Applied Geodata Science I

Session 11

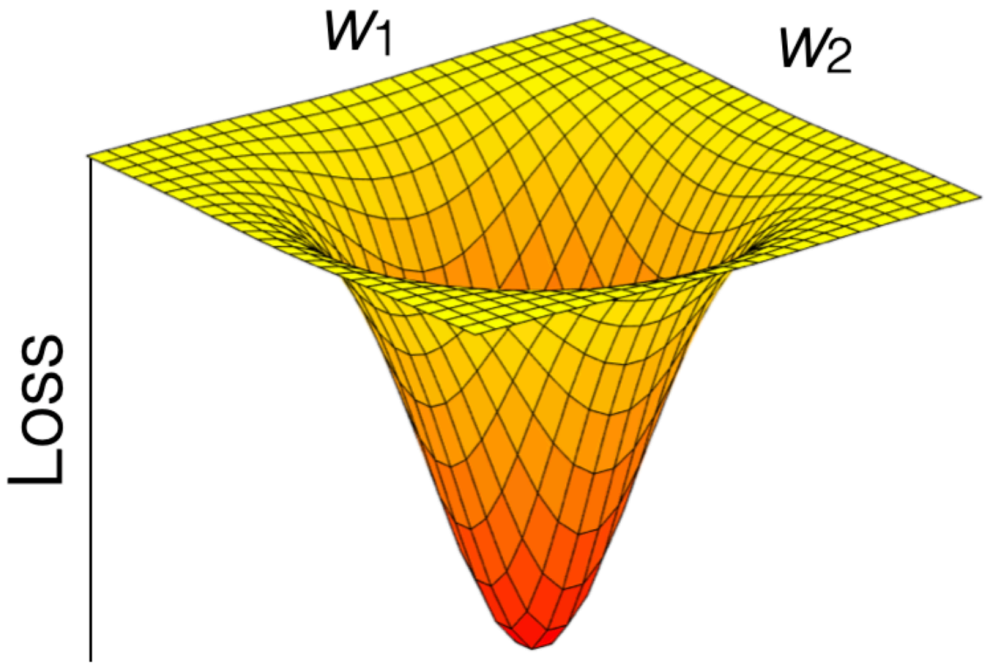
Prof. Dr. Benjamin Stocker

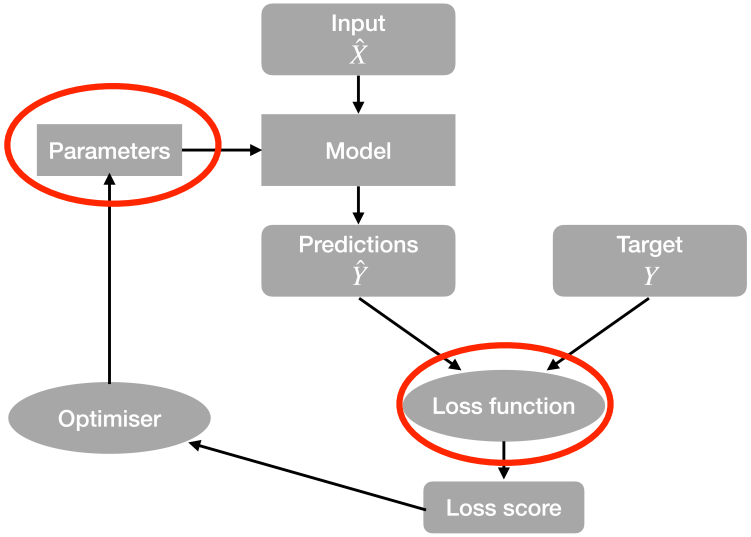
05.05.2025

GECO

UNIVERSITÄT BERN

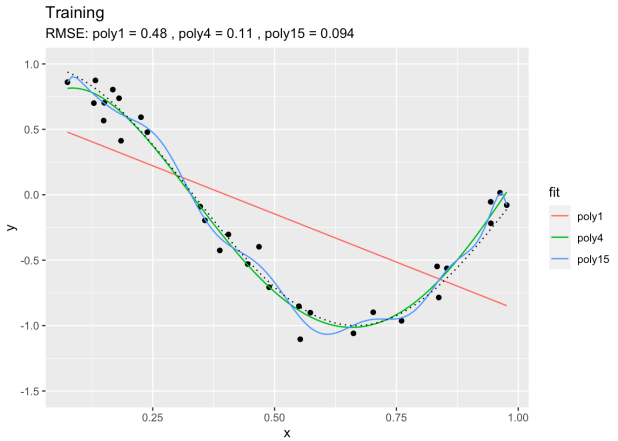
Loss as a function of the model parameters (coefficients)

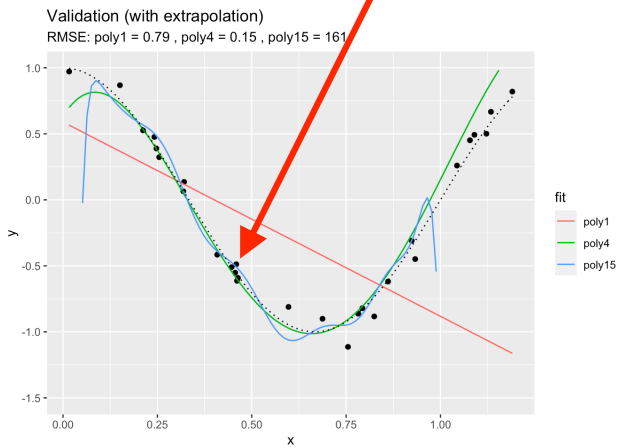




Predicting to new data

These are not the same data points as used for training. New data!





- Model is trained on data
  → yields model object: `mod`

- Trained model is applied to new data:
  `predict(mod, newdata = …)`

Predicting to new data

```
mod <- caret::train(…, data = df1)
```

or:

```
mod <- lm(…, data = df1)
```

or…

Model training

```
df1
variables: y, x1, x2, x3
```

`mod`

Fitted model object

- Contains information about model structure (model type, formula) and coefficients (parameters)

```
predict(mod, newdata = df2)
```

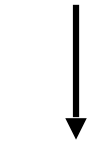
- `df2` must contain variables (columns) used in the formula.

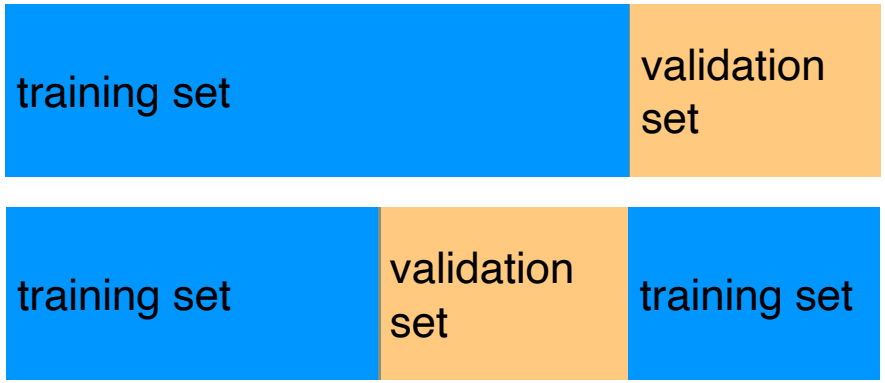
Predicting to new data

```r
eval_model <- function(mod, df_train, df_test, return_metrics = FALSE){

  require(magrittr)  # part of dplyr, required for the "old" pipe

  # add predictions to the data frames----
  df_train <- df_train %>%
    drop_na() %>%  # magrittr pipe necessary here for the dot ('.') evaluation
    mutate(fitted =  predict(mod, newdata = .))

  df_test <- df_test %>%
    drop_na() %>%
    mutate(fitted =  predict(mod, newdata = .))
```

K-fold cross validation

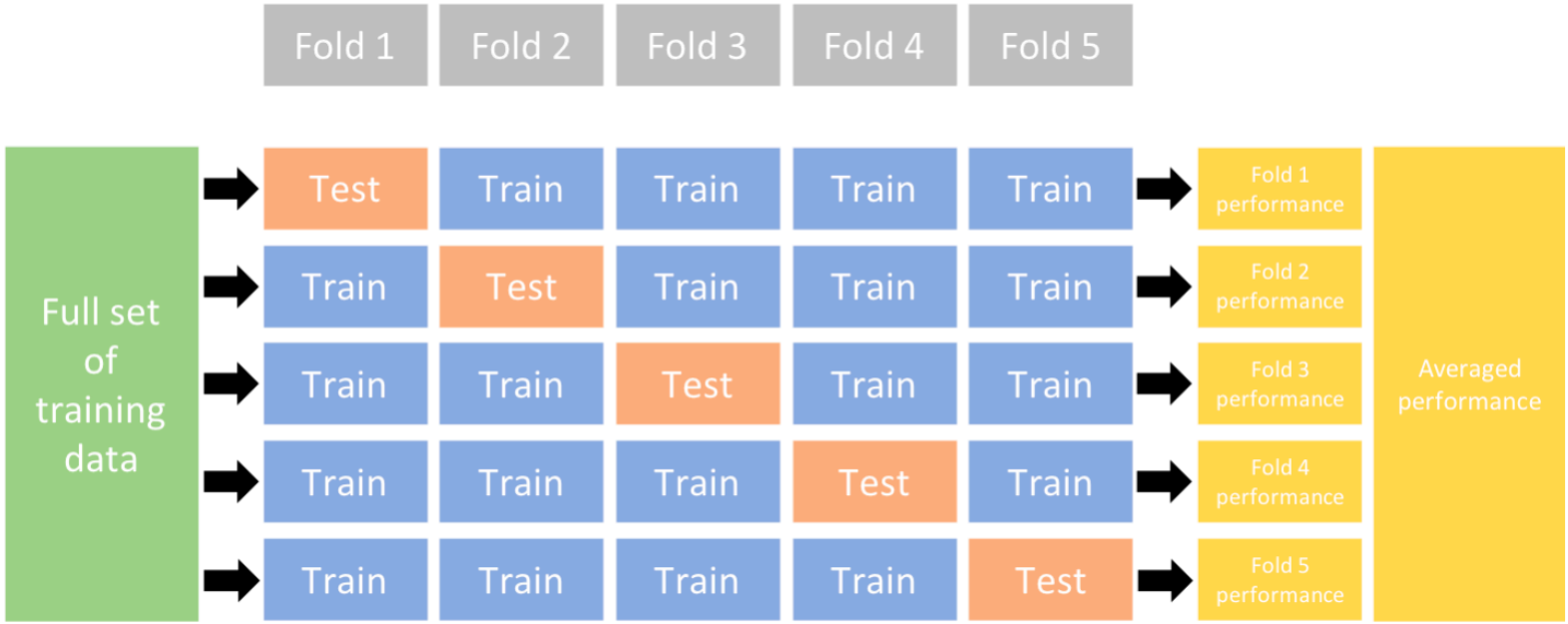
Folds





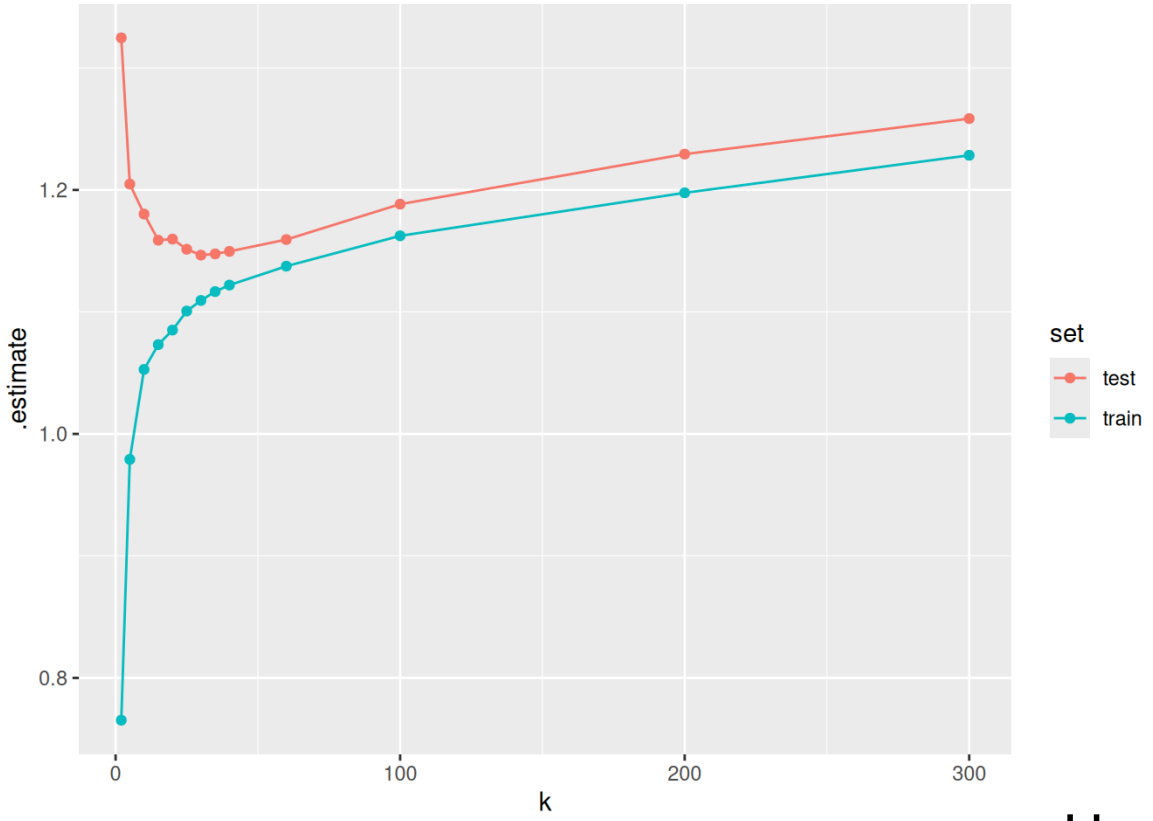
…

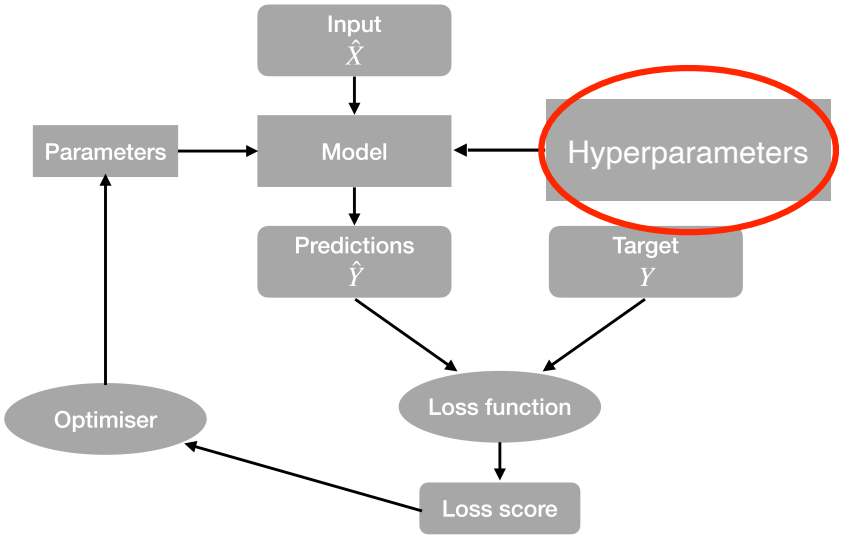
5-fold cross-validation



Boehmke & Greenwell (2019) *Hands On Machine Learning in R*

Loss profile



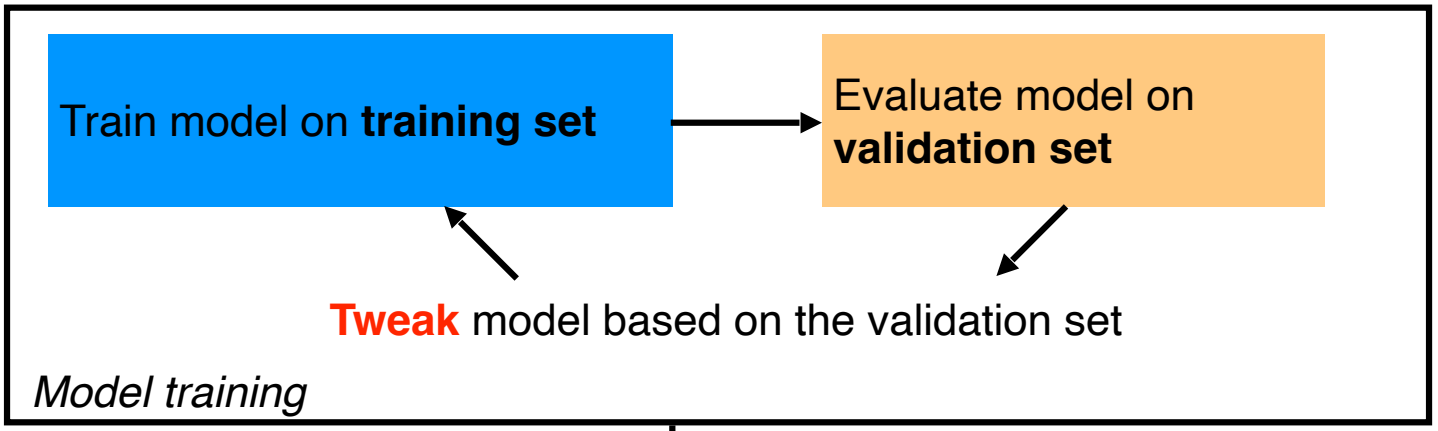


Hyperparameter

Hyperparameter tuning

- Hyperparameters determine the model structure and training setup.
- Hyperparameters are algorithm-specific.
  - *k* in KNN
  - min.node.size, mtry, splitrule in Random Forest (implementation using {ranger} with {caret})
- Not to confuse with model parameters/ coefficients.

Hyperparameter tuning



"Tweak":

- Find the best model structure, training setup, to optimise performance on validation set

Pick model with best performance on validation set.

Assess generalisability on **test set**

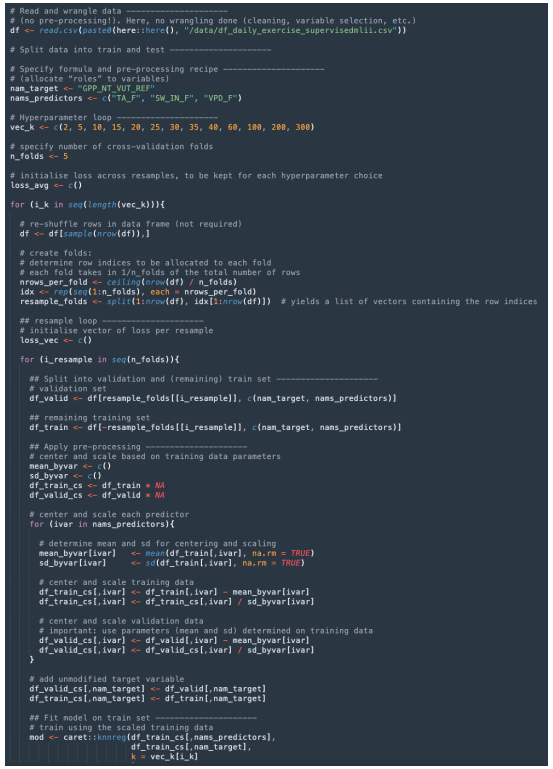
- **Hyperparameter tuning**
- Feature selection

Figure adopted form Google Machine Learning Crash Course

Hyperparameter tuning

The model training workflow

- Read and wrangle data (no pre-processing!)
- Split data into train and test
- Specify formula and pre-processing recipe (allocate “roles” to variables)
- *Hyperparameter loop*
    - *Resample loop*
        - Split into validation and (remaining) train set
        - Apply pre-processing
          (transform train and validation set based on parameters determined on train set)
        - Fit model on train set
        - Predict on validation set
        - Get loss from prediction on validation set
    - Average loss across resamples
- Pick best hyperparameter based on average loss across resamples



Switch to:
`add_material/full_workflow_no_caret.R`

Specifying recipes

The model training workflow

- Read and wrangle data (no pre-processing!)
- Split data into train and test
- Specify formula and pre-processing recipe
- *Hyperparameter loop*
    - *Resample loop*
        - Apply pre-processing (transform train and validation set based on parameters determined on train set)
        - Fit model on train set
        - Predict on validation set
        - Get loss from prediction on validation set
    - Average loss across resamples
- Pick best hyperparameter based on average loss across resamples

How many times is pre-processing transformation applied here?

```
pp <- recipes::recipe(GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F,
                      data = df) |>
  recipes::step_center(all_numeric(), -all_outcomes()) |>
  recipes::step_scale(all_numeric(), -all_outcomes())

mod <- caret::train(
  pp,
  data = df %>%
    drop_na(),
  method = "ranger",
  metric = "RMSE",
  trControl = trainControl(
    method = "cv",
    number = 5,
    savePredictions = "final"
    ),
  tuneGrid = expand.grid(
    .mtry = seq(from = 3, to = 19, by = 2),
    .min.node.size = 5:9,
    .splitrule = "variance"
  ),

  # arguments specific to "ranger" method
  replace = FALSE,
  sample.fraction = 0.5,
  num.trees = 12,
  seed = 1982
)
```

Specifying recipes

```
pp <- recipes::recipe(GPP_NT_VUT_REF ~ SW_IN_F + VPD_F + TA_F,
                      data = df) |>
  recipes::step_center(all_numeric(), -all_outcomes()) |>
  recipes::step_scale(all_numeric(), -all_outcomes())
```

```
> pp
Recipe

Inputs:

      role #variables
   outcome          1
 predictor          3

Operations:

Centering for all_numeric(), -all_outcomes()
Scaling for all_numeric(), -all_outcomes()
```

From Chapter 10 *Supervised ML I:*

To actually transform the data, we first have to "prepare" the recipe:

```
pp_prep <- recipes::prep(pp, training = daily_fluxes_train)
```

Finally we can actually transform the data. That is, "juice" the prepared recipe.

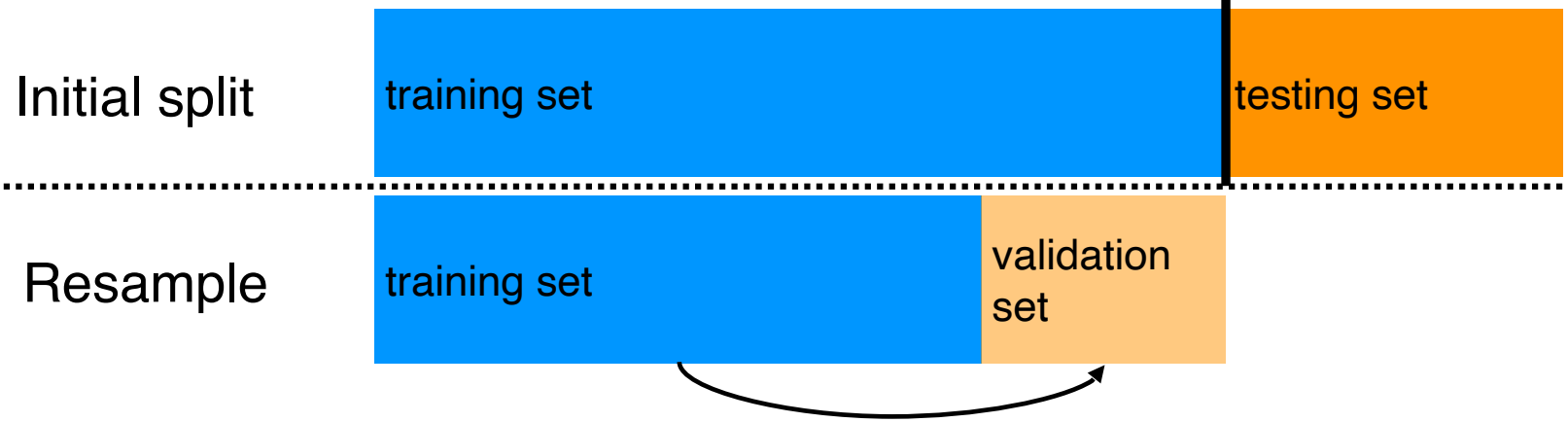
```
daily_fluxes_juiced <- recipes::juice(pp_prep)
```

Note, if we are to apply the prepared recipe to *new* data, we'll have to `bake()` it.

```
daily_fluxes_baked <- recipes::bake(pp_prep, new_data = daily_fluxes_train)

# confirm that juice and bake return identical objects when given the same data
all_equal(daily_fluxes_juiced, daily_fluxes_baked)
```

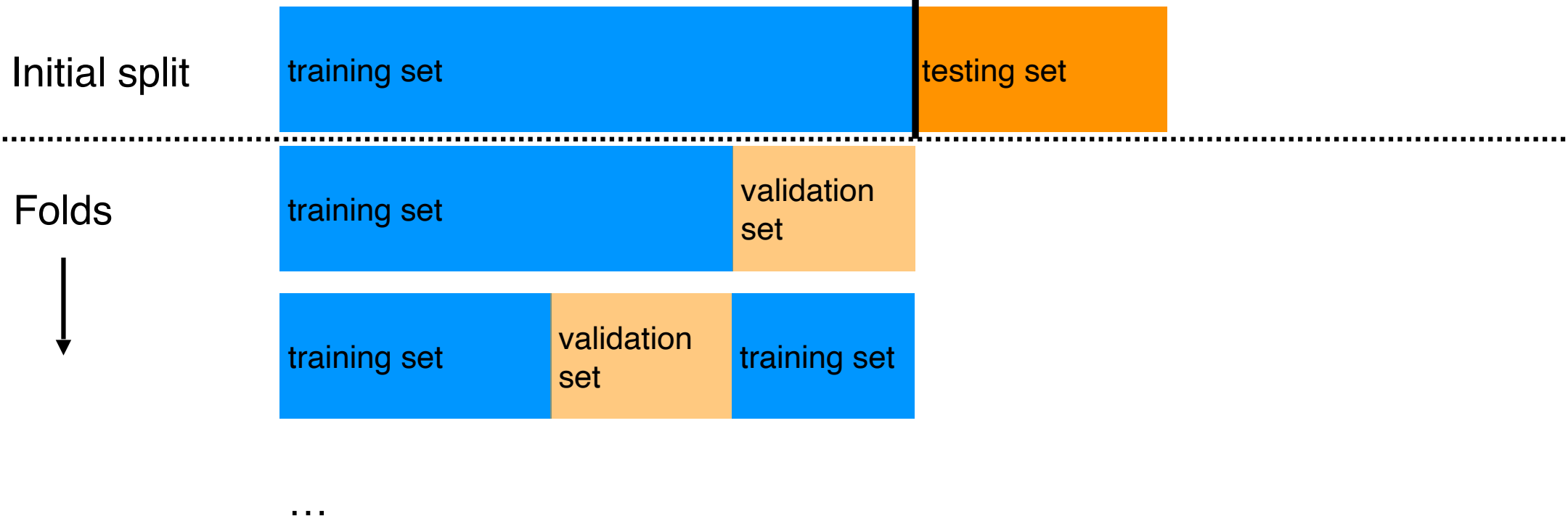
```
## [1] TRUE
```

Testing vs. validation set



- Hyperparameter tuning
- Feature selection

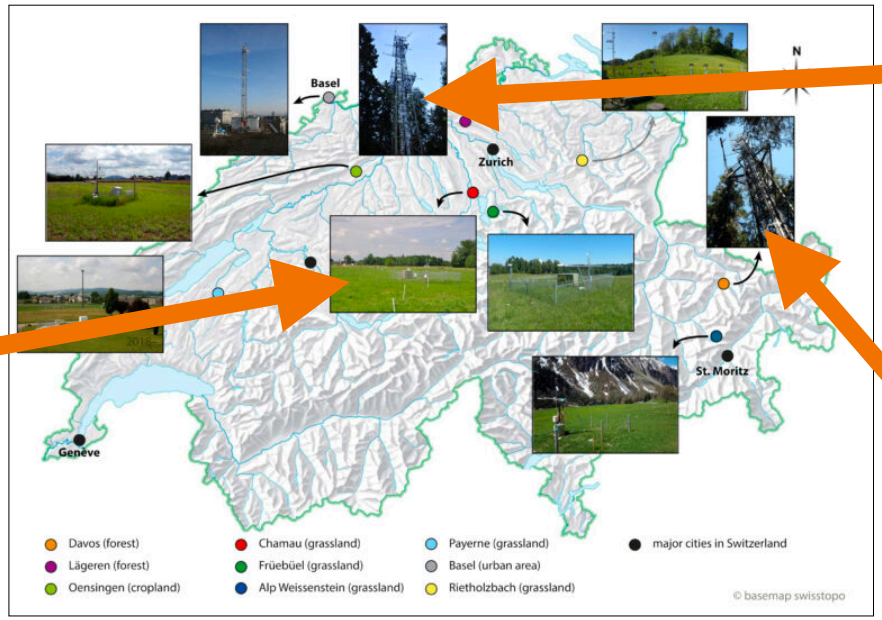
K-fold cross validation



Report Exercise 10.4: Spatial Upscaling

The data used for model training depends on our research question!

- Do we want a model that predicts GPP across Switzerland?
- Do we want a model that captures GPP responses of a deciduous forest?
- ...



Mixed Deciduous Forest

Grassland

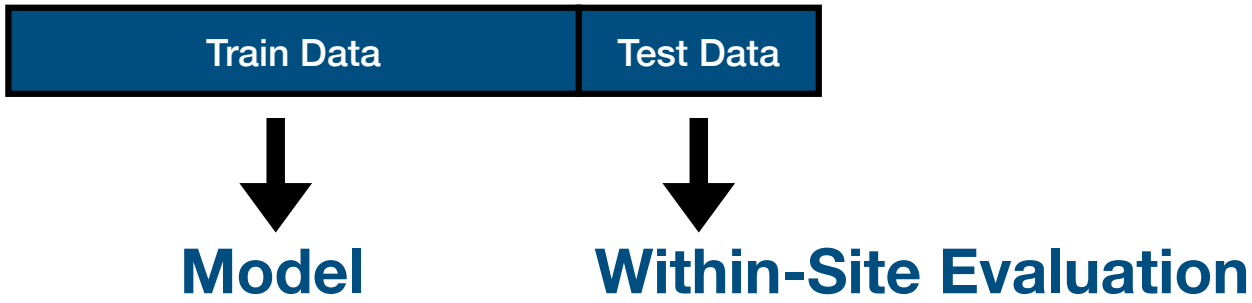
Coniferous Forest

Report Exercise 10.4: Spatial Upscaling

Training and testing on one site (Within-Site Evaluation)

Training on 1 site:

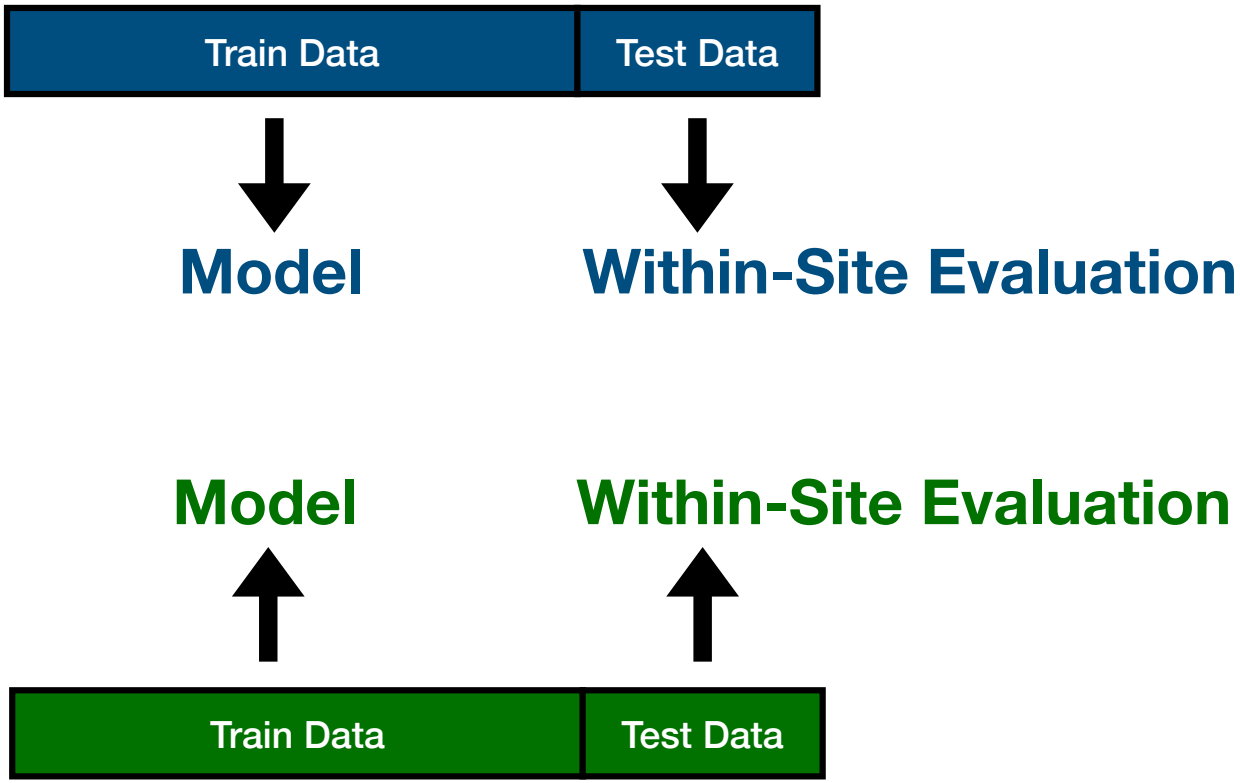
- Target is to predict response of **this site to unseen** environmental conditions
- Model is trained on time-series at that site
- Captures site-specific characteristics



Report Exercise 10.4: Spatial Upscaling

How generalisable is our model?

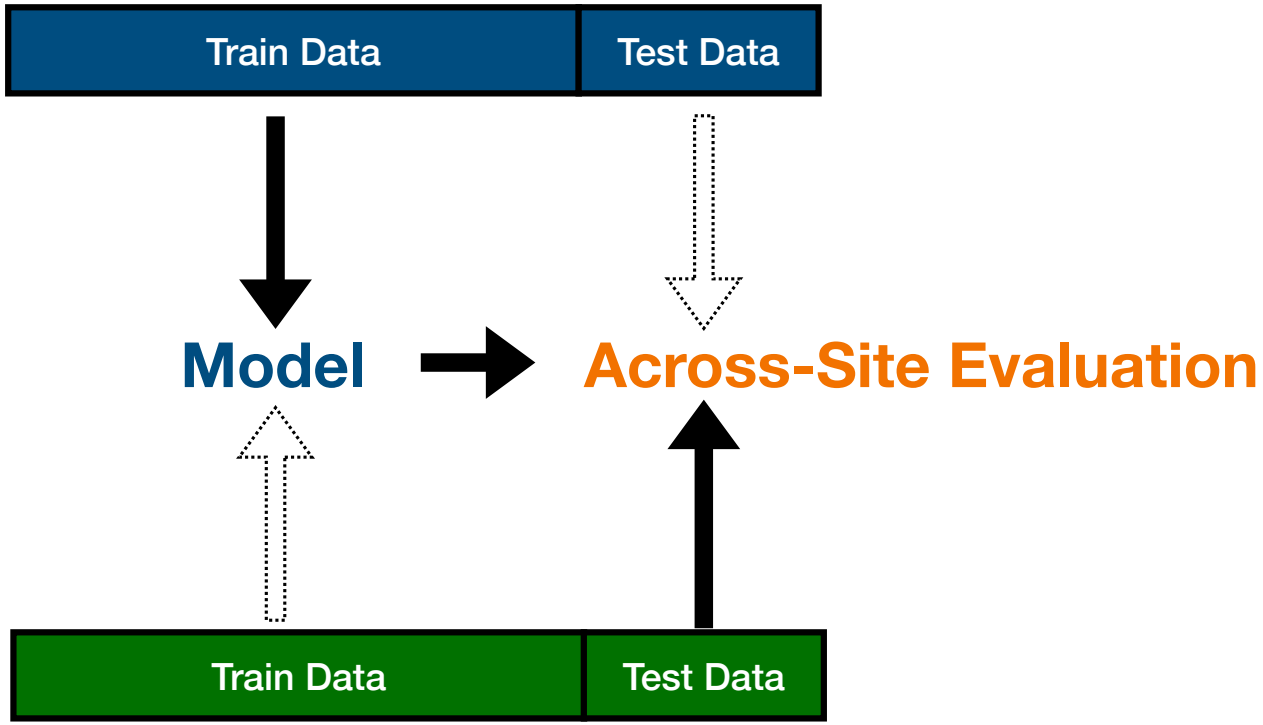
Compare within-site predictions …
considering different metrics



Report Exercise 10.4: Spatial Upscaling

How generalisable is our model? → Out-of-sample Prediction

For across-site predictions, make sure to implement a train and test setup that enables a true out-of-sample prediction test.



Report Exercise 10.4: Spatial Upscaling

Capturing spatial variability - training on two sites

Train a single model with training data pooled from both sites and predict with this single model on the test data of both sites.