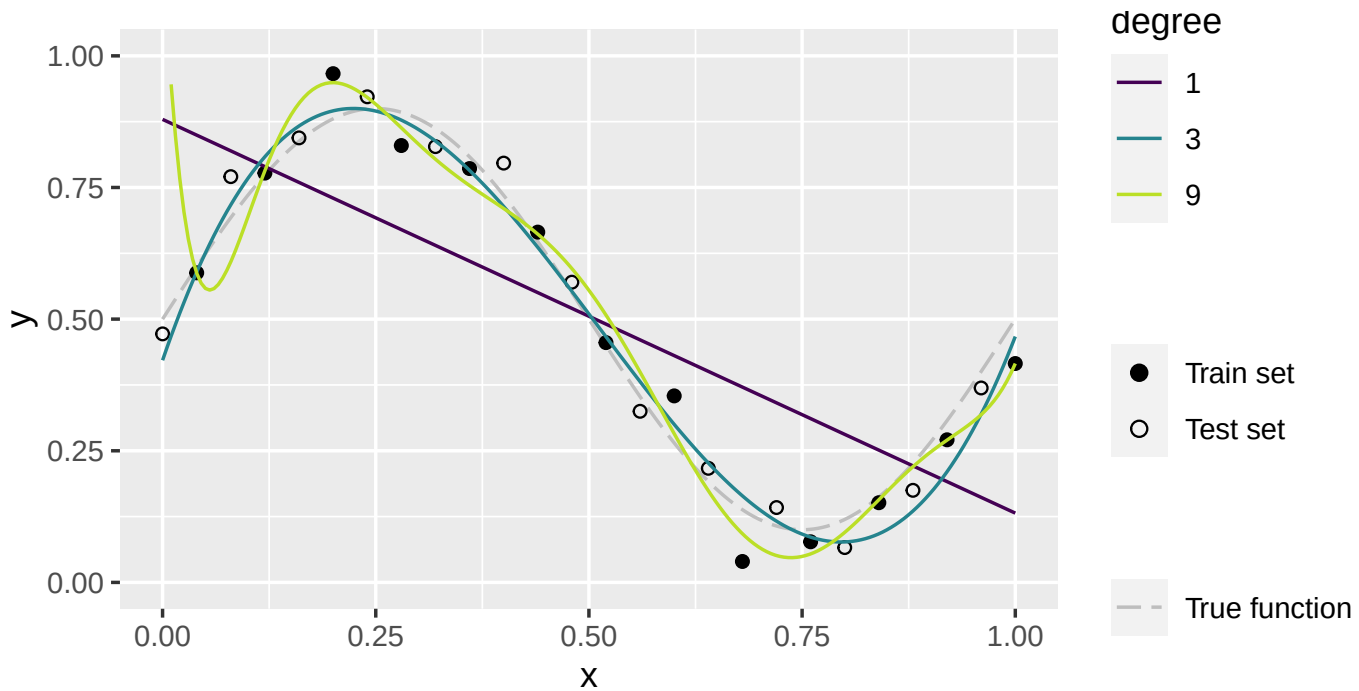
Consider the previous example with the sinusoidal function $0.5 + 0.4 \cdot \sin(2\pi x) + \epsilon$:



Again, we approximate the data with a d^{th}-degree polynomial:

$$f(\mathbf{x} \mid \boldsymbol{\theta}) = \theta_0 + \theta_1 \mathbf{x} + \cdots + \theta_d \mathbf{x}^d = \sum_{j=0}^{d} \theta_j \mathbf{x}^j.$$

EXAMPLE: POLYNOMIAL REGRESSION

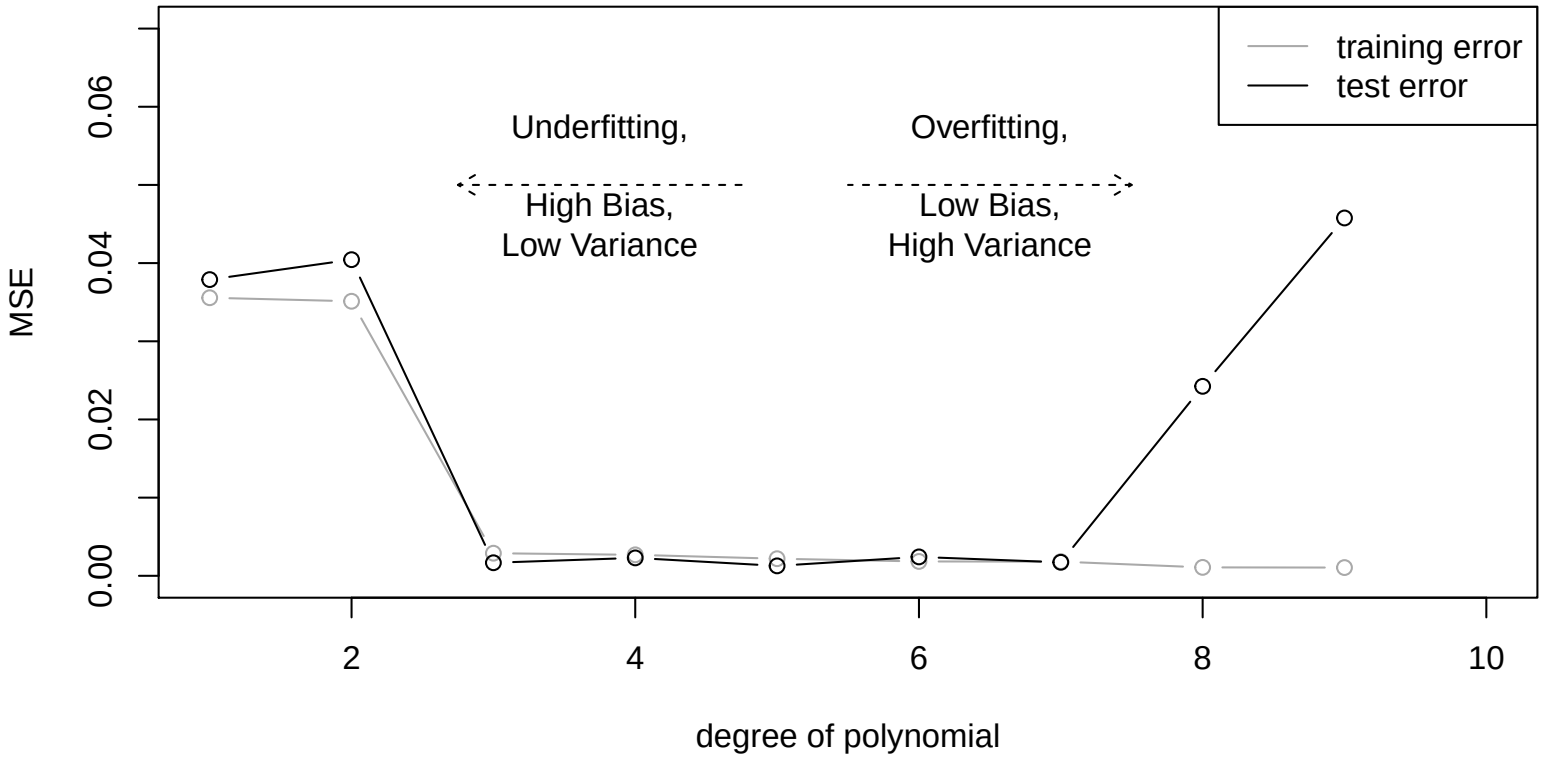


- $d = 1$: MSE = 0.038: clearly underfitting
- $d = 3$: MSE = 0.002: pretty OK
- $d = 9$: MSE = 0.046: clearly overfitting

While the training error monotonically decreases for rising d, the test error reflects the fact that higher-degree polynomials overfit the data.

TEST ERROR

Plot evaluation measure for all polynomial degrees:



Increasing model complexity tends to cause

- a decrease in training error, and
- a U-shape in test error
  (first underfit, then overfit, sweet-spot in the middle).

TRAINING VS. TEST ERROR

- We take the Boston Housing data set where the value of houses in the area around Boston is predicted based on 13 features describing the region (e.g., crime rate, name of the town, etc.).
- We fit a polynomial regression model on it

$$\mathbf{y}_{medv} = \beta_0 + \sum_{j=1}^{d} \sum_{i=1}^{n} \beta_{i,j} \left(\mathbf{x}_i\right)^j$$

  with n features and d degrees polynomials.
- We observe the train and test error when we change the size of training set, the size of the test set and the model complexity (incresing the possible degrees of the polynomials).

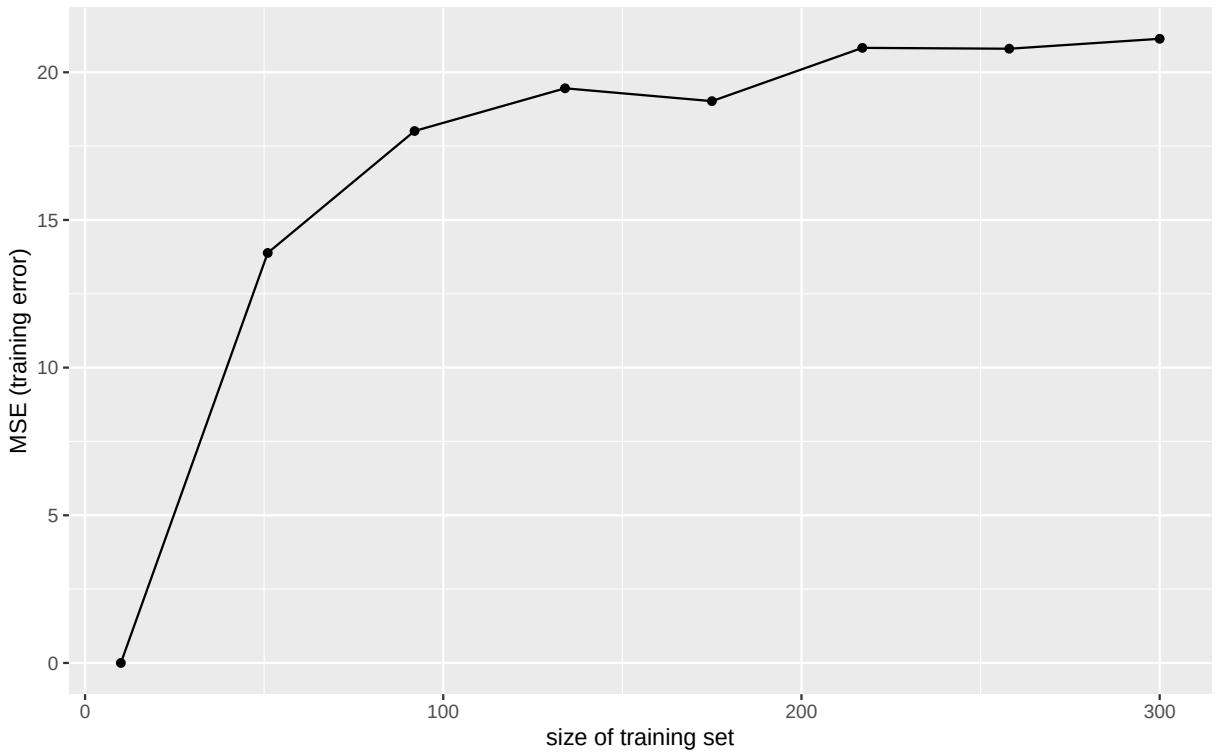
The training error...

- is a biased estimator as performance is measured on the same data the model was trained on.

TRAINING VS. TEST ERROR

The training error...

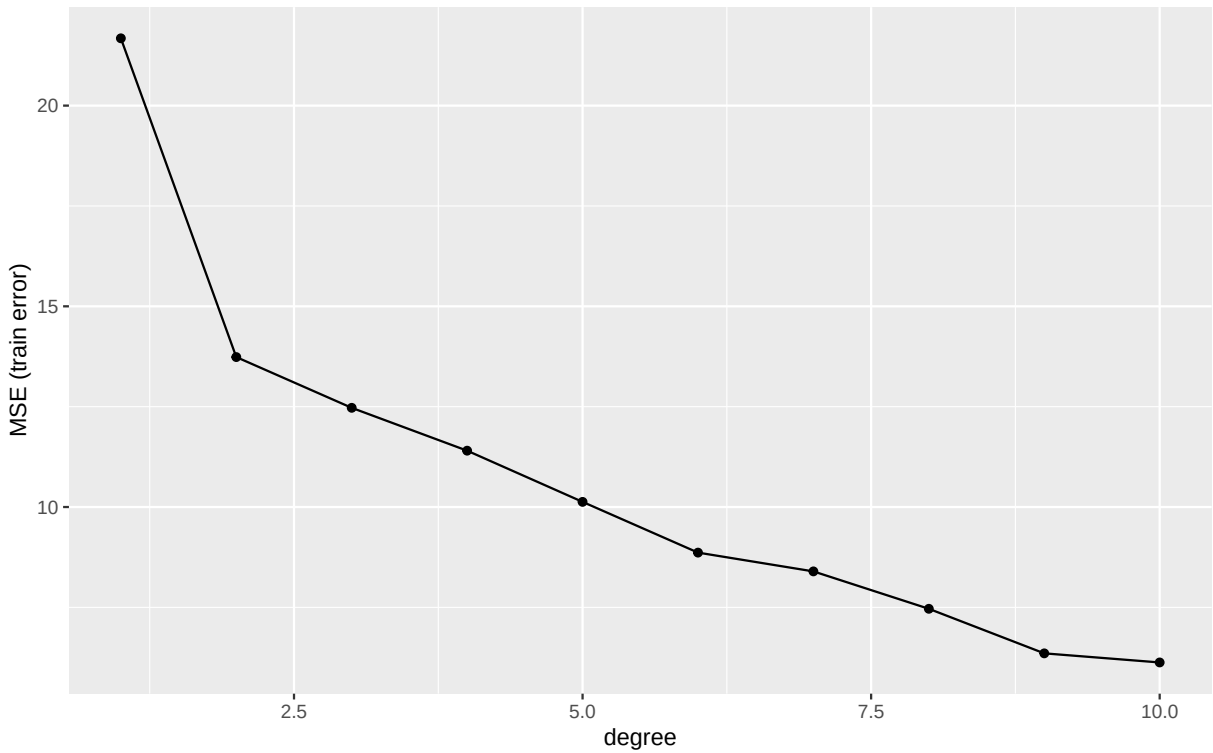
- decreases with smaller training set size as it becomes easier for the model to learn all observed patterns perfectly.



TRAINING VS. TEST ERROR

The training error...

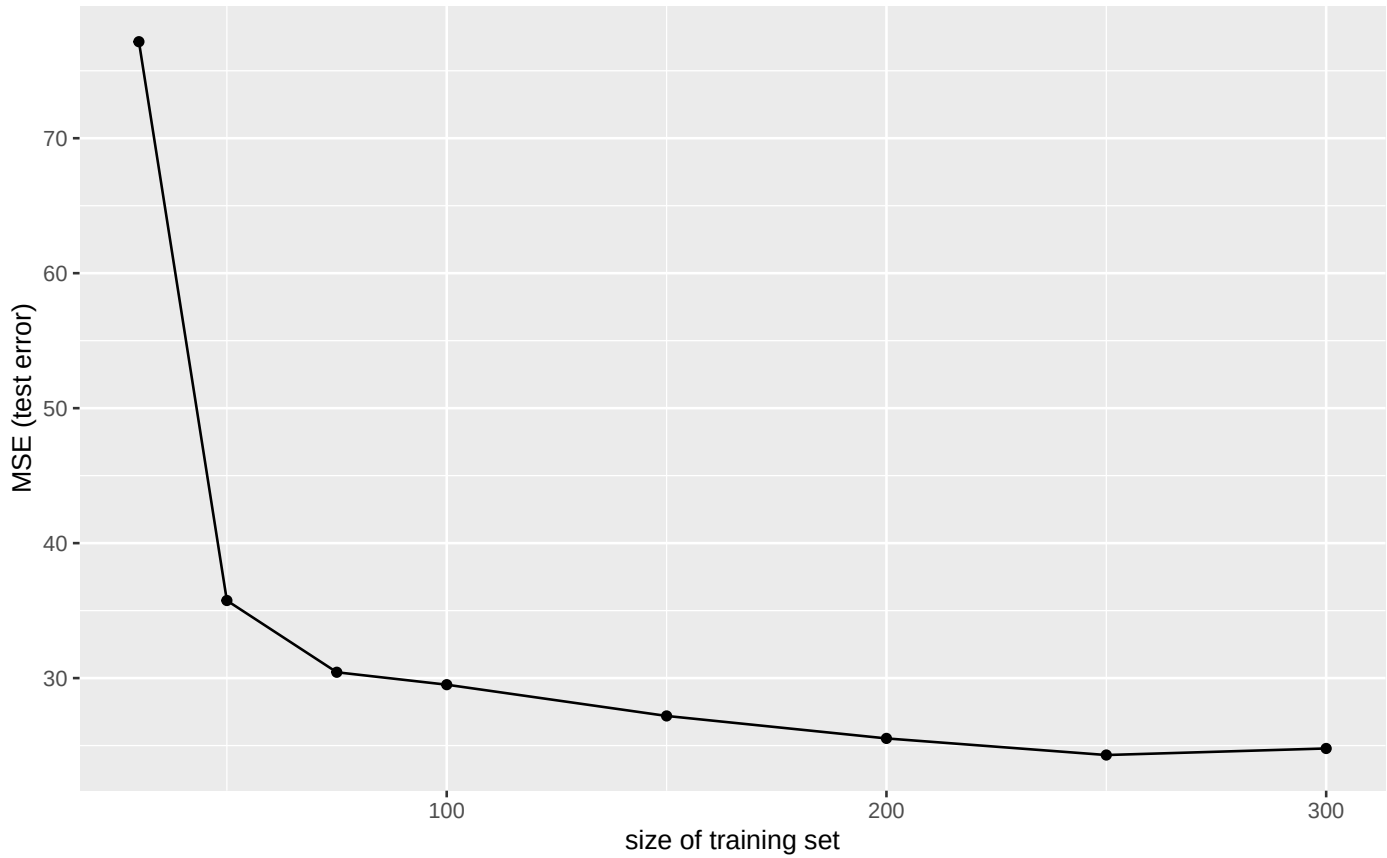
- decreases with increasing model complexity as the model gets better at learning more complex structures (here: more degrees in polynomial regression).



TRAINING VS. TEST ERROR

The test error...

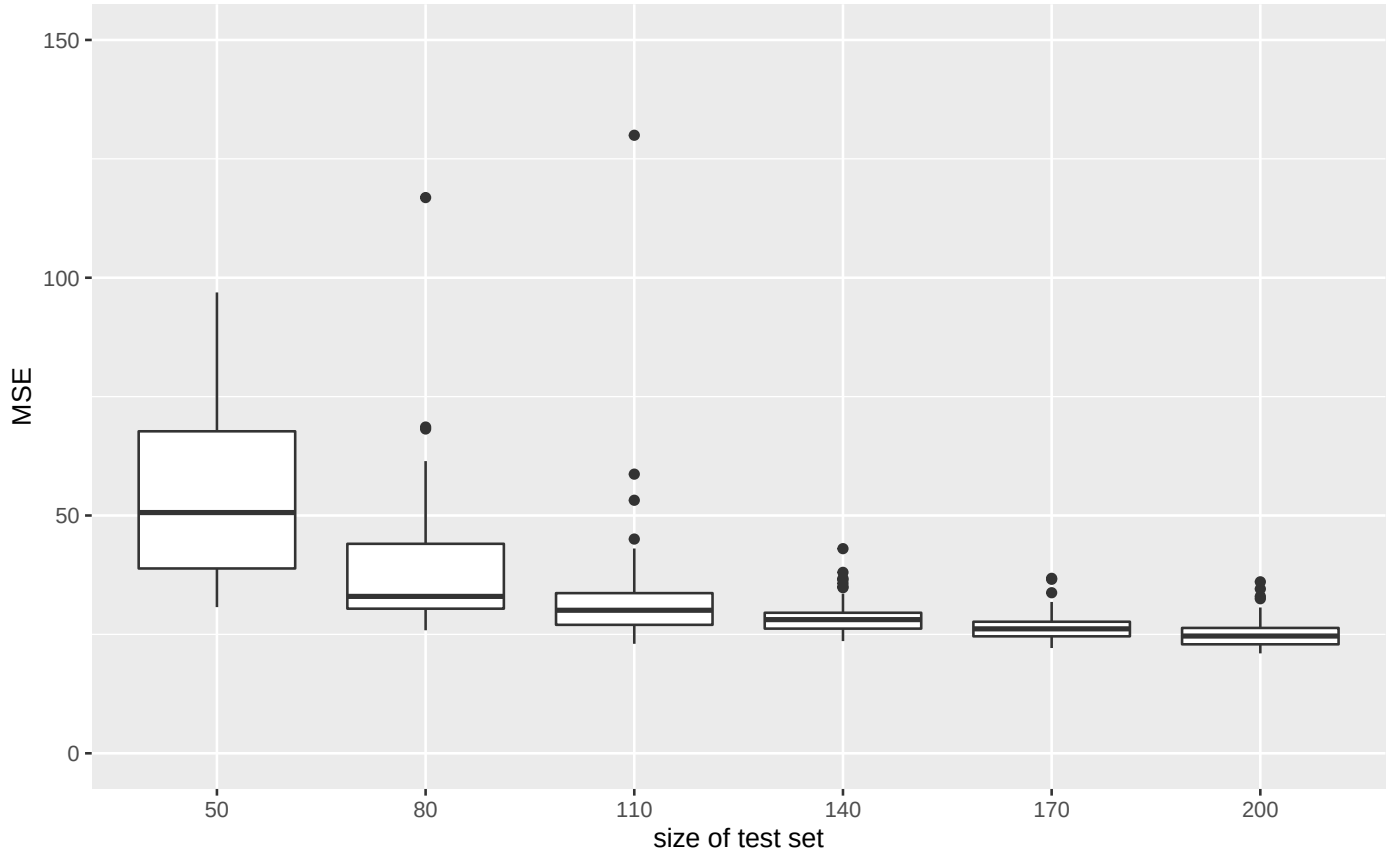
- will typically decrease with larger training set size as the model generalizes better with more data to learn on.



TRAINING VS. TEST ERROR

The test error...

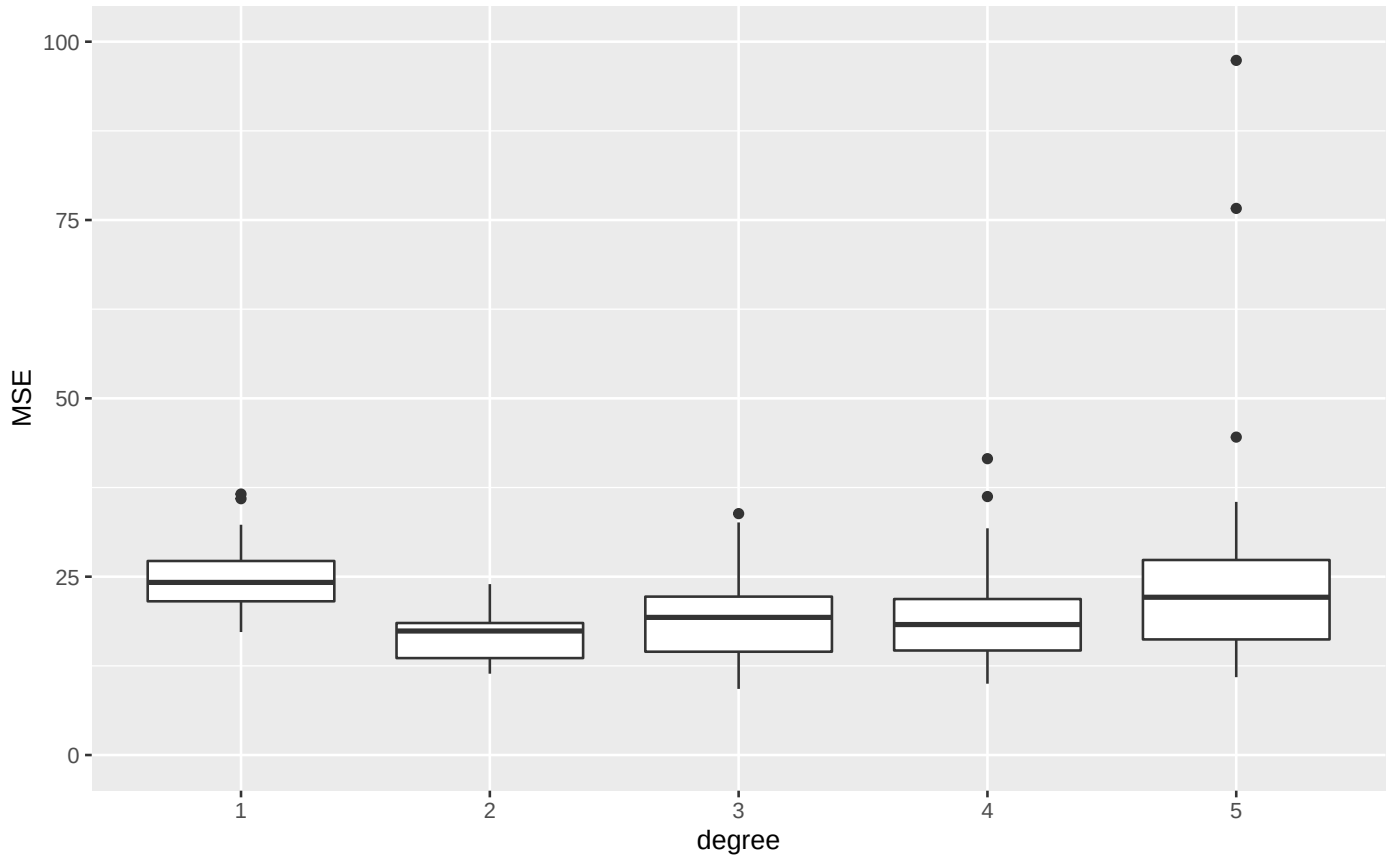
- will have higher variance with smaller test set size.



TRAINING VS. TEST ERROR

The test error...

- will have higher variance with increasing model complexity (here: more degrees in polynomial regression).



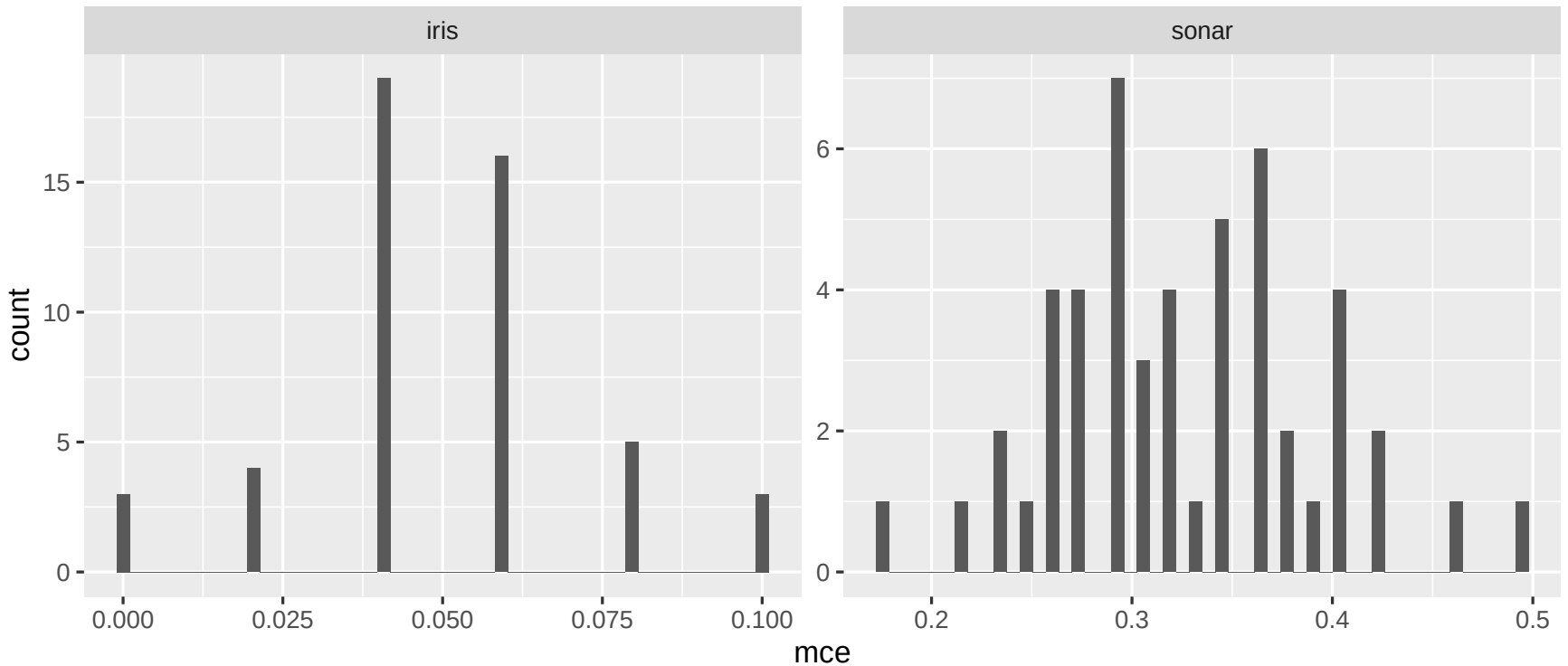
TEST ERROR PROBLEMS

- In general, the test error is a good estimator of future performance, **given** the test data and data we might see in future applications are indeed *iid* samples from the same underlying distribution.
- Hold-out sampling produces a trade-off between **bias** and **variance** that is balanced by the split ratio.
- **Sample size** plays a crucial role in deciding on a split strategy:
  - If the size of our initial, complete data set $\mathcal{D}$ is limited, single train-test splits can be problematic.
  - Small-sample problems come in different shapes in ML – maybe overall set size is sufficient but one of the classes is very small.
- It is generally advisable to try out different train-test splits and study the resulting error measurement fluctuation.

TEST ERROR PROBLEMS

We simulate repeated $\frac{2}{3}$ / $\frac{1}{3}$ train-test splits on two ML tasks:
`iris` ($n = 150$) and `sonar` ($n = 208$).
So we have about 50 (`iris`) and 70 (`sonar`) observations in our respective test sets.

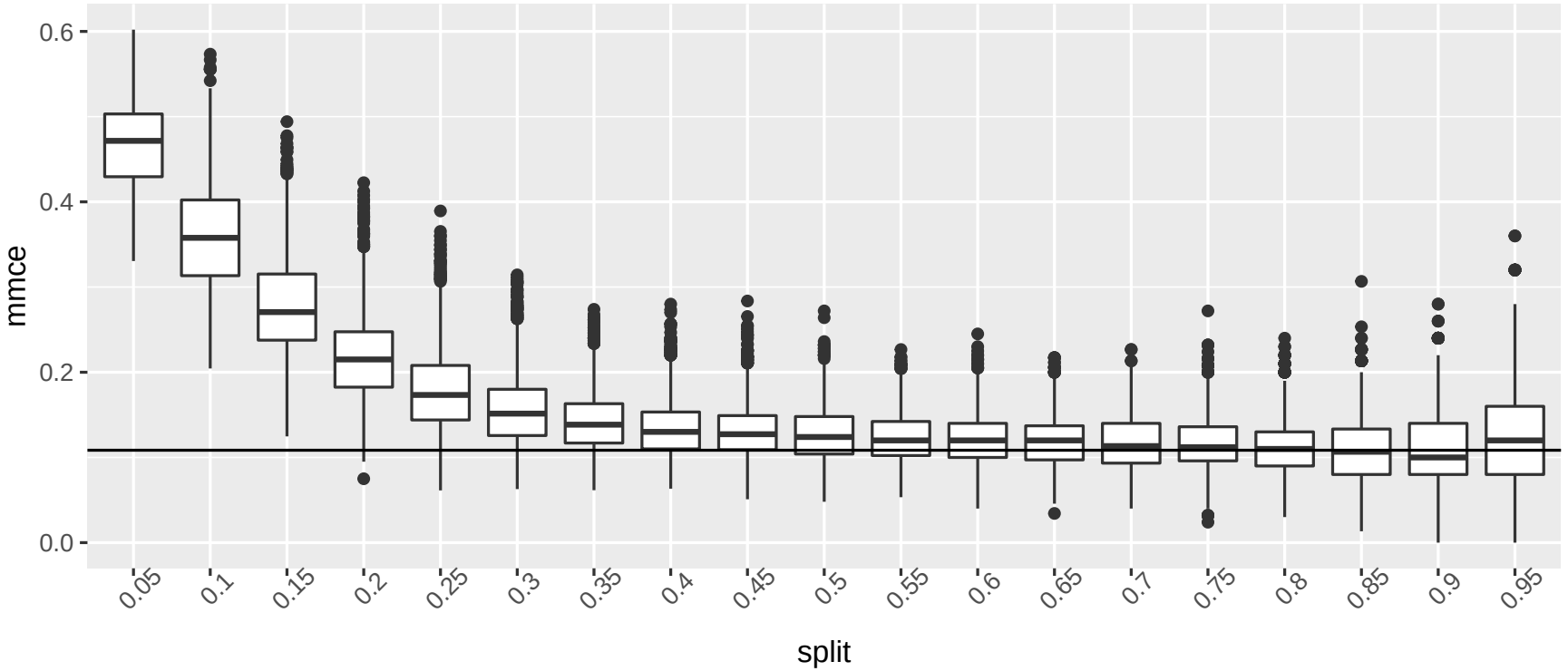
The plots below show the strong variation in test errors (50 repetitions).



BIAS-VARIANCE OF HOLD-OUT – EXPERIMENT

- Data: simulate `spirals` data ($sd = 0.1$) from `mlbench`.
- Learner: CART (`classif.rpart` from `mlr3`).
- Goal: estimate real performance of a model with $|\mathcal{D}_{\text{train}}| = 500$.
  - Get the "true" estimator by repeatedly sampling 500 observations from the simulator, fit the learner, then evaluate on 10^5 observations – obviously not feasible in practice.
  - Analyze different split rates $s \in \{0.05, 0.10, ..., 0.95\}$ with $|\mathcal{D}_{\text{train}}| = s \cdot 500$.
  - Estimate performance on $\mathcal{D}_{\text{test}}$ with $|\mathcal{D}_{\text{test}}| = (1 - s) \cdot 500$.
  - Repeat the experiment 50 times for each split rate.

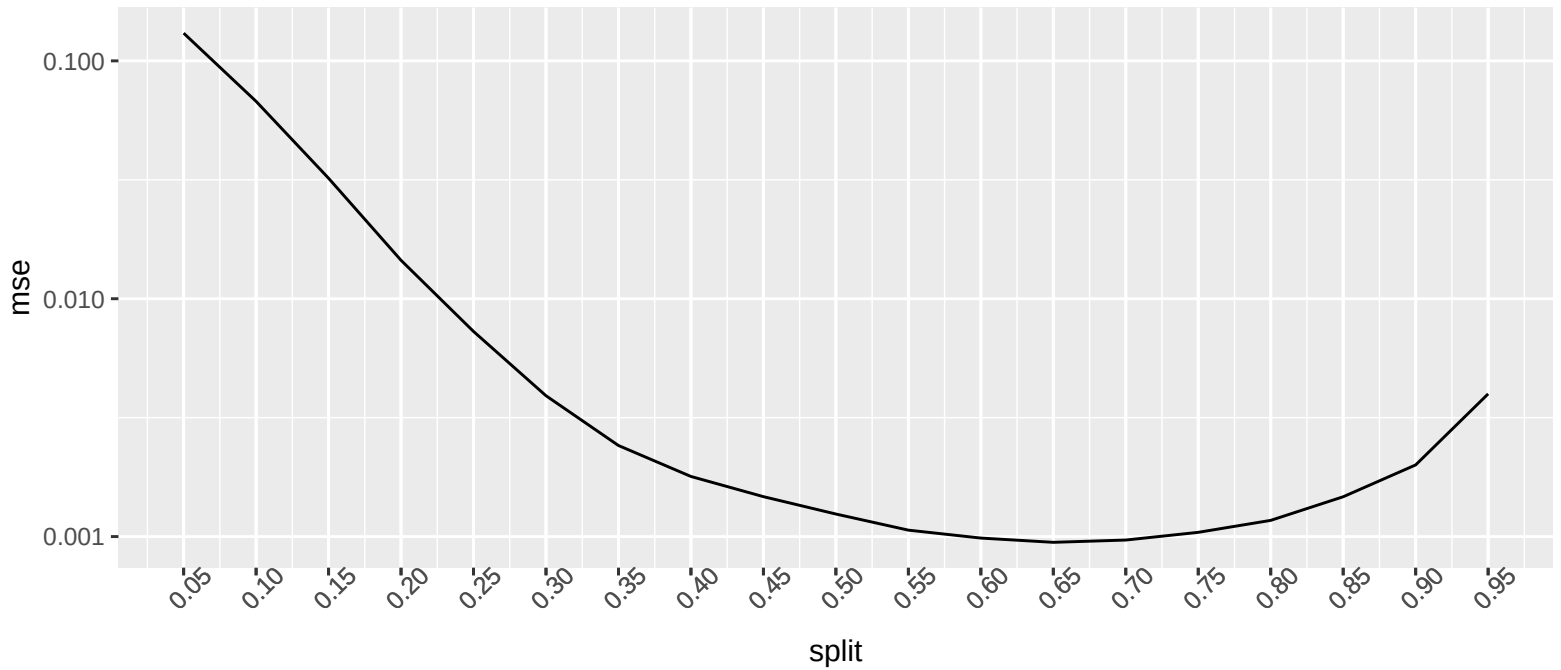
BIAS-VARIANCE OF HOLD-OUT – EXPERIMENT



- We clearly see the pessimistic bias for small training sets – we cannot learn much from substantially fewer than 500 observations.
- At the same time, we observe an increase in variance when test sets become smaller.

BIAS-VARIANCE OF HOLD-OUT – EXPERIMENT

- We now plot the MSE between true performance (horizontal line in previous plot) and hold-out values in each boxplot.
- The split rate with the lowest MSE value produces the best estimator, which is pretty close to a training set ratio of 2/3.
- NB: this is a single experiment and not a scientific study, but this rule-of-thumb has also been validated in larger studies.



TEST ERROR

To clear up a major point of confusion:

- In ML we frequently face a weird situation.
- We are usually given a single data set, and at the end of our model selection and evaluation process, we will likely fit one model on exactly that complete data set.
- As training error evaluation does not work, we have no other option but to evaluate exactly that model.
- Hold-out splitting (and **resampling**) are tools to estimate future performance in a valid manner.
- All of the models produced during that phase of evaluation are only intermediate results.