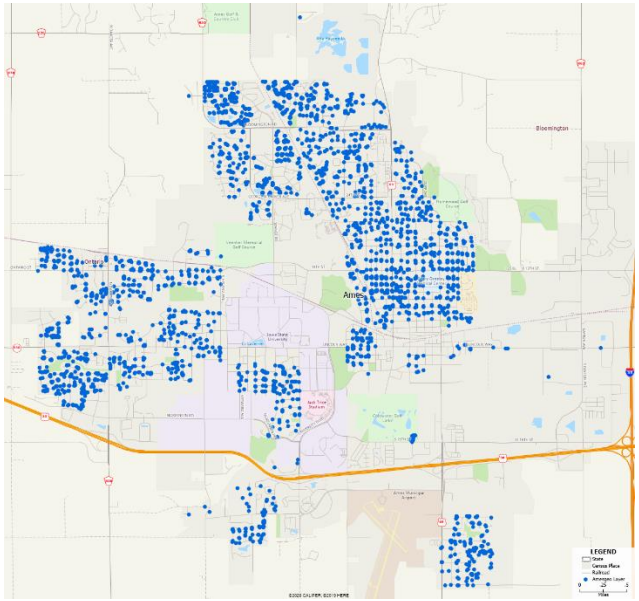
H2O package

- A popular machine learning environment is H2O, which is build around *Java*. For more information see:
  - https://www.h2o.ai/resources/
  - https://www.h2o.ai/wp-content/themes/h2o2016/images/resources/RBooklet.pdf
- It supports multiple processor cores and NVIDIA GPUs in the supplemental package **`H2O4GPU`**

Ames Housing Data Set

- This data set can be found in the package **`AmesHousing`**, which includes also locational information.



- It includes the raw data and several functions setting the data up for analysis.
- A basic description of the origins of this data set are found in http://jse.amstat.org/v19n3/decock.pdf

- The variables, many of them factors are documented in https://ww2.amstat.org/publications/jse/v19n3/decock/DataDocumentation.txt

Workflow of machine learning

- In contrast to confirmatory analysis ML is a ***heuristic*** and ***iterative*** process.
- Its objective is to build, based on sample data, a ***generalizable*** model that can be applied to perform ***predictions*** on new data.
- The modelling approach is not restricted to one algorithm. There is no guarantee that one model class is ***optimal*** for all scenarios.

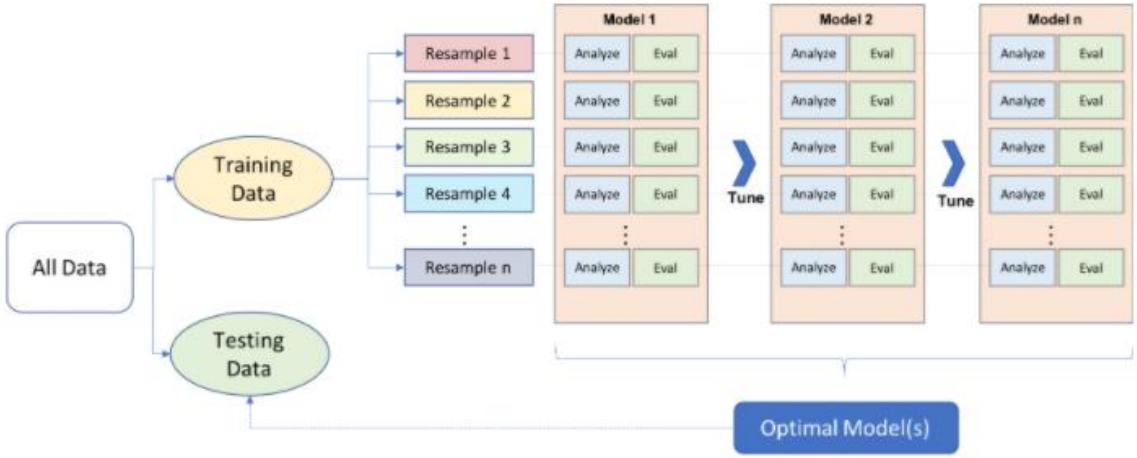


Figure 2.1: General predictive machine learning process.

Data preparation

- There are many different machine learning tools out there. Each tool requires its input data in a tool-specific format. Also, the output format many differ from tool to tool.

- For instance, since **`h2o`** is using internally *Java* the data need to be reformatted.

Sampling

- Calibrated models always are specific to the sample data. In order to check ***generalizability,*** they need to be tested on ***independent*** data sets.
- The used data need to be representative for the general population for which predictions are desired.
- One way to obtain an independent data set is to take a random sample from the available data.
  - This reduces the number of observations to calibrate the models, but it saves computing time.
  - The proportion of data points withheld depends on the circumstances.

Simple random sampling

- To achieve replicable studies the random number generator needs to be initialized (see document **`RNGVersions.pdf`**)
- A random sample is taken with the objective that the ***distributions*** of the sample and test data sets match.

Stratified random sampling

- Stratified random sampling breaks the data set with respect to the target feature up into classes (factor levels or quantiles) and then samples from each sub-groups.
- This achieves equal proportions of test and sample data in each sub-group.

Down- and up-sampling

- For categorical target variables a class of interest, e.g., rare disease, may the sparsely populated which leads to an ***imbalance*** in which the larger classes will dominate the calibration outcome.
- To overcome this problem
  - for large data sets the number of observations in the over-represented classes can be ***randomly reduced***.

- for small data sets the under-represented class “additional” observations can be “generated” by data ***repetition*** or ***bootstrapping***.

R Inconsistencies

- Depending on a package’s author(s) data structures and function call conventions can differ substantially.
- Notes:
  - For spatial data set some conventions are established as **`sp`** and the newer **`sf`** object formats.
  - For statistical procedures, the **`formula`** convention and its extensions are common practice.
- This lack of common standards is an annoying feature of the R environment.

Many formula interfaces

```r
# Sale price as function of neighborhood and year sold
model_fn(Sale_Price ~ Neighborhood + Year_Sold,
         data = ames)

# Variables + interactions
model_fn(Sale_Price ~ Neighborhood + Year_Sold +
           Neighborhood:Year_Sold, data = ames)

# Shorthand for all predictors
model_fn(Sale_Price ~ ., data = ames)

# Inline functions / transformations
model_fn(log10(Sale_Price) ~ ns(Longitude, df = 3) +
           ns(Latitude, df = 3), data = ames)
```

```
# Use separate inputs for X and Y
features <- c("Year_Sold", "Longitude", "Latitude")
model_fn(x = ames[, features], y = ames$Sale_Price)
```

- Interface used by **`h2o`** package:

```
model_fn(
  x = c("Year_Sold", "Longitude", "Latitude"),
  y = "Sale_Price",
  data = ames.h2o
)
```

Many interfaces and meta-engine

- **`caret`** package and newer **`parsnip`** package performing meta capabilities by providing wrapper interfaces to other ML packages.

```
lm_lm    <- lm(Sale_Price ~ ., data = ames)
lm_glm   <- glm(Sale_Price ~ ., data = ames,
                family = gaussian)
lm_caret <- train(Sale_Price ~ ., data = ames,
                  method = "lm")
```

Resampling methods

- Reminder: Do not assess the model performance during the training phase with the test data set.

- How can the training phase then be assessed?
  - Repeatedly splitting the training data set into a training and a validation (***holdout***) data set.
  - Evaluate the training phase for each holdout data set.
  - Average the training phase performances over all holdout data sets.
- Resampling methods repeatedly fit the model on different training data sets and validate their performance.

k-fold cross validation

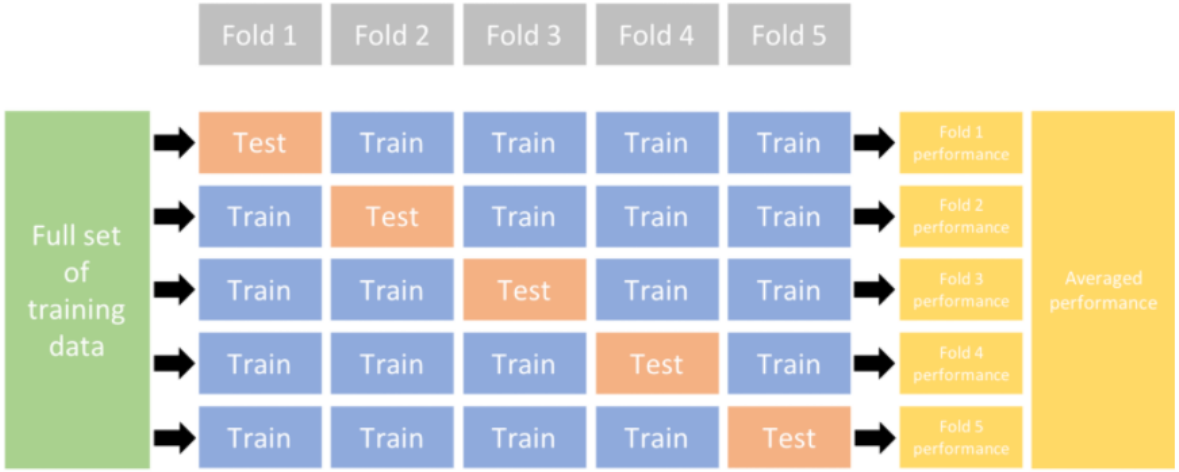


Figure 2.4: Illustration of the k-fold cross validation process.

- The number of folds depends on the circumstances. Typically 5 or 10 folds are used.
- An extreme situation of cross validation is when each fold just consists of just one observation, i.e., leave-one-out cross validation

Bootstrapping

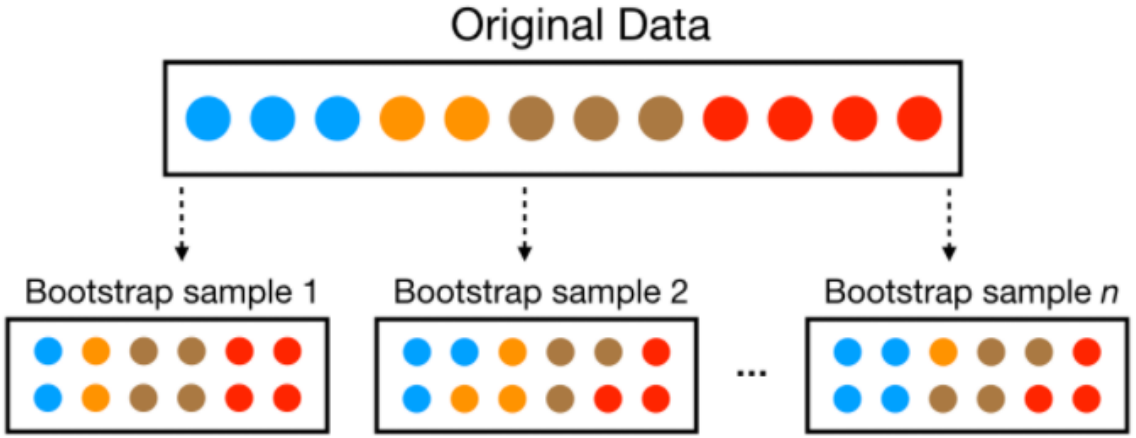


Figure 2.6: Illustration of the bootstrapping process.

- Observations not in the training sample are called “out-of-bag“ (OOB).
- Approximately 63% of unique observations are in the training sample.
- Comparison of cross-validation against bootstrapping

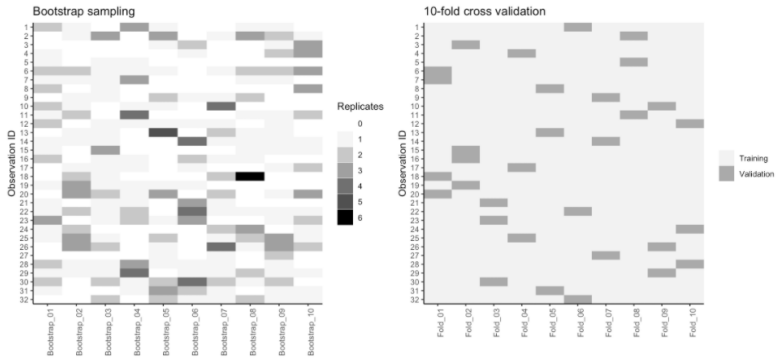


Figure 2.7: Bootstrap sampling (left) versus 10-fold cross validation (right) on 32 observations. For bootstrap sampling, the observations that have zero replications (white) are the out-of-bag observations used for validation.

Bias-variance trade-off

Bias

- This is a systematic mis-specification of the estimator relative to the underlying data generating process.

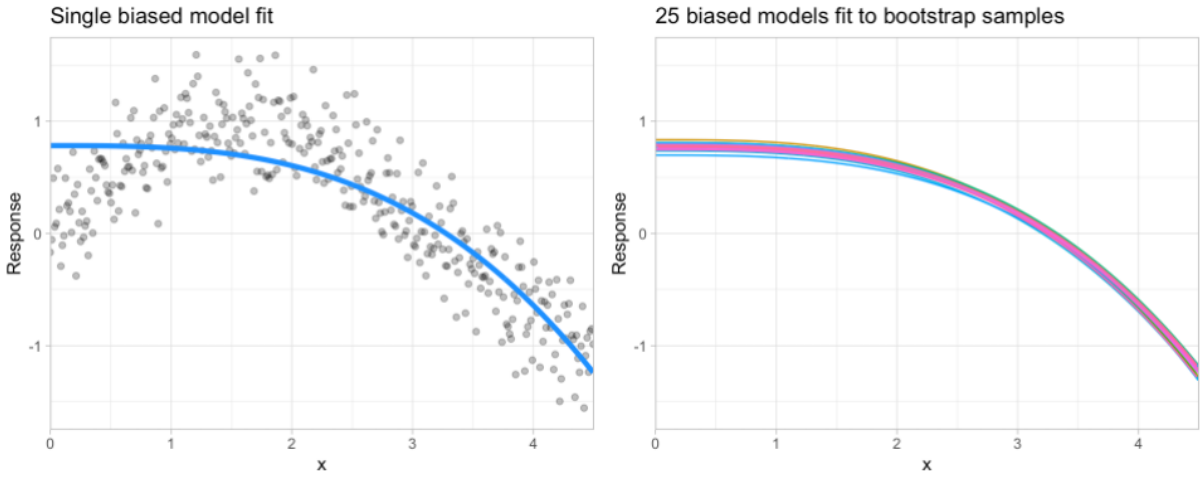


Figure 2.8: A biased polynomial model fit to a single data set does not capture the underlying non-linear, non-monotonic data structure (left). Models fit to 25 bootstrapped replicates of the data are underterred by the noise and generates similar, yet still biased, predictions (right).

Variance

- Over-modelling the data by also capturing the sample specific random variation of the underlying data generating process

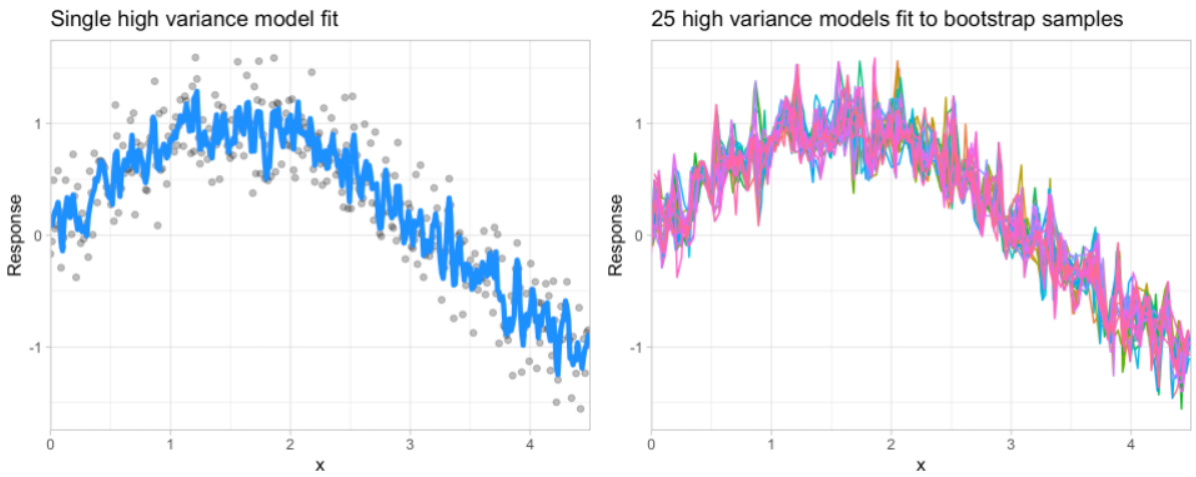


Figure 2.9: A high variance *k*-nearest neighbor model fit to a single data set captures the underlying non-linear, non-monotonic data structure well but also overfits to individual data points (left). Models fit to 25 bootstrapped replicates of the data are deterred by the noise and generate highly variable predictions (right).

Hyperparameter tuning

- Many ML procedures depend on externally giving parameters also called ***hyperparameters***. E.g., order of polygon, smoother factor of splines, bandwidth of kernel densities, or number of nearest neighbors k in knn.
- An optimal hyperparameter strikes the necessary balance between over-fitting the data (increase in variance) and under-fitting the systematic pattern (increase in bias).
- The optimal hyperparameter depend on the given data and the unknown data generating process.
- It needs to be empirically determined using cross-validation based on holdout samples.
- An automated approach is ***grid-search***, which evaluates the model's performance over a set of externally given parameter combinations.

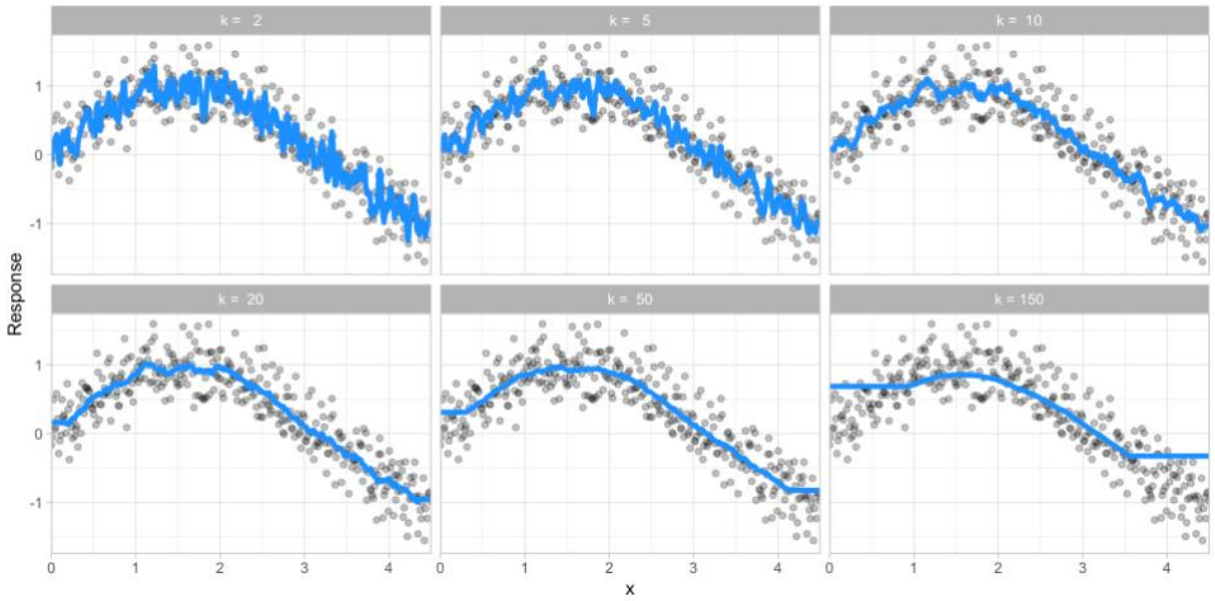


Figure 2.10: *k*-nearest neighbor model with differing values for *k*.

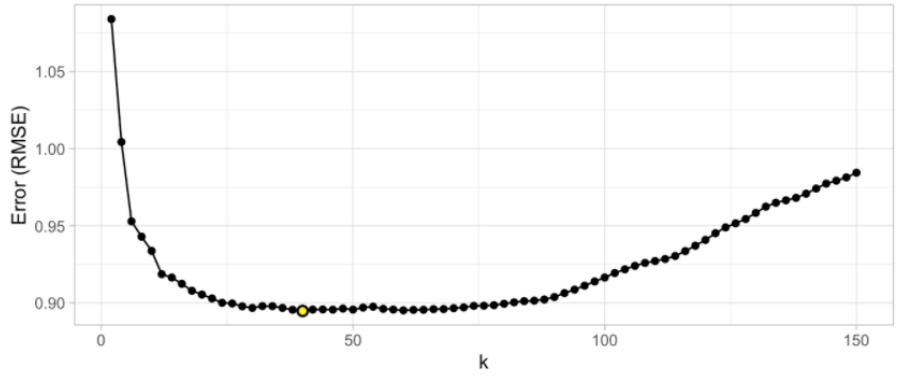


Figure 2.11: Results from a grid search for a *k*-nearest neighbor model assessing values for *k* ranging from 2-150. We see high error values due to high model variance when *k* is small and we also see high errors values due to high model bias when *k* is large. The optimal model is found at *k* = 46.

Model evaluation

- Different metrics of model evaluation for either regression or classification models are discussion in greater details throughout this semester.