An Introduction to Machine Learning in R

Workshop

Timo Koch, Jan Digutsch, Max Bergmann

21. September 2025

Welcome!

You can find all materials for this workshop here:
https://github.com/DreamTeamSG/ml_workshop/

This workshop is based on the following tutorial paper:
Pargent F., Schoedel R., Stachl C. (2023). Best Practices in Supervised Machine Learning: A Tutorial for Psychologists. *Advances in Methods and Practices in Psychological Science, 6(3)*. doi:10.1177/25152459231162559

These workshop materials are a "living" joint project with
Ramona Schoedel, **Florian Pargent**, & **Clemens Stachl**.

Agenda

Programm for today

Time	Session	Duration
9:00 AM – 9:25 AM	Welcome & Get-to-know	25 min
9:25 AM – 9:50 AM	Introduction to Machine Learning	25 min
9:50 AM – 10:35 AM	Performance Evaluation	45 min
10:35 AM – 10:50 AM	Coffee Break	15 min
10:50 AM – 12:00 PM	Coding Session I	70 min
12:00 PM – 1:00 PM	Lunch Break	60 min
1:00 PM – 1:45 PM	Model Overview	45 min
1:45 PM – 2:30 PM	Coding Session II	45 min
2:30 PM – 2:45 PM	Coffee Break	15 min
2:45 PM – 3:45 PM	Model Interpretation	60 min
3:45 PM – 4:00 PM	Coffee Break	15 min
4:00 PM – 4:45 PM	Coding Session III	45 min
4:45 PM – 5:00 PM	Summary & Closing Remarks	15 min

Introduction to Machine Learning

Applications in Personality Psychology

Machine learning methods are increasingly being used for personality psychological research questions.

Examples for predictive modeling:

- Facebook activity data -> personality traits
  (e.g., Kosinski, Stillwell, and Graepel 2013; Youyou, Kosinski, and Stillwell 2015)
- Facebook language -> affective states
  (Eichstaedt and Weidman 2020)
- Demographics & performance -> student retention
  (Matz et al. 2023)
- Online banking records -> personality traits
  (Gladstone, Matz, and Lemaire 2019)
- Psychological panel data -> sick days, life satisfaction, ...
  (Pargent and Albert-Von Der Gönna 2018)
- ...

Applications in Personality Psychology

. . . with mobile sensing data:

- situational characteristics (Schoedel et al. 2023)
- personality states (Rüegger et al. 2020)
- personality traits (Stachl, Au, et al. 2020)
- life outcomes (Digutsch et al. 2025)
- satisfaction with life (Bergmann et al. 2025)
- authoritarianism (Koch et al. 2025)

Predictive Modeling Mindset

We can distinguish between two modeling cultures (Breiman et al. 2001):

- Classical
  - data can be described with a stochastic model
  - "we assume a linear relationship..."
  - model-evaluation: goodness-of-fit indices (R^2, AIC, etc.), effect direction, residuals
  - sometimes good for explanation
  - often bad for prediction
- Algorithmic
  - data is created by an unknown, potentially complex process
  - find a function that can predict the target with high accuracy
  - model-evaluation = prediction accuracy
  - often good for prediction
  - often (still) bad for explanation

Predictive Modeling Mindset

Explanation vs. Prediction

- Social science's heavy focus on explanation (Yarkoni and Westfall 2017)
- Models often not adequately validated
- "... irrelevant theory and questionable conclusions ..." (Breiman et al. 2001)
- High-dimensional data - complex relationships, hard to hypothesize
- Create new measures, reflect on and improve existing theories
- See if theories predict relevant criteria (Shmueli 2010)

Predictive Modeling Mindset

Summary

- **Goal:** Make data-driven predictions/decisions
- **Focus:** Predicting new, unseen observations
- Examples:
  - predict rent for new flats
  - recognize individual differences in user data
  - identify faces/objects in images
  - recognize words/language in audio data

Basic Terminology

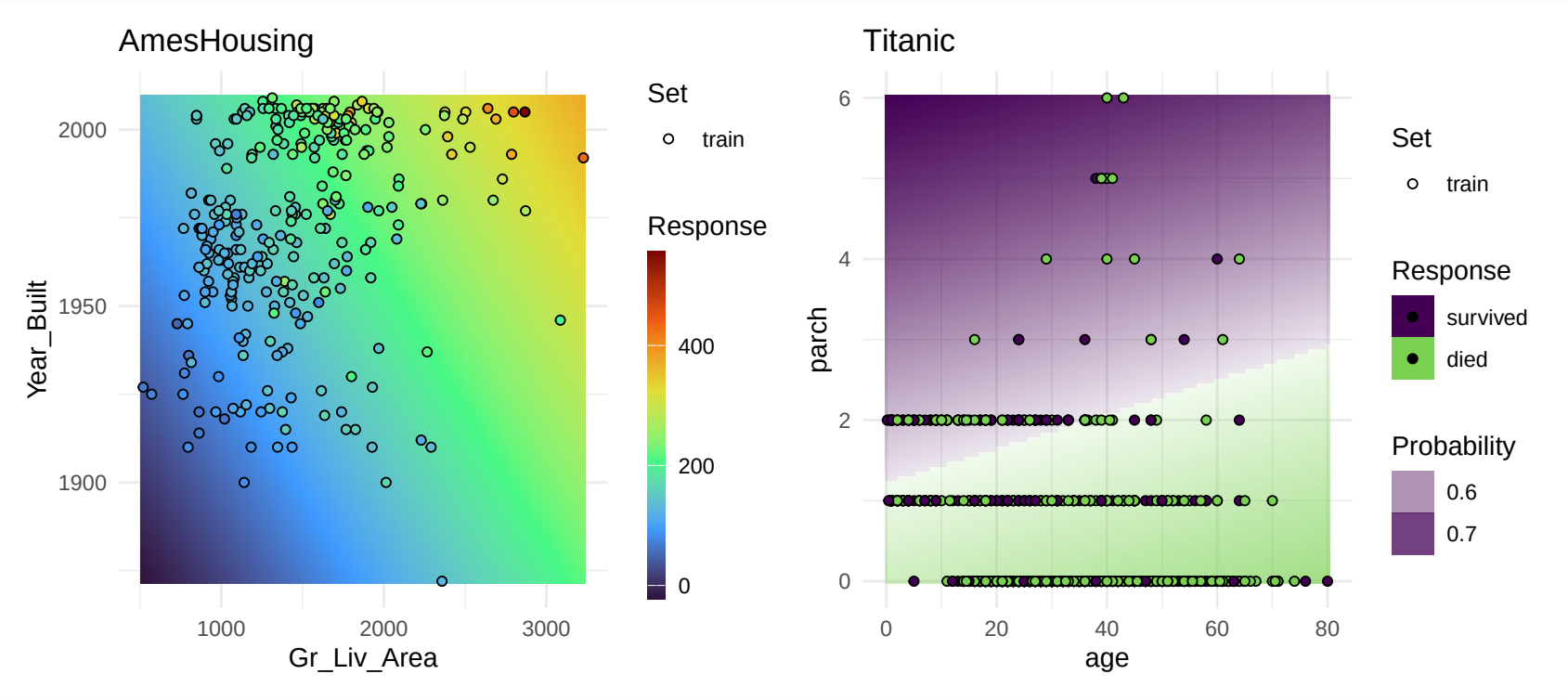
The 1x1 of Supervised Machine Learning Vocabulary

- **Target**: variable of interest (e.g., personality traits score)
- **Features**: predictor variables (e.g., smartphone usage behaviors)

Basic Terminology

The 1x1 of Supervised Machine Learning Vocabulary

- **Task**:
  - Regression: for continuous targets
  - Classification: for categorical/binary targets



Basic Terminology

The 1x1 of Supervised Machine Learning Vocabulary

- **Predictive Model**:
  - Any (statistical) model that generates (accurate) predictions of some target variable, based on (a series of) features (Kuhn and Johnson 2013; Pargent, Schoedel, and Stachl 2023)
  - Synonym: Machine Learning Model
  - Examples:
  - ordinary linear regression
  - penalized linear models: lasso, ridge, elastic net
  - tree models: decision tree, random forest, gradient boosting
  - support vector machines
  - deep neural networks
  - ...

Basic Terminology

The 1x1 of Supervised Machine Learning Vocabulary

- **Predictive Model**
- **Model Parameters**:
  - Different types of models differ in their type of parameters (e.g., β coefficients in linear regression)
  - They have to be estimated before a model can make predictions
- **Algorithms**:
  - Formal set of rules used to estimate appropriate values for the model parameters from data (e.g., least squares algorithm)

Data & methods used today

Data Set Used

In this workshop, we use a publicly available mobile sensing data set:

- Predicting personality from patterns of behavior collected with smartphones (Stachl, Au, et al. 2020, PNAS)
- Self-report questionnaire data:
  - German Big Five Structure Inventory (BFSI) (Arendasy, Sommer, and Feldhammer 2011)
  - Demographics (age, gender, education)
- Mobile sensing data: recorded for 30 days on the smartphone
  - communication & social behavior, app-use, music consumption, overall phone use, day-nighttime activity

Data & methods used today

Software Used

Machine learning meta packages in R:

- **mlr3** package (Lang et al. 2019):
  - standardized interface for machine learning
  - detailed tutorial at https://mlr3book.mlr-org.com/
  - mlr-org packages: mlr3pipelines, mlr3oml, ...
  - mlr3 is written in R6 (more about that later!)

- Alternative: **tidymodels** packages (Kuhn and Wickham 2018)

Performance Evaluation

Performance Evaluation

The quality of a trained predictive model is evaluated based on its magnitude of **error on new (unseen) data**, drawn from the same population:

> *"How well does a predictive model I have already estimated work when I use it to predict observations from my practical application, in which I do not know the target values?"*

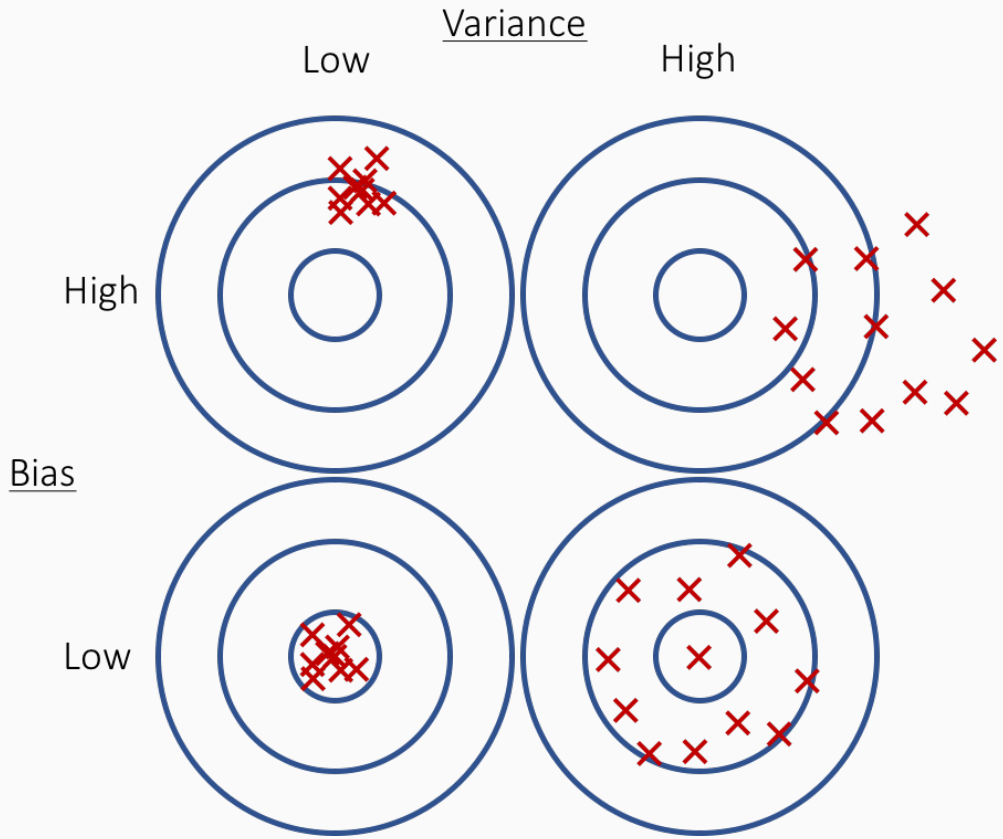
First things first: How to quantify the quality of models?

The Bias-Variance Trade-Off

Helpful mental model to reflect on which factors influence the performance of a predictive model in theory:

Prediction Error = f(Bias, Variance, Noise)

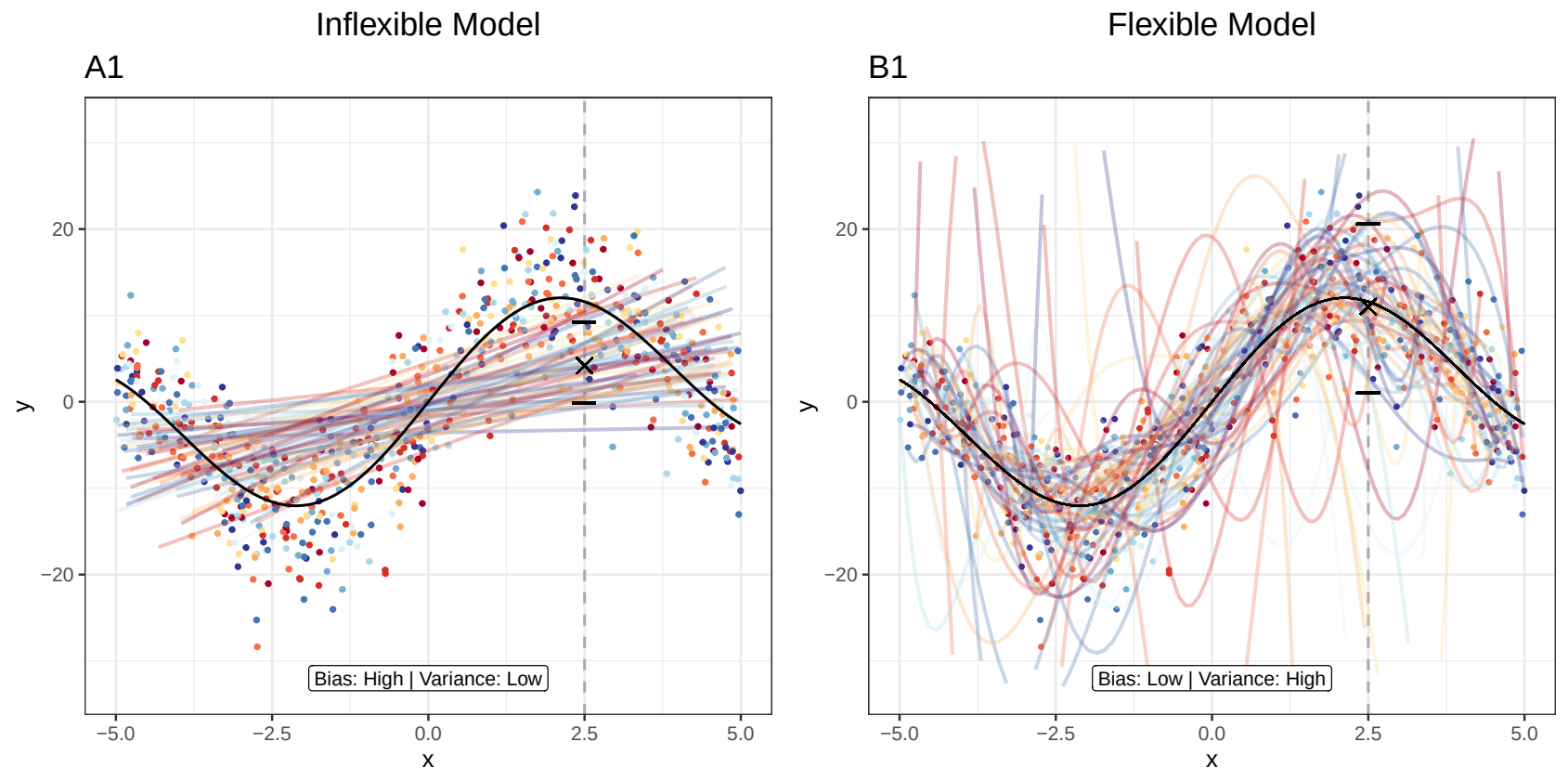
The Bias-Variance Trade-Off



- **Bias:** deviation of the average prediction from the true value
- **Variance:** variability of predictions based on different samples
- **Noise:** irreducible error of the true population model
- **Goal: Find a predictive model with low bias AND low variance**

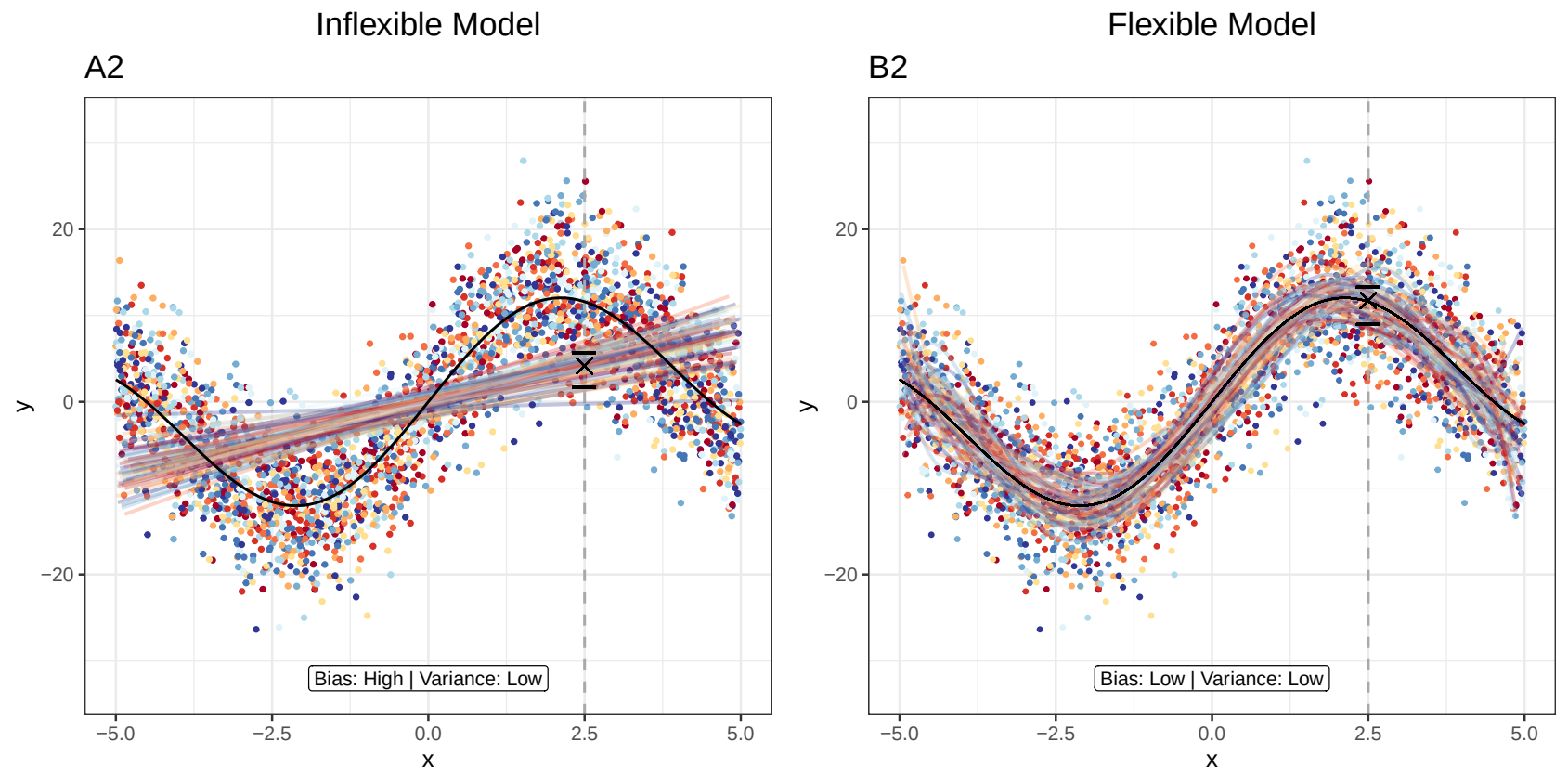
The Bias-Variance Trade-Off

Example 1: 60 Samples with $\mathbf{N = 12}$ each



The Bias-Variance Trade-Off

Example 2: 60 Samples with $\mathbf{N = 50}$ each



Performance Measures

Regression Tasks

Idea behind: Quantify a "typical" deviation from the true value

The statistician's favorite:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The social scientist's favorite:

$$R^2 = 1 - \frac{\textit{residual sum of squares}}{\textit{total sum of squares}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Performance Measures

Classification Tasks

Idea behind: Quantify the proportion of misclassified cases in all cases

Mean misclassification error ($\widehat{=}$ MSE)

$$MMCE = \frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

Performance Measures

Classification Tasks

Confusion Matrix		Truth y_i	
		1	0
Prediction $\hat{y}_i$	1	TP	FP
	0	FN	TN

- Sensitivity = TP/(TP+FN)
- Specificity = TN/(TN+FP)
- Positive predictive value = TP/(TP+FP)
- Negative predictive value = TN/(TN+FN)

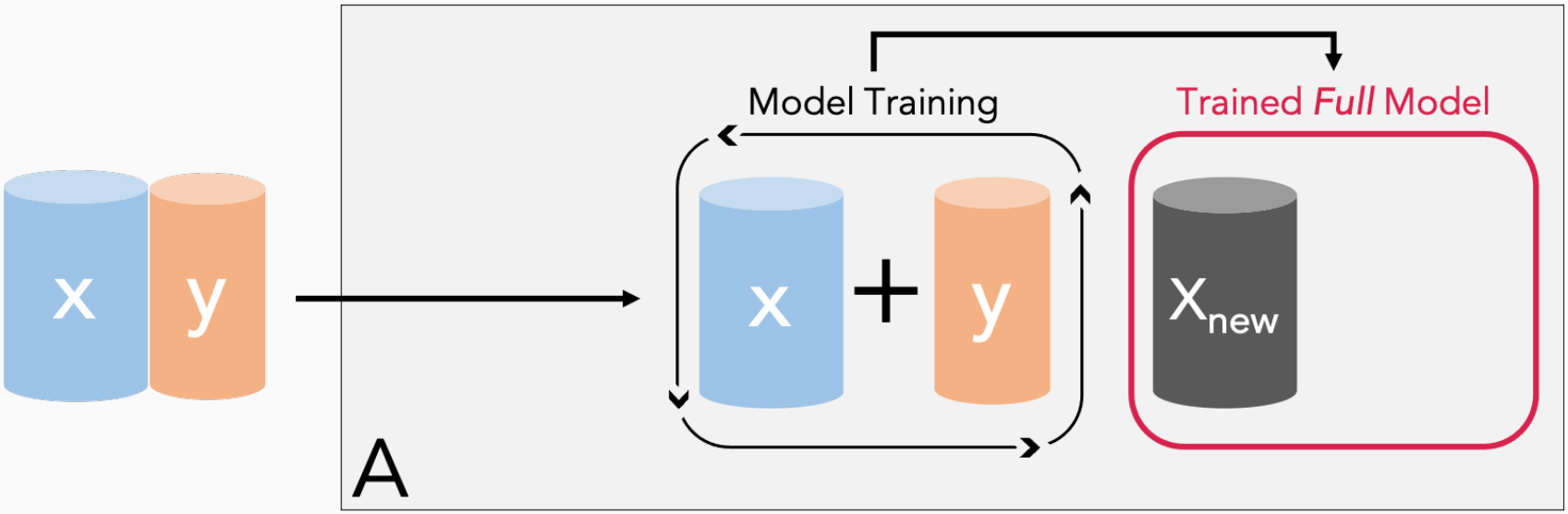
Resampling Strategies for Model Evaluation

The Idea Behind Resampling

- How well does our model predict **new data**?
- But how to get **new data**?
  - Option 1: collect new data **;-)**
  - Option 2: use prediction error in-sample **:-(**
  - Option 3: use available data in a smart way **:-)**
- Resampling = **Smart Recycling**!
- That is, to estimate the performance of our model, split the data set:
  - **Training set**: train the model
  - **Test set**: compute performance

The Idea Behind Resampling

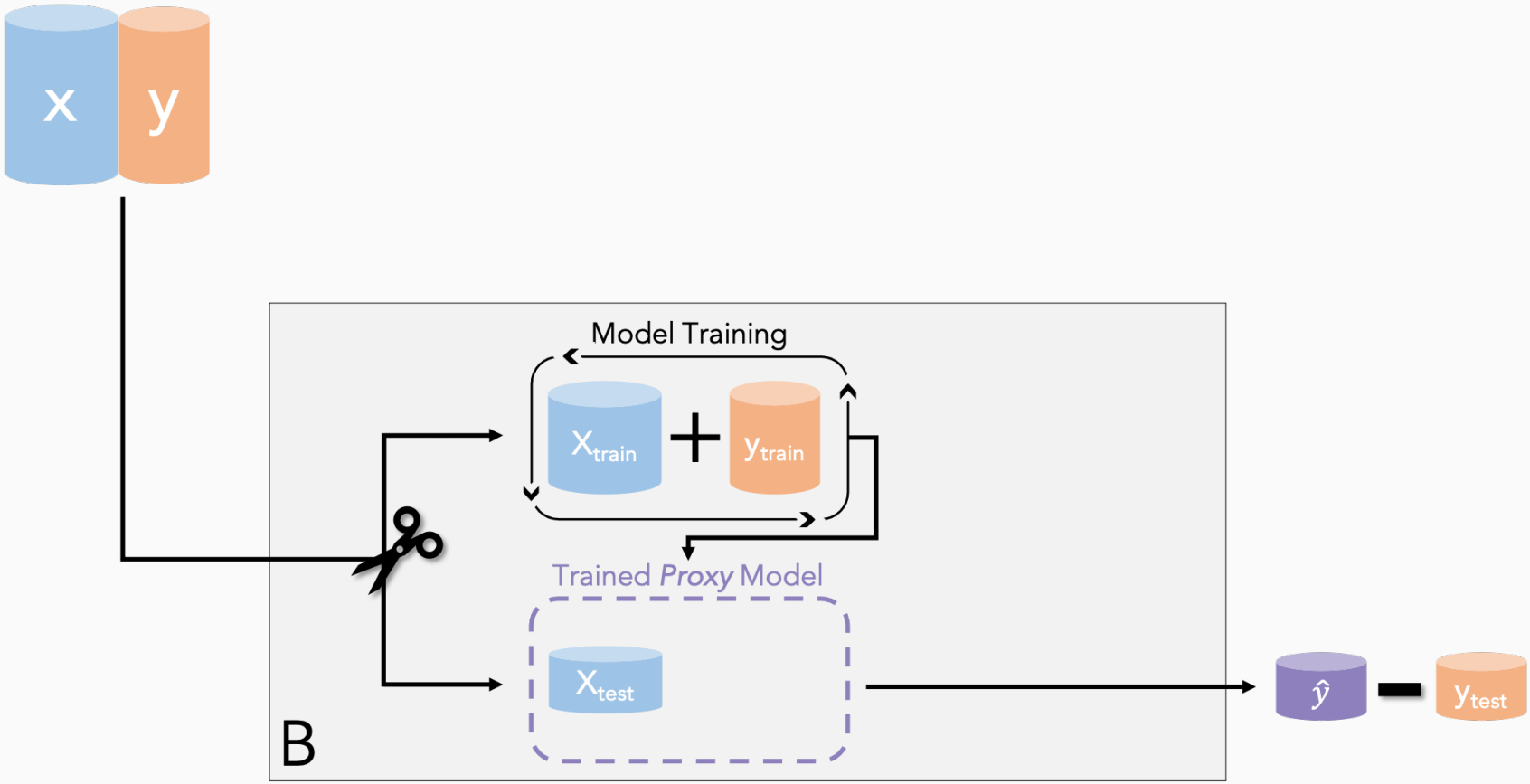
Learn a functional relationship between **X** and **y**:



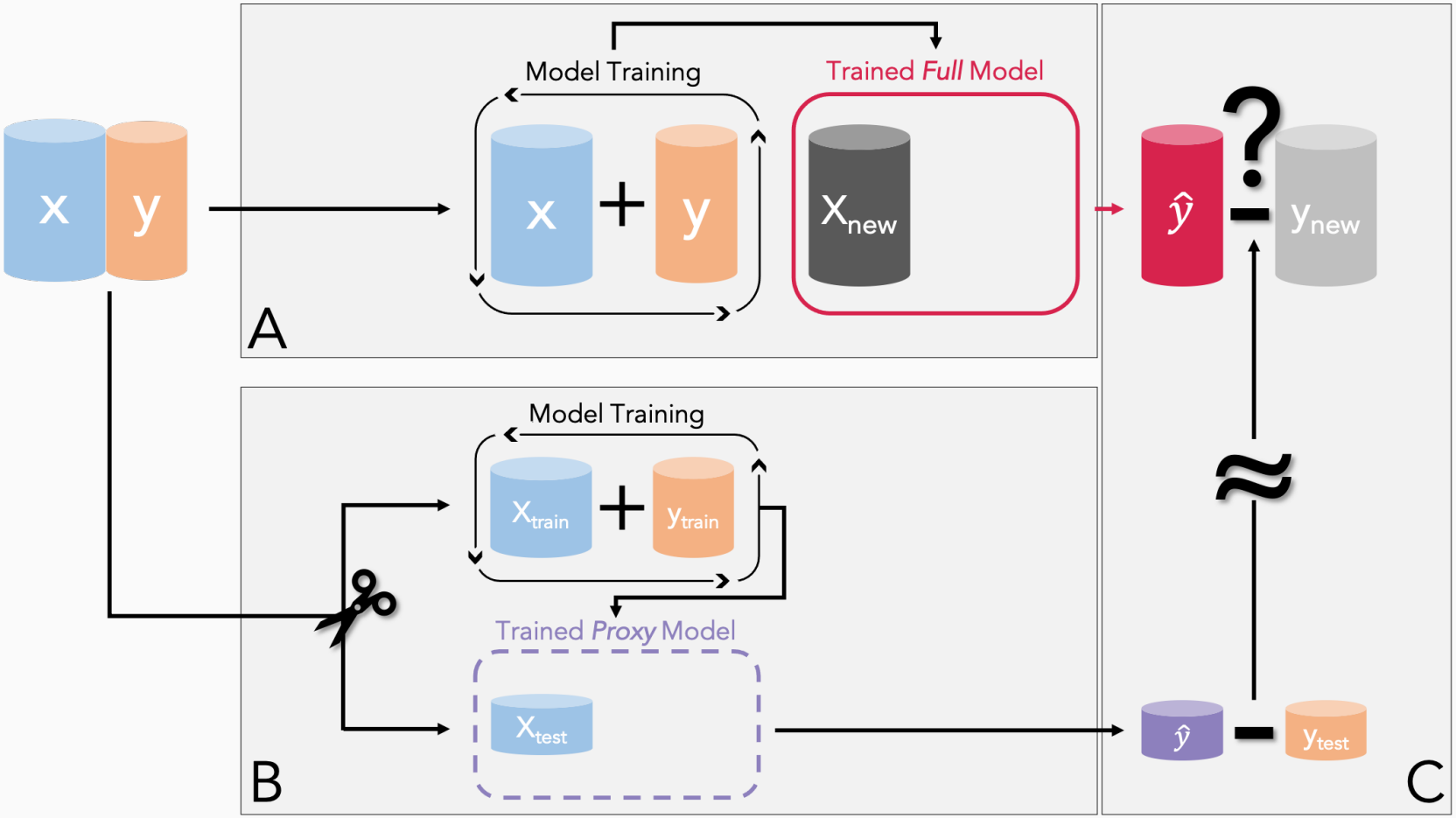
Central question: How does the model perform during application?

Resampling Strategies for Model Evaluation

The Idea Behind Resampling



The Idea Behind Resampling



Resampling Strategies for Model Evaluation

IMPORTANT NOTE: Do not get confused by the different models!

Full Model:

- trained on the **whole dataset**
- will be used in **practical applications**

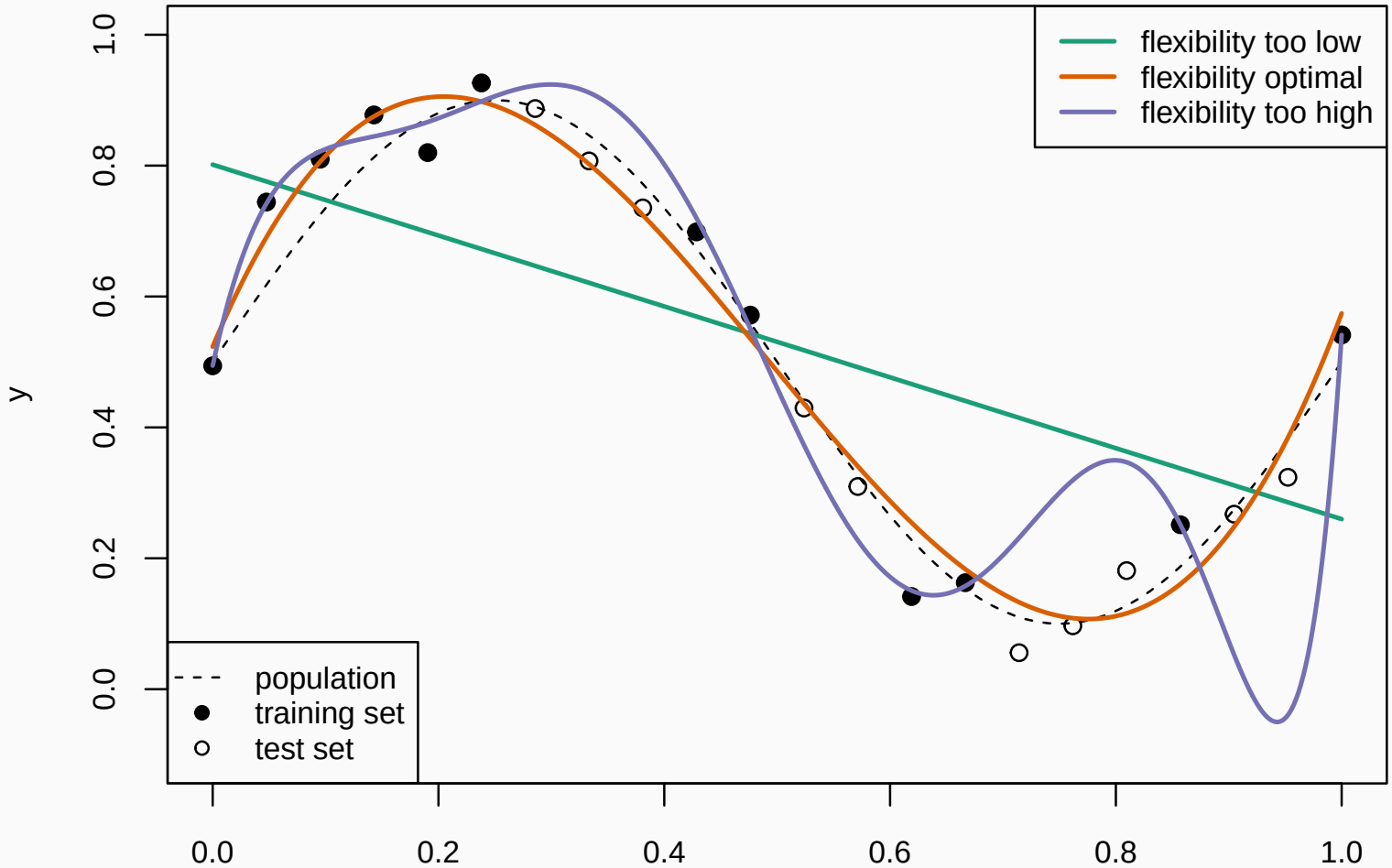
Proxy Model:

- trained on a **training set**
- is only a **tool for performance estimation**
- can be **discarded** after test set predictions

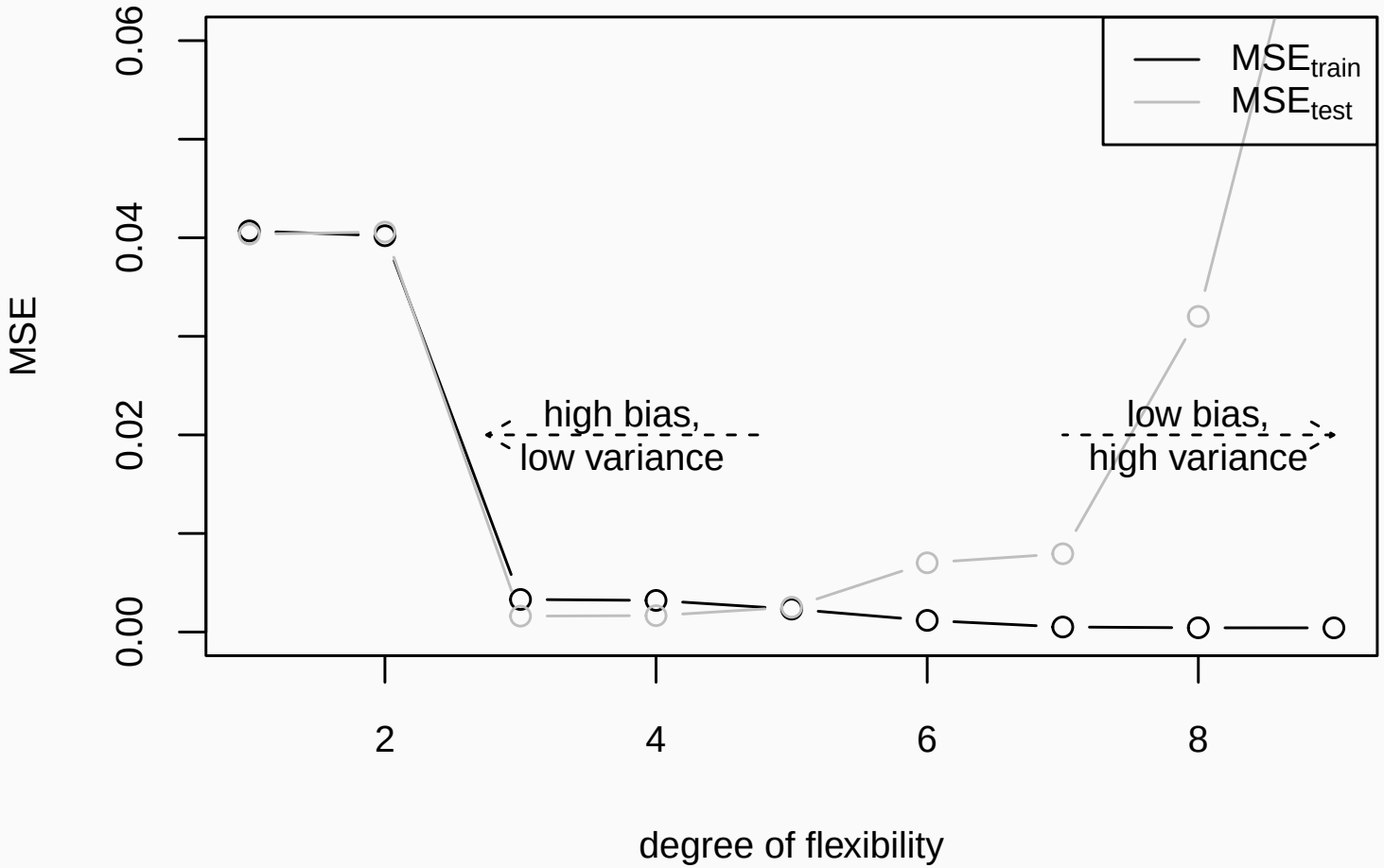
Why Do We Have to Separate Training and Test Data?

- To avoid getting fooled by **Overfitting**:
  - Model adjusts to a set of given data points too closely
  - Sample specific patterns are learned ("fitting the noise")
  - Can be compared to "learning something by heart"
- Many flexible models predict training data (almost) perfectly:
  - Training ("in-sample") performance is useless to judge the model's performance on new data ("out-of-sample")!

The Concept of Overfitting



The Concept of Overfitting



Resampling Strategies for Model Evaluation

Training Set vs. Test Set – Dilemma

- Training set
  - as large as possible, else performance is underestimated
  - large enough to learn well
  - rule of thumb: 2/3
- Test set
  - as large as possible, else high variance in performance estimates
  - large enough for stable performance evaluation
  - rule of thumb: 1/3
- [Validation set (only if you have many many observations!)]
  - evaluation of the best final model on unused data

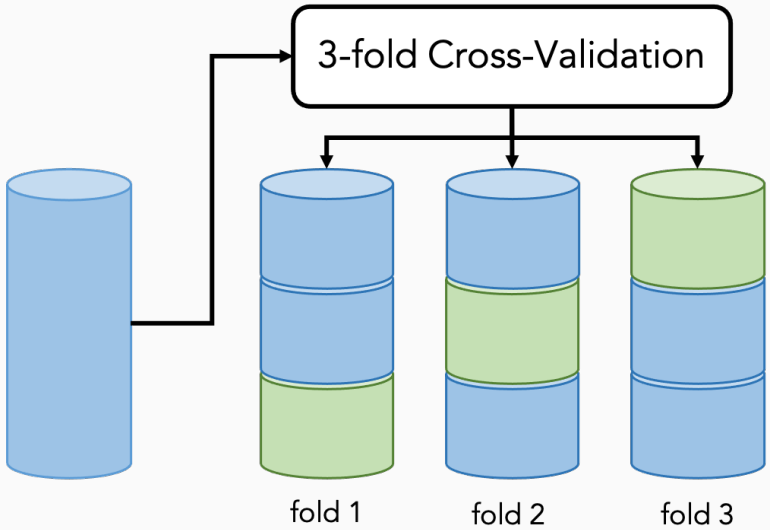
Resampling Strategies for Model Evaluation Bischl et al. (2012)

- The resampling strategy that we have learned about so far is called **holdout**
- But we can optimize the partitioning of the data set even further:
  - several splits in training/test sets
  - aggregation of results
- Resampling methods:
  - **cross-validation (CV)**
  - repeated CV
  - leave-one-out CV
  - bootstrap
  - subsampling

Resampling Strategies for Model Evaluation

Cross-Validation (CV)

- **Bias reduction** via big training sets
- **Variance reduction** via aggregation
- Random partitioning in k equally sized parts (often 5 or 10)
- Each part test set once, remaining parts combined training set
- Average the estimated prediction error from all *folds*



Let's Take a Break!

Picture by Kaboompics.com: https://www.pexels.com/de-de/foto/kaffee-tasse-becher-loffel-6398/

Coding Session I

Overview over data used today

Reminder: We use the PhoneStudy Behavioral Patterns Dataset (Stachl et al., 2020, PNAS):

- Self-report questionnaire data:
  - German Big Five Structure Inventory (BFSI) (Arendasy, Sommer, and Feldhammer 2011)
  - Demographics (age, gender, education)
- Mobile sensing data: recorded for 30 days on the smartphone
  - communication & social behavior, app-use, music consumption, overall phone use, day-nighttime activity

Overview over data used today

Our predictive modeling exercises with mlr3 are based on the following:

- **Sample**: $N = 620$ participants bundled from smaller studies (e.g., Stachl et al. 2017)
- **Features**: 1821 predictors (aggregated communication behavior); small subset of available sensor variables
- **Target**: Sociability facet of Extraversion
- **Task**: Regression (as our target is continuous)

Coding Session I

Preparation

```
# Download the data from
# https://osf.io/mnfbd/
phonedata = readRDS(file = "clusterdata.RDS")
# remove participants who did not indicate their gender
phonedata = phonedata[complete.cases(phonedata$gender),]
# remove variables we do not use in our tutorial
phonedata = phonedata[, c(1:1821,1837)]
```

Load **mlr3verse** - package.

```
library(mlr3verse)
```

Coding Session I

Exercise 1: Compute In-sample Performance the mlr3-Way

1. Fit a multiple linear regression model for the target variable **E2.Sociableness** (latent person parameter (PCM) of the Sociability facet)
2. Use all other variables as predictors
3. Check out the in-sample R^2

Create a *task* object which contains the necessary data.

```
task_Soci <- as_task_regr(
  phonedata,
  id = "Sociability_Regr",
  target = "E2.Sociableness"
)
```

Inspect the metadata of our task object.

```
task_Soci
```

```
-- <TaskRegr> (620x1822) ------------------------------------------------
* Target: E2.Sociableness
* Properties: -
* Features (1821):
  * dbl (1821): AR_num_calls_in1, AR_num_calls_in12, AR_num_calls_in24,
  AR_num_calls_in4, AR_num_calls_in8, AR_num_calls_out1, AR_num_calls_out12,
  AR_num_calls_out24, AR_num_calls_out4, AR_num_calls_out8, AR_num_calls_ring1,
  AR_num_calls_ring12, AR_num_calls_ring24, AR_num_calls_ring4,
  AR_num_calls_ring8, IVI_call_in, IVI_call_in_week, IVI_call_in_weekend,
  IVI_call_miss, IVI_call_miss_week, IVI_call_miss_weekend, IVI_call_out,
  IVI_call_out_week, IVI_call_out_weekend, IVI_call_ring, IVI_call_ring_week,
  IVI_call_ring_weekend, IVI_call_week, IVI_call_weekend, IVI_calls,
  IV_Academia, IV_Artistic_Hobby, IV_Beauty, IV_Betting_Risk, IV_Calculator,
  IV_Calendar_Apps, IV_Calling, IV_Camera, IV_Checkup_Monitoring,
  IV_ComicsBooks, IV_Dating_Mating, IV_E_Mail, IV_Eating, IV_Education,
  IV_Emergency_Warning, IV_Entertainment, IV_Financial, IV_Gallery,
  IV_Gaming_Action, IV_Gaming_Adventure, IV_Gaming_Casual, IV_Gaming_Knowledge,
  IV_Gaming_Logic, IV_Gaming_Role_Playing, IV_Gaming_Simulation,
  IV_Gaming_Sports, IV_Gaming_Strategy, IV_Gaming_Tools_Community,
  IV_Health_SelfMonitoring, IV_Image_And_Video_Editing, IV_Internet_Browser,
```

Create a *learner* object. Checkout the full list of **mlr3**'s learners at https://mlr3book.mlr-org.com/learners.html

```
lm <- lrn("regr.lm")
lm$id <- "linear_model"
```

Train the *learner* on the *task*.

```
lm$train(task = task_Soci)
```

```
Error in assert_task_learner(task, learner, param_values): Task 'Sociability_Re
```

```
# po defines a single pipeline operation
imputer <- po("imputemedian")
# combine po and learner into a pipeline
lm <- as_learner(imputer %>>% lm)
```

Train the *GraphLearner* on the *task*.

```
lm$train(task = task_Soci)
```

Make *predictions* on the same data used for training.

```
prediction <- lm$predict(task_Soci)
```

Compute in-sample *performance* (R^2 and *MSE*).

```
options(scipen = 999)
measures <- msrs(c("regr.rsq", "regr.mse"))
prediction$score(measures)
```

```
                 regr.rsq                   regr.mse
1.0000000000000000000000 0.0000000000000000000059
```

We could also inspect other performance measures. We can get an exhaustive overview of available measures via: `as.data.table(mlr_measures)`.

Coding Session I

Exercise 2: Cross-Validation

1. Check the **out-of-sample** performance (R^2)
2. Create a description of and perform the *resampling* - strategy.
3. Compare in-sample and out-of-sample prediction performance.

We use 5-fold *cross-validation* here.

```
rdesc <- rsmp("cv", folds = 5)
```

Set a *seed* to make your analysis reproducible.

```
set.seed(1)
```

Perform *cross-validation* and compute out-of-sample performance.

```
res <- resample(
  learner = lm,
  task = task_Soci,
  resampling = rdesc,
  store_models = TRUE)
res$aggregate(measures)
```

```
regr.rsq regr.mse
   -2341     6994
```

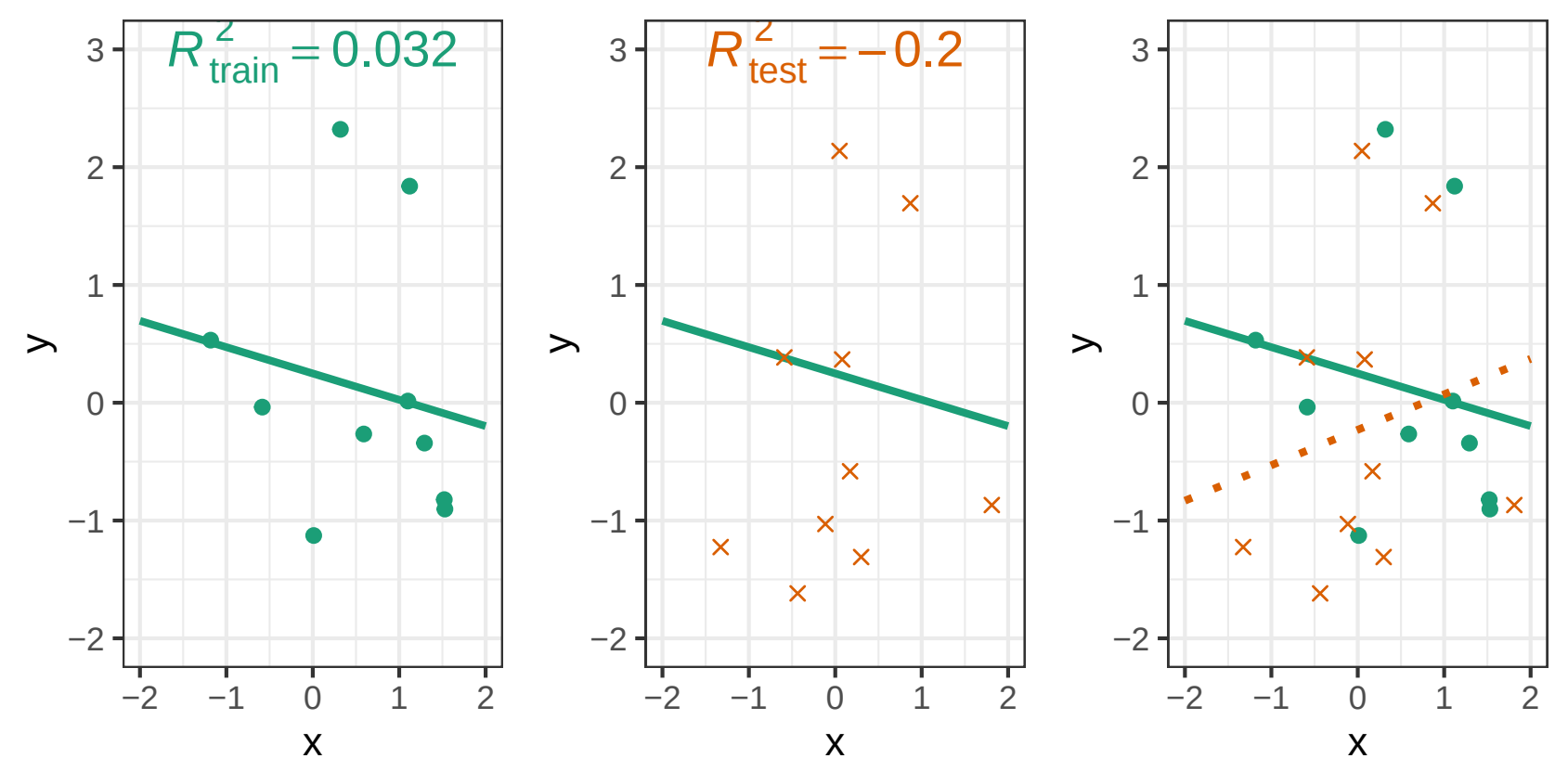
Compare our out-of-sample with the in-sample performance (s. Exercise 1).

```
matrix(c(prediction$score(measures), res$aggregate(measures)), nrow = 2,
  dimnames = list(c("R^2", "MSE"), c("in-sample", "cross-validation")))
```

```
                    in-sample cross-validation
R^2 1.000000000000000000000000            -2341
MSE 0.000000000000000000000059             6994
```

Resampling shows that the prediction does not work well ...

Negative R^2



- Train model on **training data** (positive R^2_{train})
- Predict **test data** with trained model (negative R^2_{test})

Some Advanced Aspects on Model Evaluation

Model Optimization

We often have to make many decisions in predictive modeling:

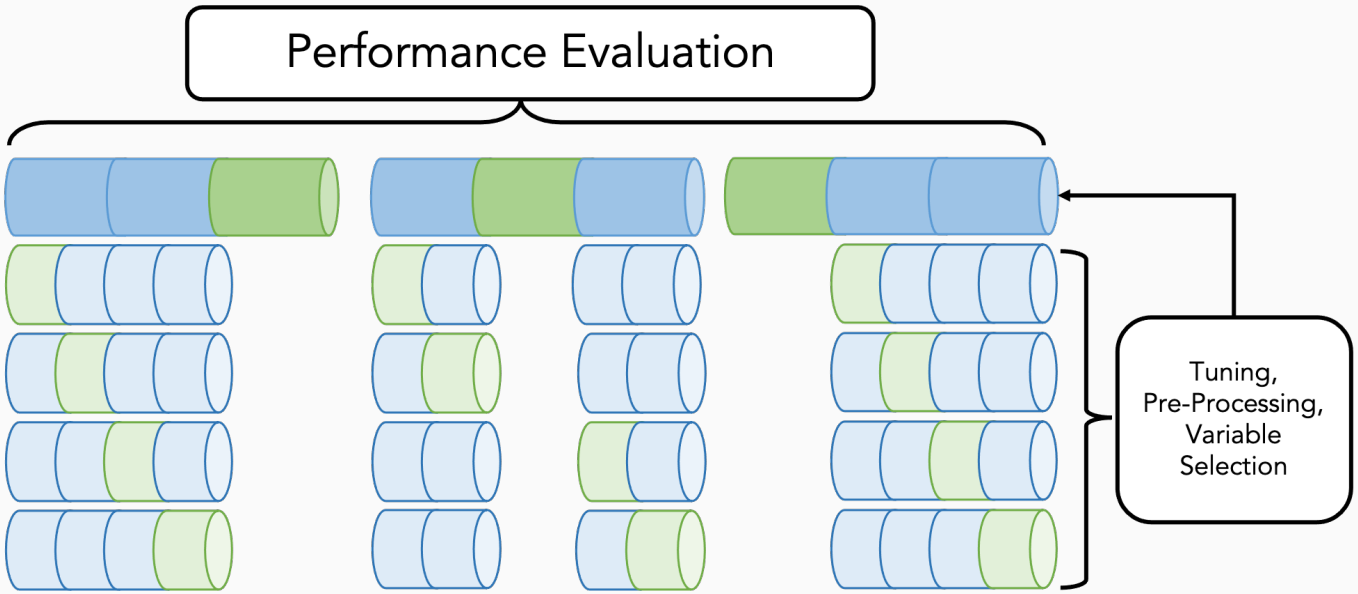
- Hyperparameter Tuning (e.g., *mtry* in random forests)
- Pre-processing (transformations, dimensionality-reduction, imputation)
- Variable selection (use only the best predictors?)

To prevent overestimated predictive performance (e.g., due to overfitting), implement those decisions into the resampling process (i.e., repeat each time the model is trained)
-> **Simulate what happens in model application!**

If the decision itself requires resampling
-> **Nested Resampling**

Nested Resampling

- Inner loop: tuning, pre-processing, variable selection
- Outer loop: evaluation of model performance
- **Goal:** separate data used in model training from data used in model evaluation



Some Advanced Aspects on Model Evaluation

Variable Selection

- Common pitfall in (social science) research:
  - correlate all predictors with the target in the complete dataset
  - choose only highly correlated predictors for modeling
- **Problem:** The decision of which variables to select is based on the complete dataset (training set + test set)
  –> **Overestimated performance**

Don't fool yourself! This shares similarities with:

- *multiple testing*
- *p-hacking*
- *HARKING*

Variable Selection: Illustration of Common Mistakes

WARNING: This is a recipe for potential disaster!

Simulate completely random data (same size as phonedata) and define an arbitrary target.

```
set.seed(1)
dat <- matrix(rnorm(prod(dim(phonedata))), nrow = nrow(phonedata),
              ncol = ncol(phonedata))
dat <- as.data.frame(dat)
colnames(dat)[ncol(phonedata)] <- "Target"
```

Compute correlations of all predictor variables with the target. Only retain the ten most correlated predictors, remove the others.

```
target_cors <- cor(dat)[-ncol(phonedata), "Target"]
ten_best_vars <- names(sort(abs(target_cors), decreasing = TRUE)[1:10])
task_10 <- as_task_regr(dat[,c(ten_best_vars, "Target")],
                        id = "ten_best", target = "Target")
```

Let's fit a simple regression model using those top-ten "best" variables to predict the target values, using the complete data set.

```
lm <- lrn("regr.lm")
lm$train(task_10)
preds <- lm$predict(task_10)
preds$score(msr("regr.rsq"))
```

```
regr.rsq
    0.11
```

Oh woooow: a positive R^2, although we know that there really is no relationship between the target and the features.

Let's try to be more "sophisticated" by using some intense, repeated CV.

```
rdesc <- rsmp("repeated_cv", folds = 10, repeats = 10)
res_overfit <- resample(lm, task = task_10, resampling = rdesc)
```

Amazing:

```
res_overfit$aggregate(msr("regr.rsq"))
```

```
regr.rsq
   0.058
```

R^2 is still positive! This must be an important discovery!

... No it is not. We just produced a much too optimistic performance estimate :-(

What did we do wrong here?

To obtain a realistic estimate of the predictive performance of our algorithm, the variable selection has to be integrated **into the resampling process**.

Therefore, we have to create a **GraphLearner** with a **Filter** that will choose the top-ten variables in each CV iteration (those can be different).

```
filter <- flt("correlation", method = "pearson")
filter <- po("filter", filter = filter)
filter$param_set$values$filter.nfeat <- 10
filtered_lm <- as_learner(filter %>>% lm)
```

Run the resampling with the **GraphLearner** on the complete dataset, using all variables.

```
task_1000 <- as_task_regr(dat, id = "all_vars", target = "Target")
res_nested <- resample(filtered_lm, task = task_1000, resampling = rdesc)
```

```
res_nested$aggregate(msr("regr.rsq"))
```

```
regr.rsq
   -0.22
```

Result: negative R^2... We did worse than predicting with the target mean of the respective test sets and ignoring all predictors.

Lunchbreak

Picture by Kaboompics.com: https://www.pexels.com/de-de/foto/kaffee-tasse-becher-loffel-6398/

Model Overview

1 Regularized Regression

Ordinary Linear Regression

- What you already know: estimate a linear relationship between predictors and an outcome.
- Works well when the number of predictors is small, predictors aren't too correlated, and overfitting risk is modest.
- Limitations in ML settings: can overfit with many predictors, struggles when predictors are highly correlated, and fails when p $>$ n (many features, few cases).

-> Regularization is a remedy (see James et al. (2013)).

Why regularize?

- Goal: better generalization and stability when you have many and/or correlated predictors.
- Intuition: regularized models discourage large coefficients, which reduces variance and overfitting.

1 Regularized Regression

Ridge Regression

- Shrinks all coefficients toward zero but rarely to exactly zero.
- Best when many predictors have small/medium effects and are correlated; tends to keep groups of correlated predictors together.
- No hard variable selection (everything stays in the model, just smaller).

Least Absolute Shrinkage and Selection Operator (LASSO)

- Does shrinkage and automatic variable selection: some coefficients become exactly zero (Tibshirani 1996).
- Good when you expect many predictors to be irrelevant; produces sparse models that are easy to explain.
- With highly correlated predictors, LASSO may pick one and drop the rest somewhat arbitrarily.

Elastic Net (EN)

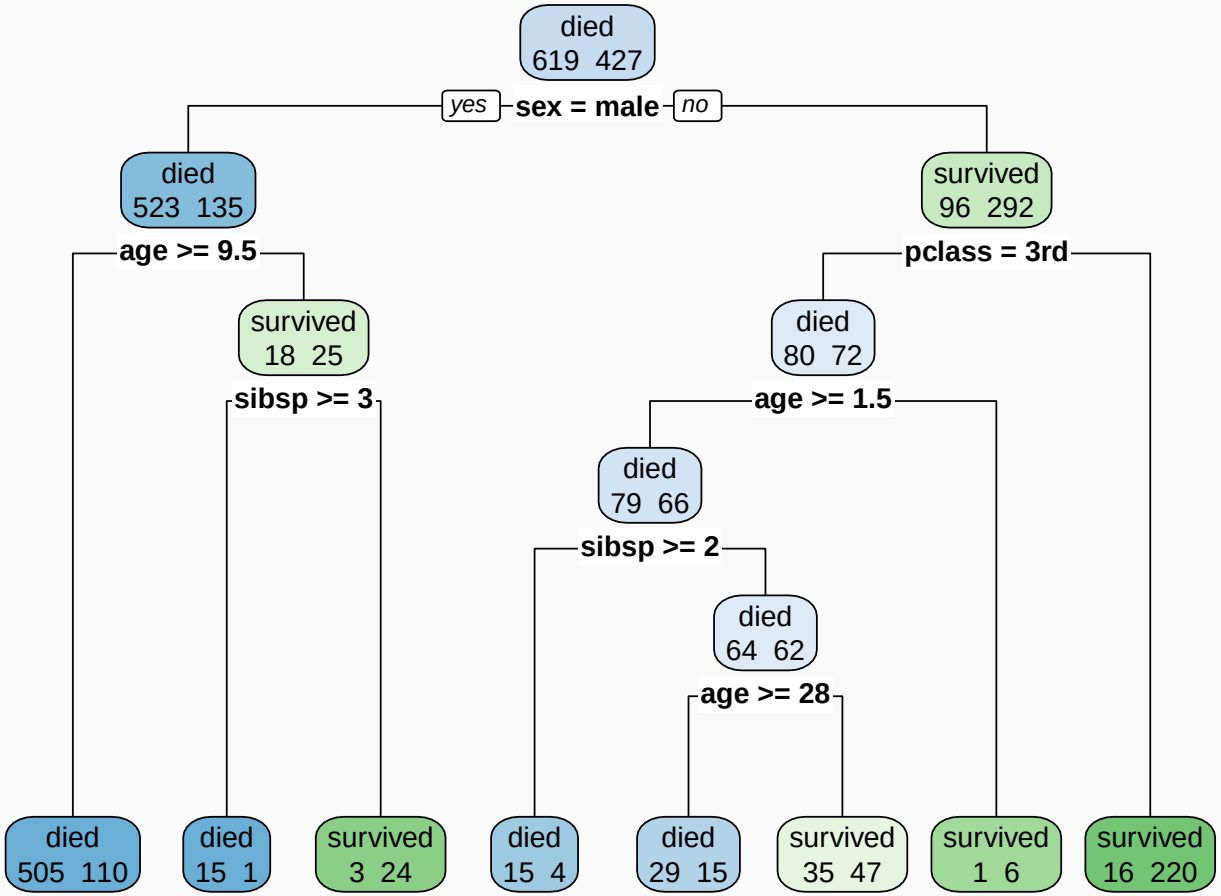
Regularized Regression - Summary of Advantages & Disadvantages

- Advantages:
  - Handles multicollinearity (ridge shrinks correlated groups; elastic net balances selection + grouping).
  - Fast to train, easy to deploy, and coefficients support straightforward communication and simple scoring rules.
- Disadvantages:
  - Assumes mostly linear/additive relations; needs feature engineering (interactions, splines) for nonlinear effects.
  - Selection can be unstable with highly correlated predictors (LASSO may pick one and drop others); ridge does not select at all.

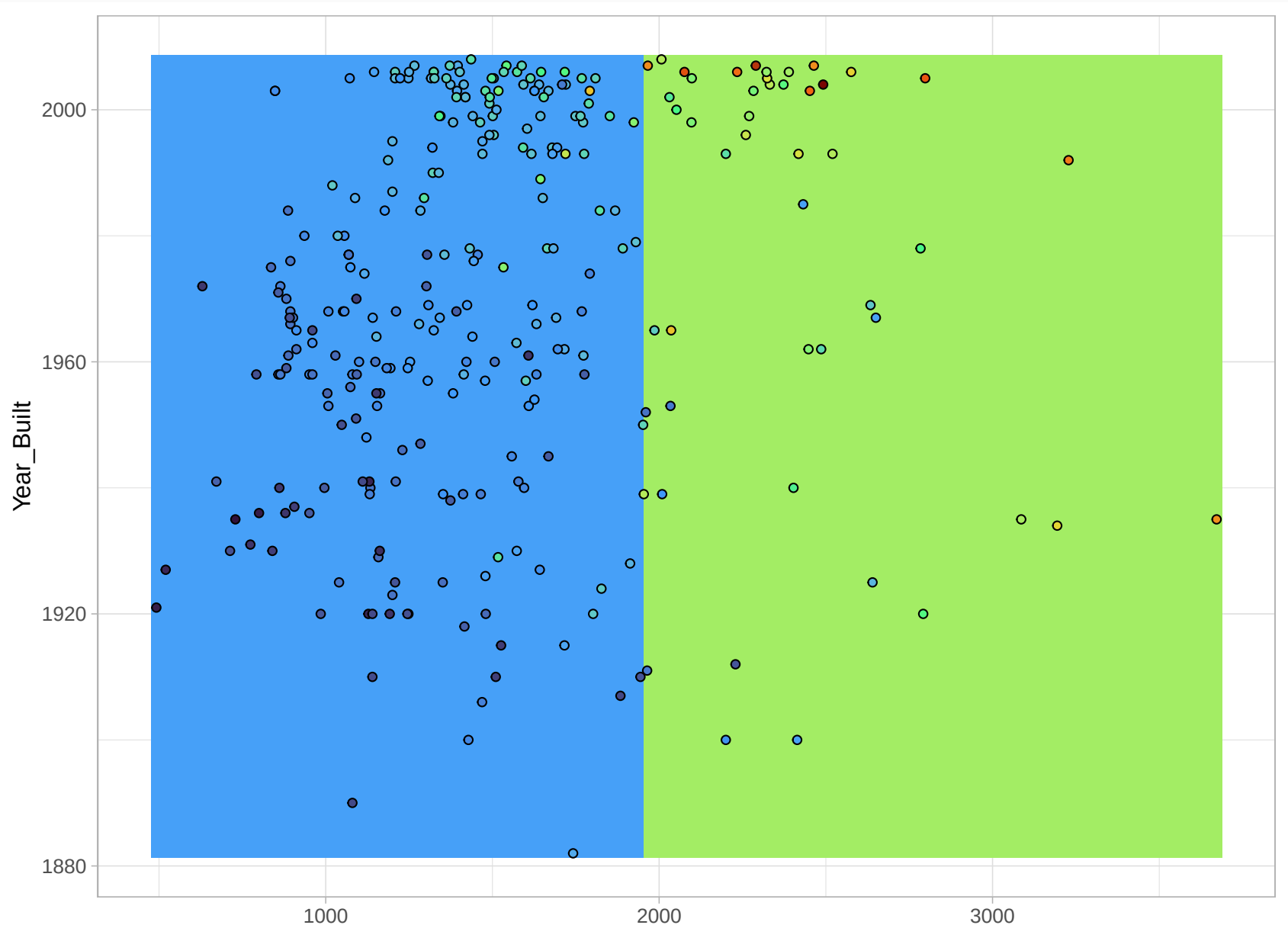
Basic Principles of Decision Trees (Breiman et al. 1984)

- Use predictor variables to iteratively partition the data space into single nodes
- Constant prediction within nodes (mean or mode)
- Optimization criterion determines split-variables and split-points simultaneously
- **Goal:** maximize "purity" within nodes
- Node-purity is defined by an **impurity** function, for example:
  - **MSE** (regression) = variance in target
  - **MMCE** (classification) = proportion of smaller class

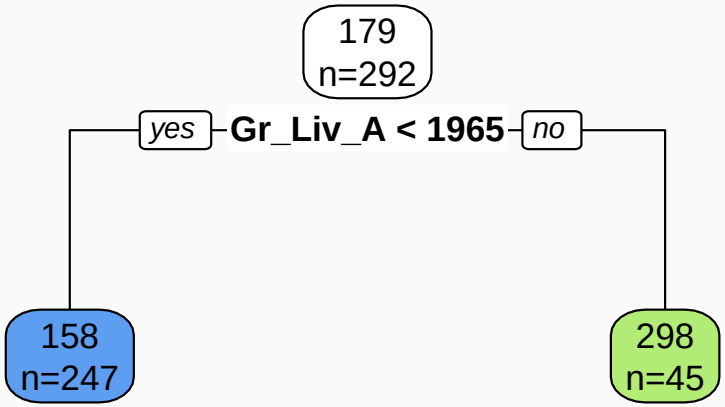
Example: Titanic dataset



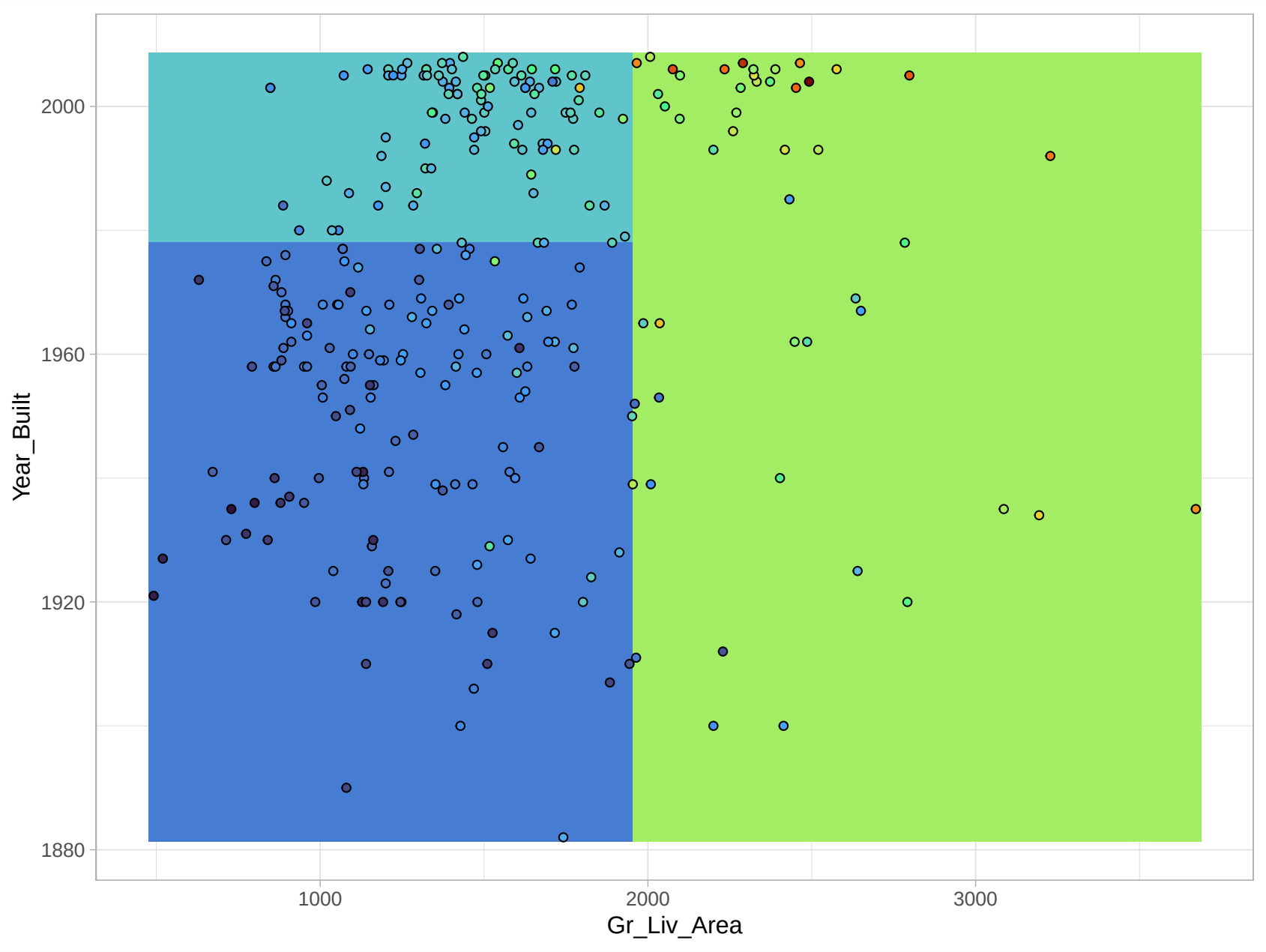
Example: 1 Split



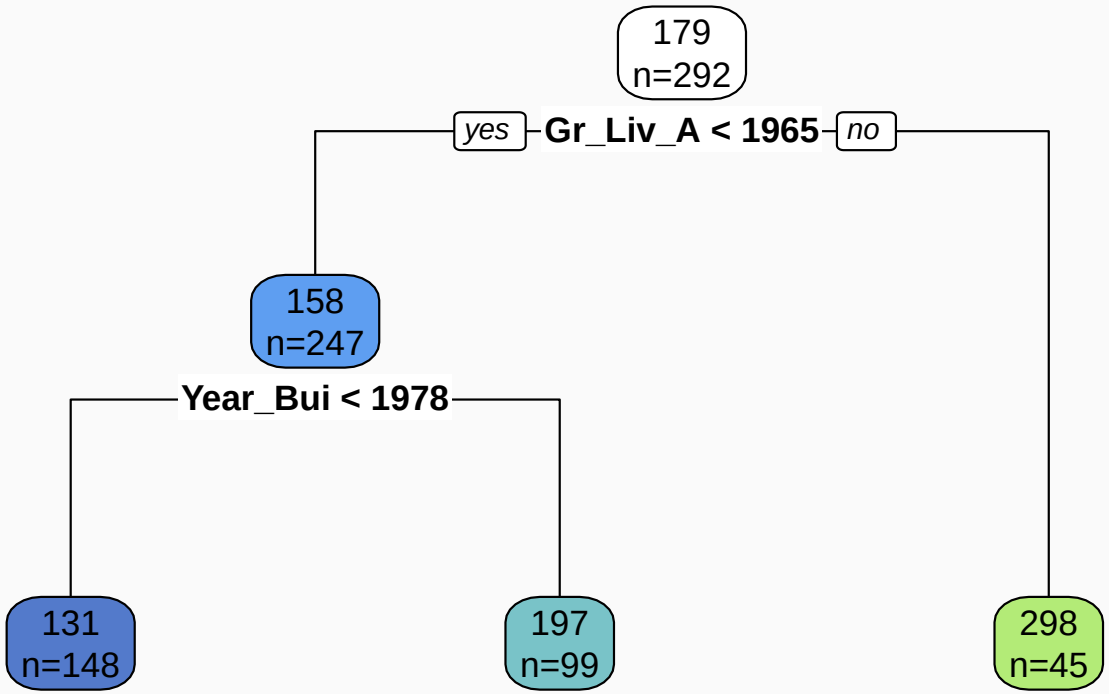
Example: 1 Split



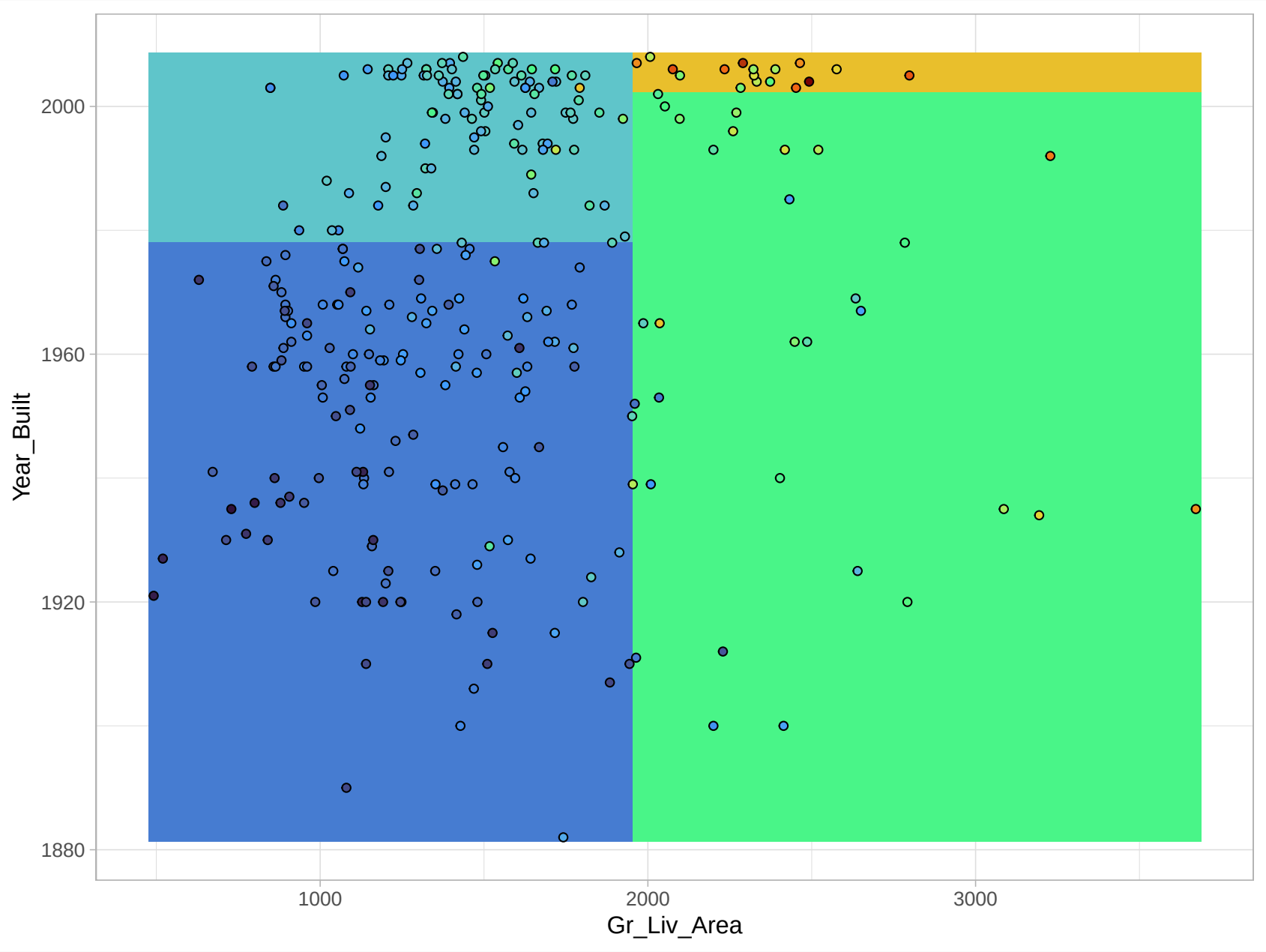
Example: 2 Splits



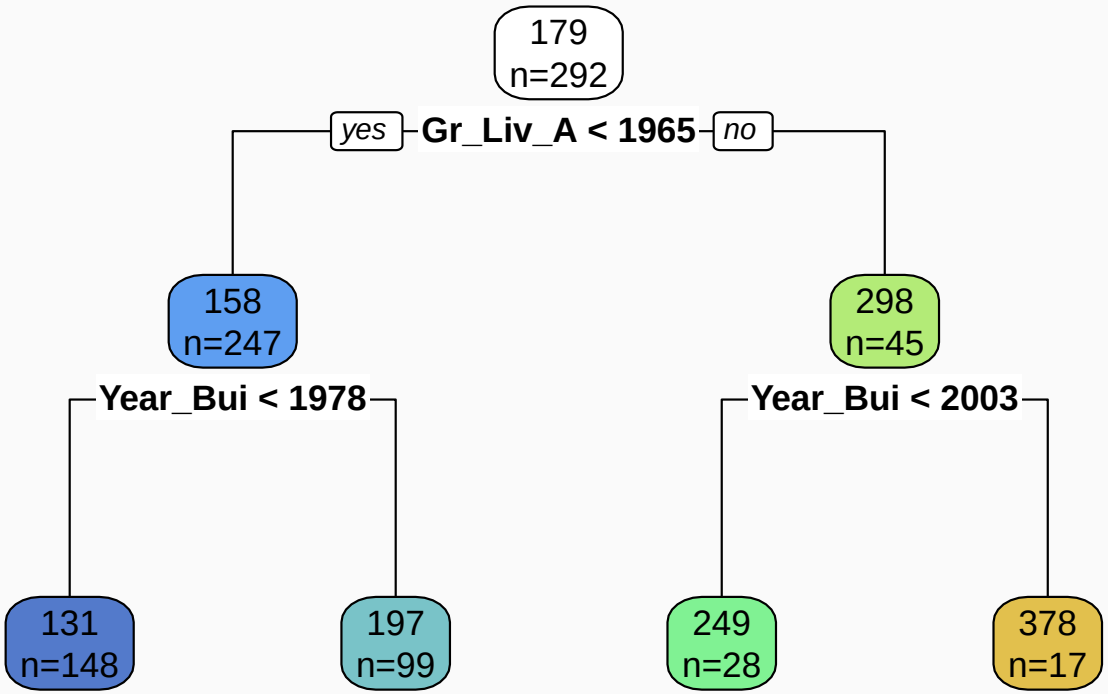
Example: 2 Splits



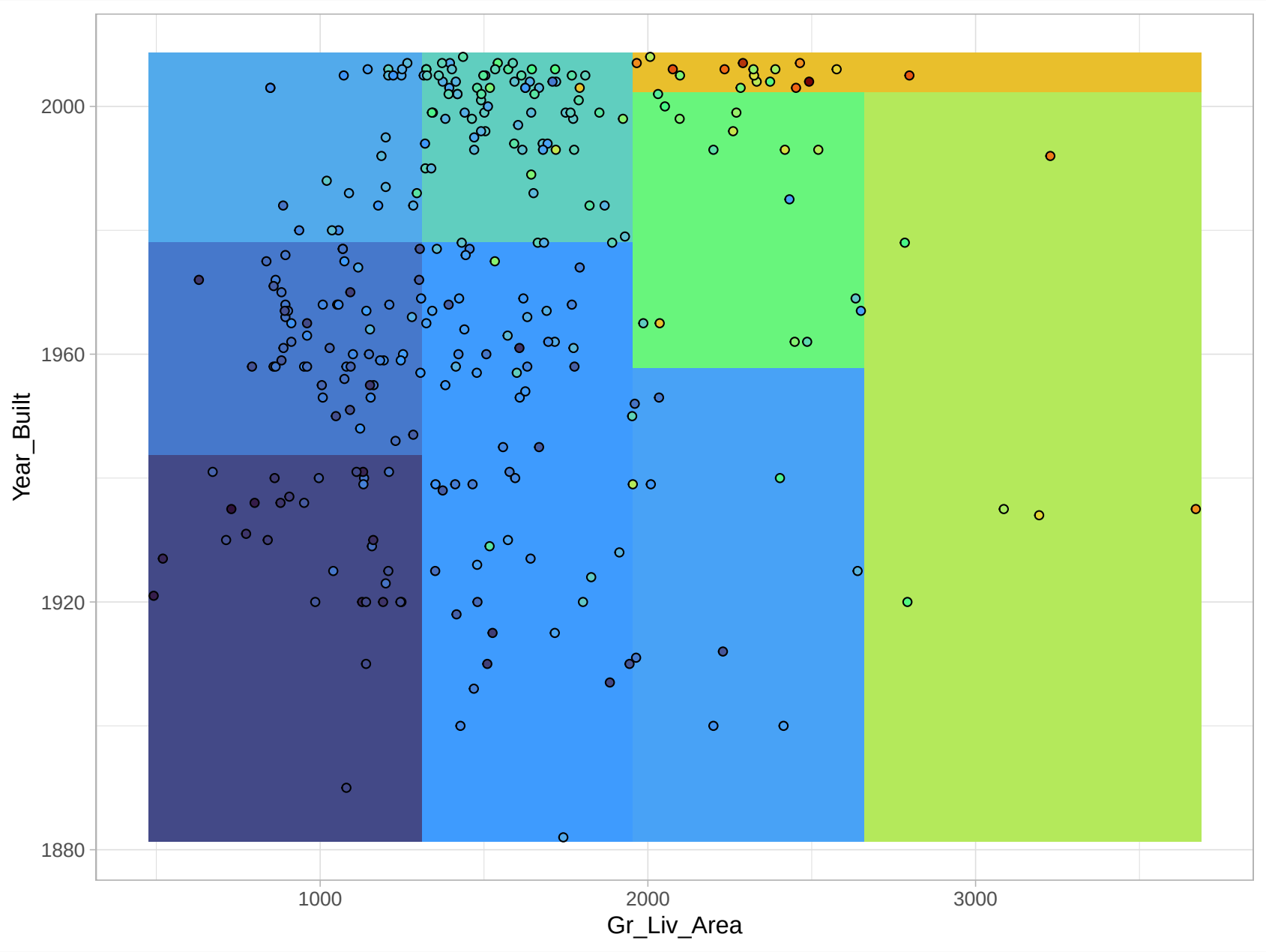
Example: 3 Splits



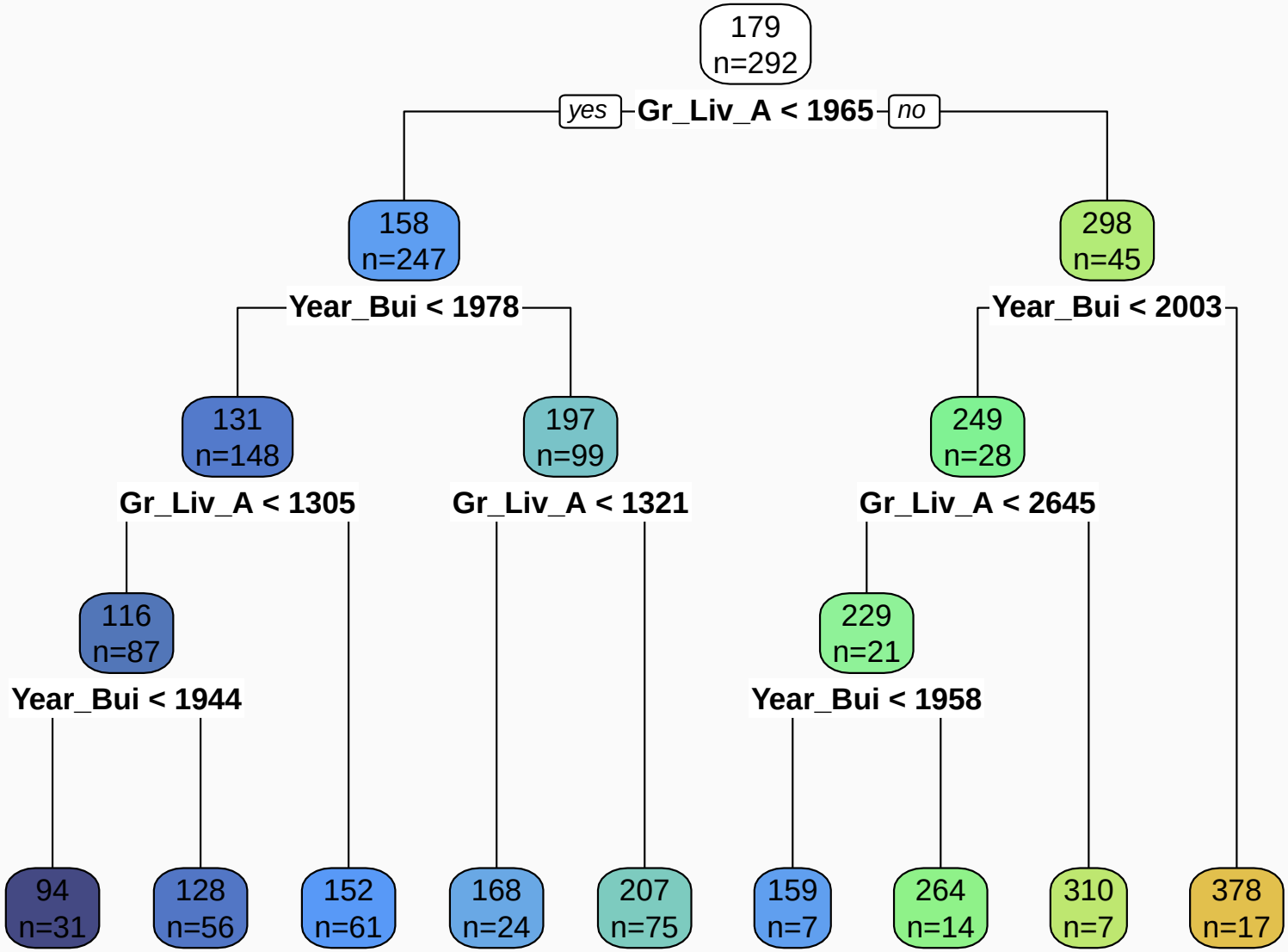
Example: 3 Splits



Example: Default Number of Splits



Example: Default Number of Splits



Tree Bagging - Bootstrap Aggregation

Goal: improve predictive performance by variance reduction

-> aggregate predictions of many trees

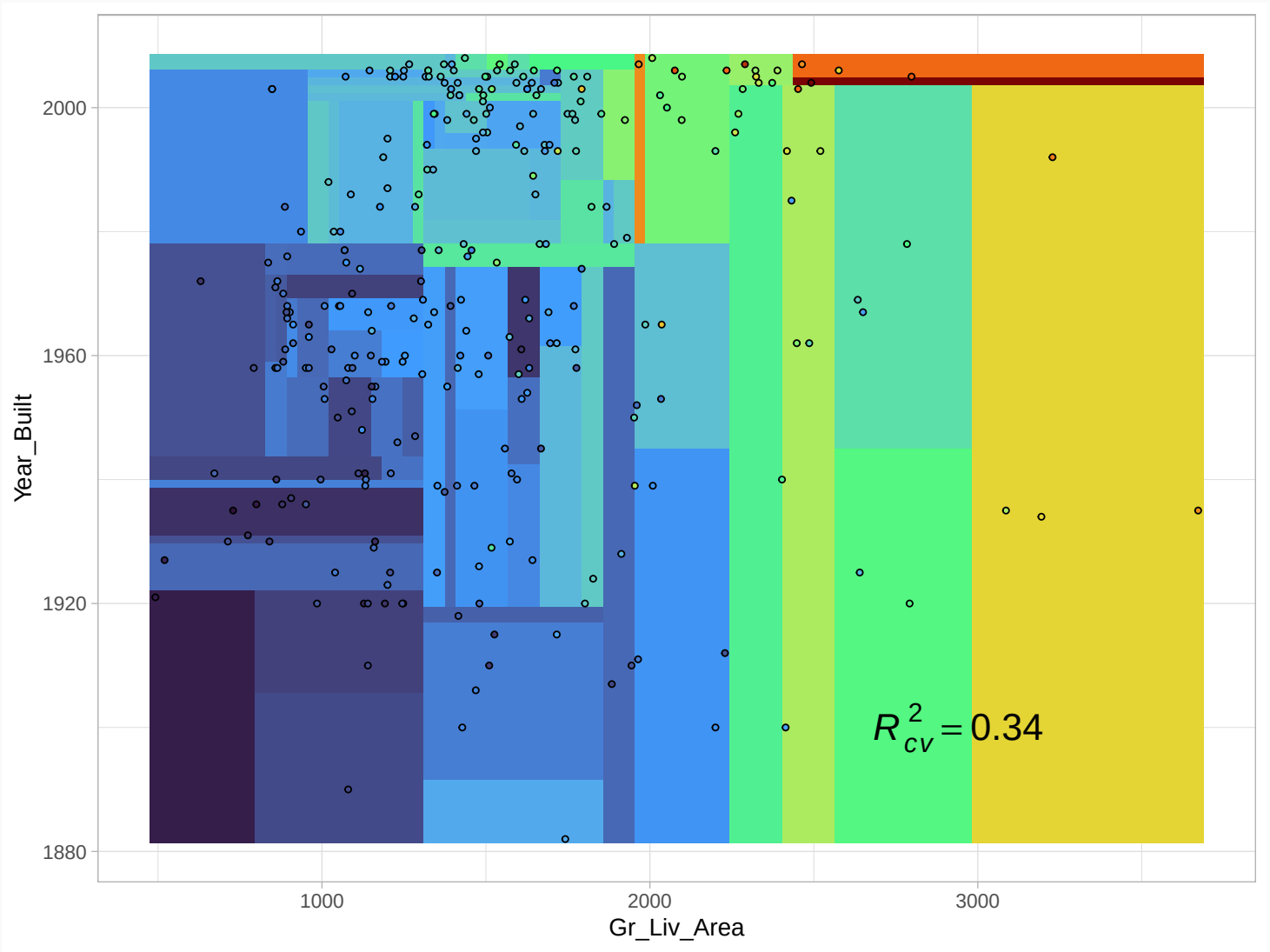
- Algorithm:
  - draw B bootstrap samples (with replacement)
  - fit deep decision tree on each bootstrap sample (liberal stopping criteria)
  - aggregate predictions across all B trees: mean (regression) or majority vote (classification)
- Limitation: highly correlated trees
  - good predictors are often selected early
  - variance reduction works best for uncorrelated trees

Random Forest (Breiman 2001)

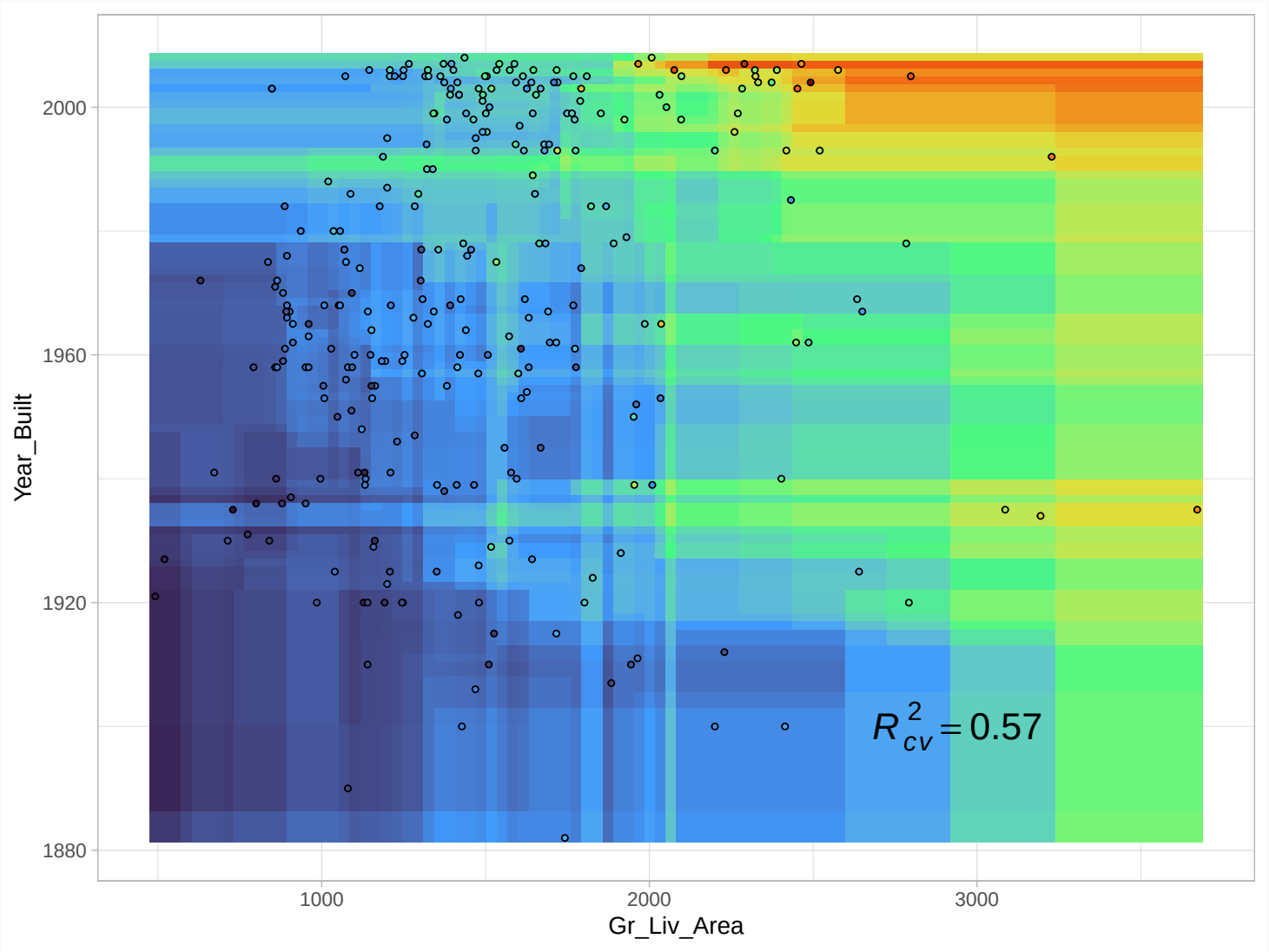
Goal: improve variance reduction by reducing the correlation between trees

- Random subset of predictors at each split
- Hyperparameters:
  - $mtry$ = number of predictors to consider at each split
  - $min.node.size$ = minimal node size to continue splitting
- Rules of thumb:
  - classification ($mtry = \sqrt{p}, min.node.size = 1$)
  - regression ($mtry = \frac{p}{3}, min.node.size = 5$)
- Alternative: determine optimal values by **tuning**

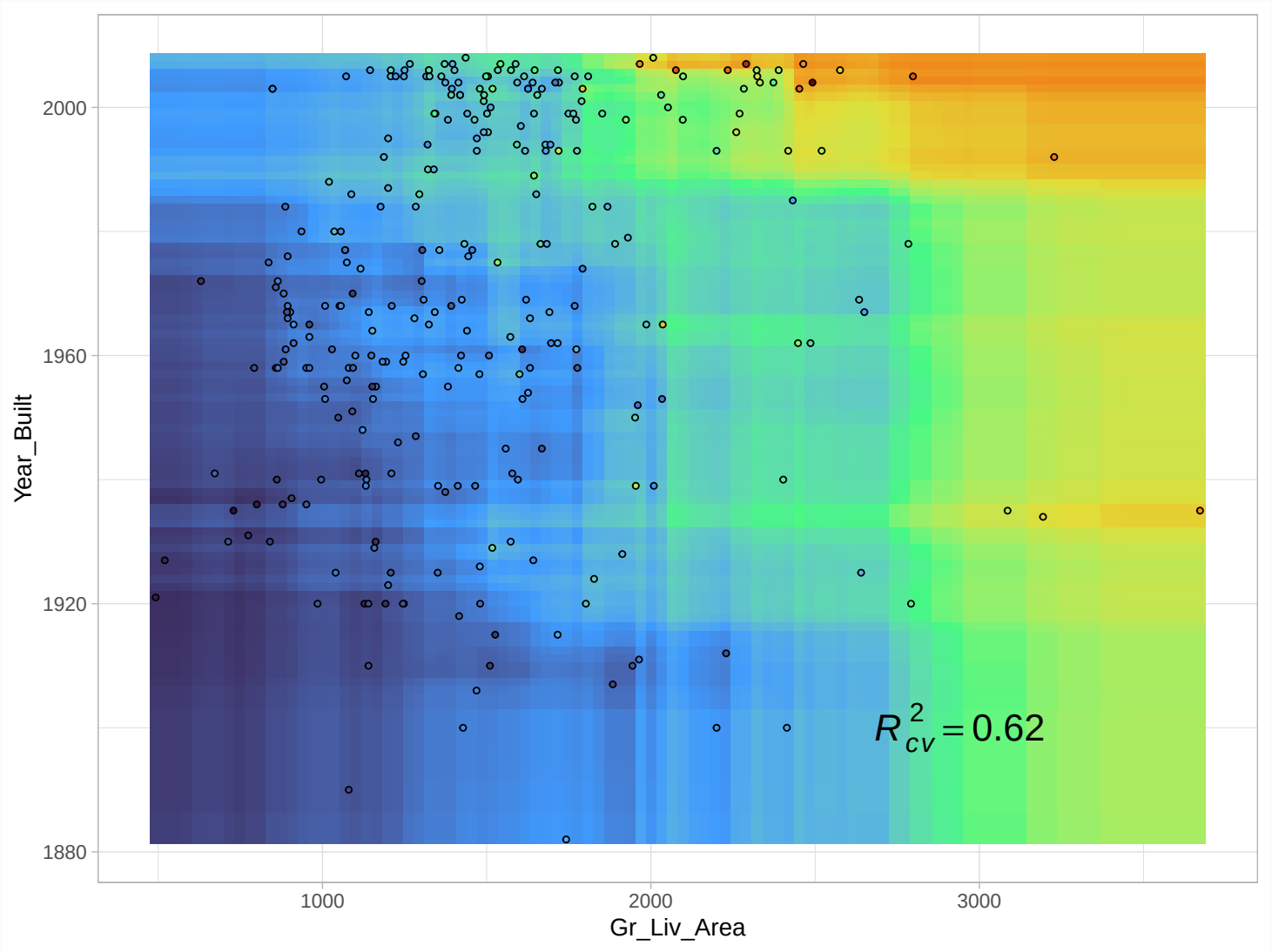
Random Forest (1 tree)



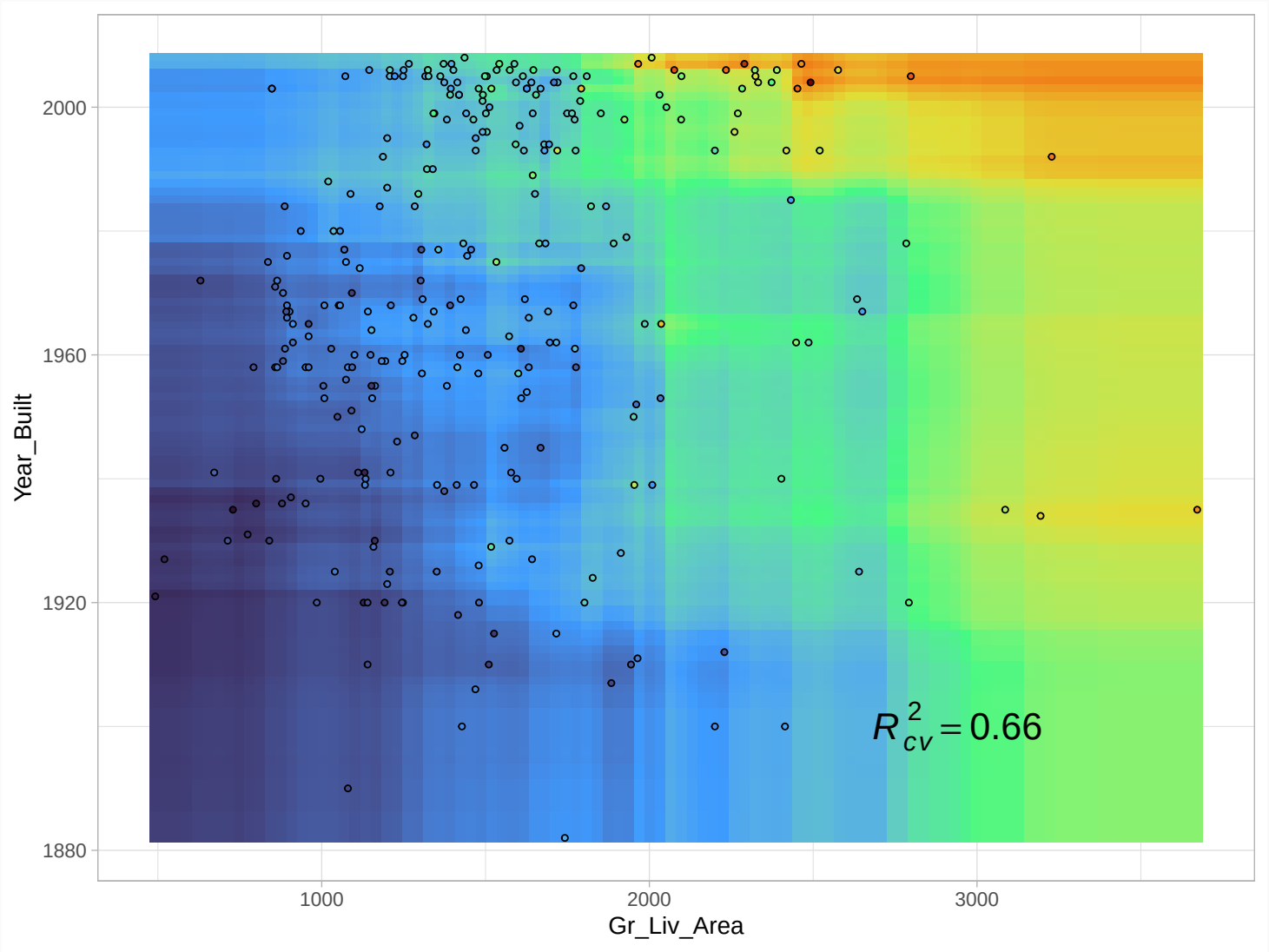
Random Forest (5 trees)



Random Forest (50 trees)



Random Forest (500 trees)



Random Forests - Summary of Advantages & Disadvantages

- Advantages:
  - one of the best "off-the-shelf" algorithms (Fernández-Delgado et al. 2014)
  - inherits all advantages of decision trees
  - better performance than single trees
  - better handles nonlinearity and interactions
  - low bias, low variance
  - no overfitting (you can't fit too many trees)
  - tuning often not necessary
- Disadvantages:
  - still not optimal for truly linear regression tasks
  - harder to interpret (additional tools necessary)

Let's Take a Break!

Picture by Kaboompics.com: https://www.pexels.com/de-de/foto/kaffee-tasse-becher-loffel-6398/

3 HANDS-ON: Random Forest Training

Exercise 3:

1. Repeat Earlier Steps
2. Prepare the Data for Classification
3. Train the Model
4. Inspect the Model
5. Estimate out-of-sample performance

Switch to RStudio and Repeat Earlier Steps

```
# load the package
library(mlr3verse)

# load the data
phonedata <- readRDS(file = "clusterdata.RDS")
# remove participants who did not indicate their gender
phonedata <- phonedata[complete.cases(phonedata$gender),]
# remove variables we do not use in our tutorial
# but this time: include gender (variable 1823)
phonedata <- phonedata[, c(1:1821, 1823, 1837)]

# we will need the regression task later
task_Soci <- as_task_regr(phonedata,
                          id = "Sociability_Regr",
                          target = "E2.Sociableness"
                          )
task_Soci$set_col_roles("gender",
                        remove_from = "feature")
```

Prepare the Data for Classification

We will start with a classification exercise. As our *target* is originally continuous, we categorize into high and low sociability.

```
phonedata$E2.Sociableness_bin <- ifelse(
  phonedata$E2.Sociableness >= median(phonedata$E2.Sociableness),
  "high", "low")
phonedata$E2.Sociableness_bin <-
  as.factor(phonedata$E2.Sociableness_bin)
```

Let's create a *task* object for classification.

```
task_Soci_bin <- as_task_classif(phonedata,
                                 id = "Sociability_Classif",
                                 target = "E2.Sociableness_bin",
                                 positive = "high")
```

Remove *Sociability* and *gender* as feature.

(We do not want to use gender as a feature. But we want to check our models for gender fairness later.)

```
task_Soci_bin$set_col_roles("E2.Sociableness",
                            remove_from = "feature")
task_Soci_bin$set_col_roles("gender",
                           remove_from = "feature")
```

```
task_Soci_bin
```

```
-- <TaskClassif> (620x1822) ------------------------------------------------
* Target: E2.Sociableness_bin
* Target classes: high (positive class, 55%), low (45%)
* Properties: twoclass
* Features (1821):
  * dbl (1821): AR_num_calls_in1, AR_num_calls_in12, AR_num_calls_in24,
  AR_num_calls_in4, AR_num_calls_in8, AR_num_calls_out1, AR_num_calls_out12,
  AR_num_calls_out24, AR_num_calls_out4, AR_num_calls_out8, AR_num_calls_ring1,
  AR_num_calls_ring12, AR_num_calls_ring24, AR_num_calls_ring4,
  AR_num_calls_ring8, IVI_call_in, IVI_call_in_week, IVI_call_in_weekend,
  IVI_call_miss, IVI_call_miss_week, IVI_call_miss_weekend, IVI_call_out,
  IVI_call_out_week, IVI_call_out_weekend, IVI_call_ring, IVI_call_ring_week,
  IVI_call_ring_weekend, IVI_call_week, IVI_call_weekend, IVI_calls,
  IV_Academia, IV_Artistic_Hobby, IV_Beauty, IV_Betting_Risk, IV_Calculator,
  IV_Calendar_Apps, IV_Calling, IV_Camera, IV_Checkup_Monitoring,
  IV_ComicsBooks, IV_Dating_Mating, IV_E_Mail, IV_Eating, IV_Education,
  IV_Emergency_Warning, IV_Entertainment, IV_Financial, IV_Gallery,
  IV_Gaming_Action, IV_Gaming_Adventure, IV_Gaming_Casual, IV_Gaming_Knowledge,
  IV_Gaming_Logic, IV_Gaming_Role_Playing, IV_Gaming_Simulation,
  IV_Gaming_Sports, IV_Gaming_Strategy, IV_Gaming_Tools_Community,
  IV_Health_SelfMonitoring, IV_Image_And_Video_Editing, IV_Internet_Browser,
  IV_Jobs_Additional_Income, IV_Language_Learning, IV_Messaging,
```

Train the Model

Create a *learner* for the random forest classifier.

```
imputer <- po("imputemedian")
rf <- lrn("classif.ranger", num.trees = 500)
rf <- as_learner(imputer %>>% rf)
```

Train the *learner* on the *task*.

```
set.seed(1)
rf$train(task_Soci_bin)
```

Inspect the Trained Model

Extract the random forest model which was computed by the **ranger** package (Wright and Ziegler 2017) as we *trained* the learner.

```
rf$model$classif.ranger$model
```

```
Ranger result

Call:
 ranger::ranger(dependent.variable.name = task$target_names, data = task$data()

Type:                             Classification
Number of trees:                  500
Sample size:                      620
Number of independent variables:  1821
Mtry:                             42
Target node size:                 1
Variable importance mode:         none
Splitrule:                        gini
OOB prediction error:             38.23 %
```

Compute predictions for the same data used in training.

```
pred <- rf$predict(task_Soci_bin)
```

Create a **confusion matrix** for the in-sample predictions.

```
pred$confusion
```

```
        truth
response high low
    high  341   0
    low     0 279
```

All observations have been classified without error. Obviously, this **is not** a realistic estimate for the performance on new data!

-> In-sample performance is useless in machine learning!

-> To evaluate our model, we have to use resampling!

Estimate out-of-sample performance

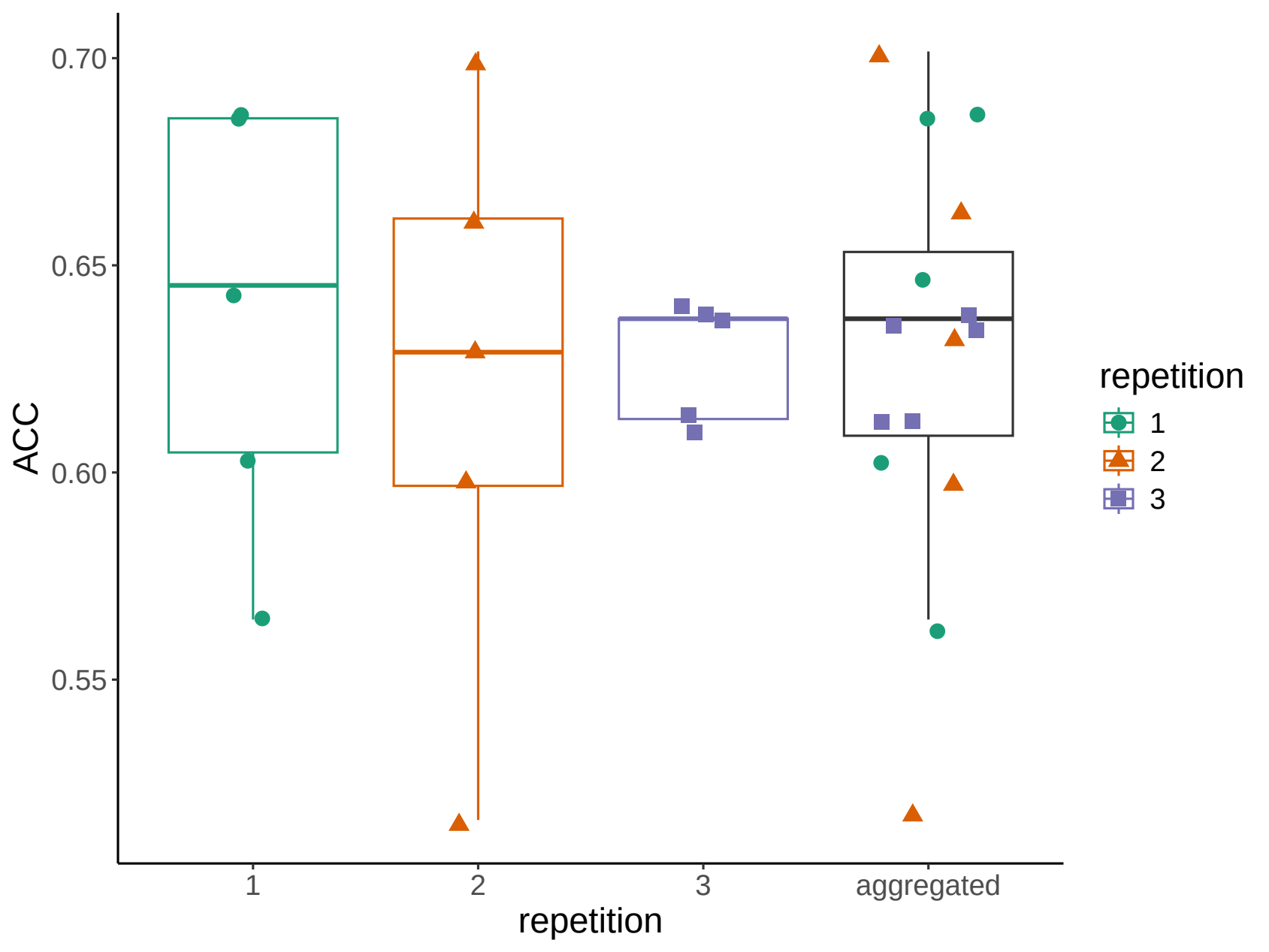
In the following demonstration, we ...

- use 3-times repeated 5-fold cross-validation
- aggregate accuracy across test sets with median
- quantify variability of test set performance with interquartile range

```
rep_cv <- rsmp("repeated_cv", folds = 5, repeats = 3)
res <- resample(task_Soci_bin, learner = rf, resampling = rep_cv)
mes <-  list(msr("classif.acc", id = "ACC (Md)", aggregator = median),
  msr("classif.acc", id = "ACC (IQR)", aggregator = IQR))
res$aggregate(mes)
```

```
 ACC (Md) ACC (IQR)
    0.637     0.044
```

Variability of resampling estimates:



Benchmark Experiments

1 Model Comparisons

We often want to compare different model types because in ML practice it is not always clear from the outset which model type will work best:

- Use benchmark experiments
- including a "featureless" learner = simple baseline model
- ensure fair comparisons (e.g., performance measures, resampling strategies)

Exercise 4: Set up a benchmark experiment

Create *learners* for regression and classification (featureless learner, LASSO, and random forest).

```
imputer <- po("imputemedian")

# regression learners
fl_regr <- lrn("regr.featureless")
lasso_regr <- lrn("regr.cv_glmnet", nfolds = 5)
lasso_regr <- as_learner(imputer %>>% lasso_regr)
rf_regr <- lrn("regr.ranger", num.trees = 100)
rf_regr <- as_learner(imputer %>>% rf_regr)

# classification learners
fl_classif <- lrn("classif.featureless", predict_type = "prob")
lasso_classif <- lrn("classif.cv_glmnet", nfolds = 5,
                     predict_type = "prob")
lasso_classif <- as_learner(imputer %>>% lasso_classif)
rf_classif <- lrn("classif.ranger", num.trees = 100,
                  predict_type = "prob")
rf_classif <- as_learner(imputer %>>% rf_classif)
```

Run the *benchmarks* for regression and classifications.

```
# regression
design_regr <- benchmark_grid(
  tasks = task_Soci,
  learners = list(fl_regr, lasso_regr, rf_regr),
  resamplings = rsmp("cv", folds = 10))
bm_regr <- benchmark(design_regr)

# classification
design_classif <- benchmark_grid(
  tasks = task_Soci_bin,
  learners = list(fl_classif, lasso_classif, rf_classif),
  resamplings = rsmp("cv", folds = 10))
bm_classif <- benchmark(design_classif)

plan("sequential")
```

Choose performance measures for regression and classification.

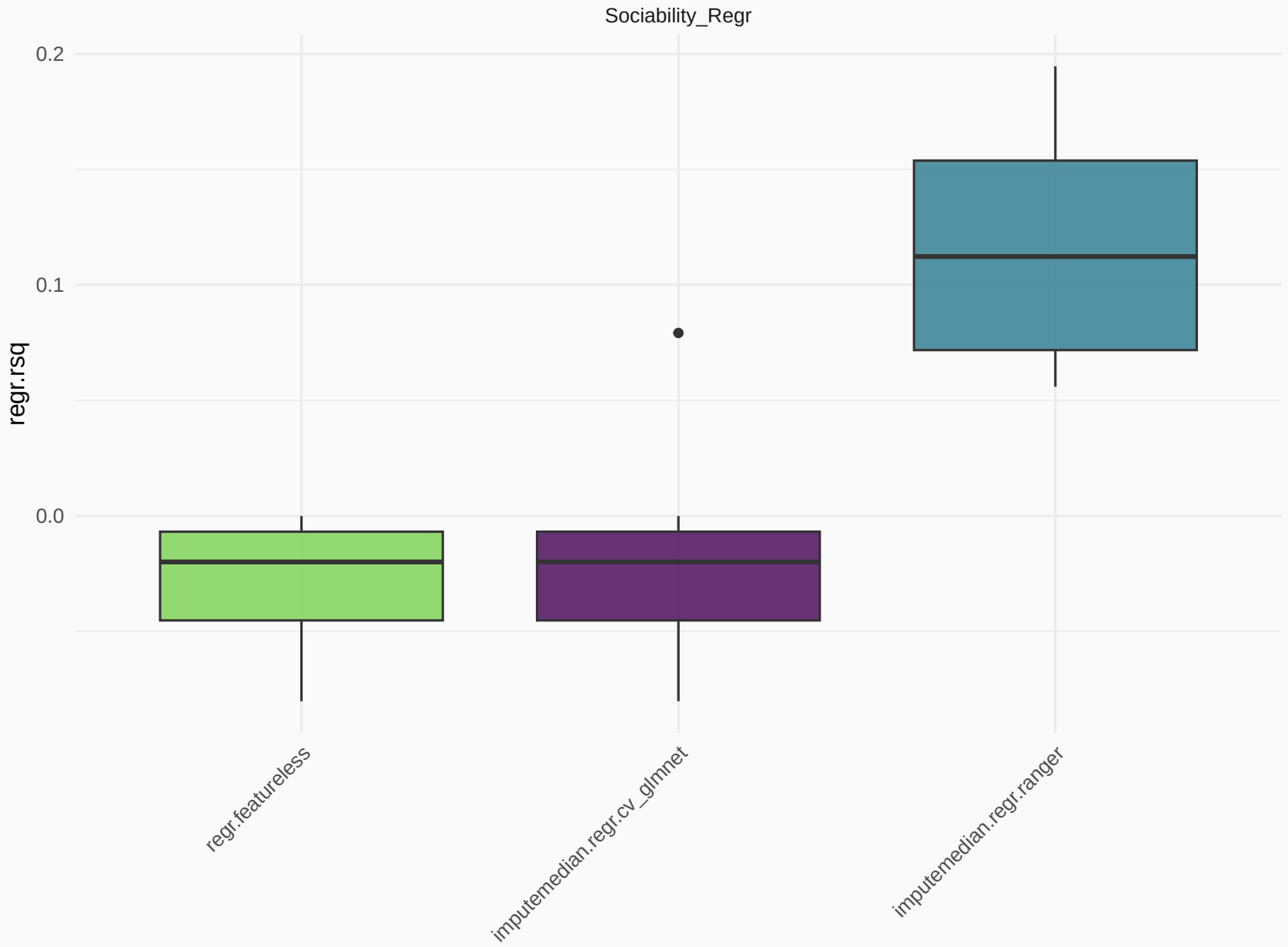
```
mes_regr <- msrs(c("regr.rsq", "regr.mse", "regr.srho"))
mes_classif <- msrs(c("classif.ce", "classif.tpr", "classif.tnr"))
# tpr: sensitivity, tnr: specificity
```

Compute aggregated performance for regression

```
bmr_regr <- bm_regr$aggregate(mes_regr)
bmr_regr[, c(4,7:9)]
```

learner_id	regr.rsq	regr.mse	regr.srho
regr.featureless	-0.03	3.1	NA
imputemedian.regr.cv_glmnet	-0.02	3.1	NA
imputemedian.regr.ranger	0.12	2.7	0.36

```r
plot(bm_regr, measure = msr("regr.rsq"))
```

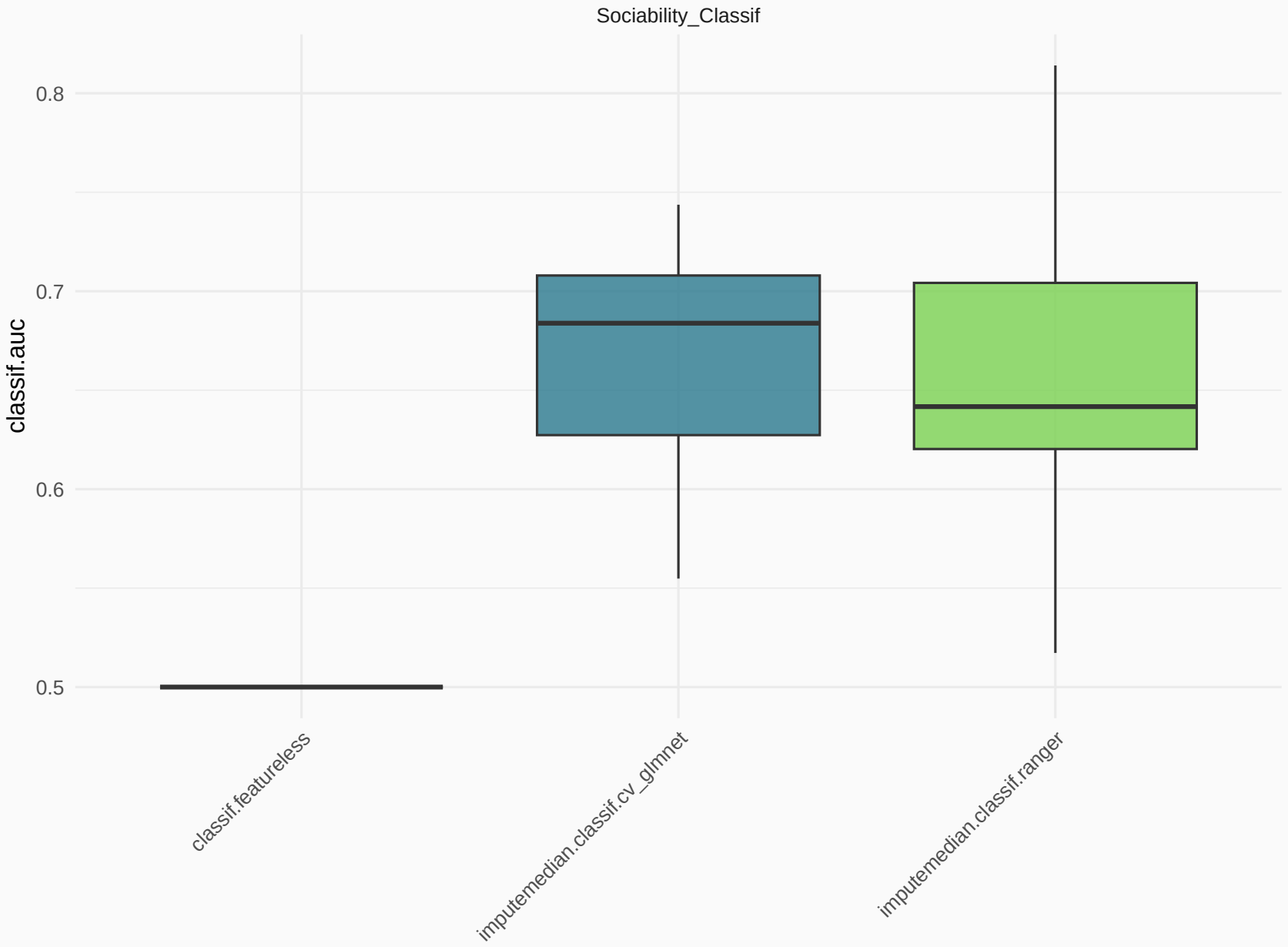


Compute aggregated performance for classification

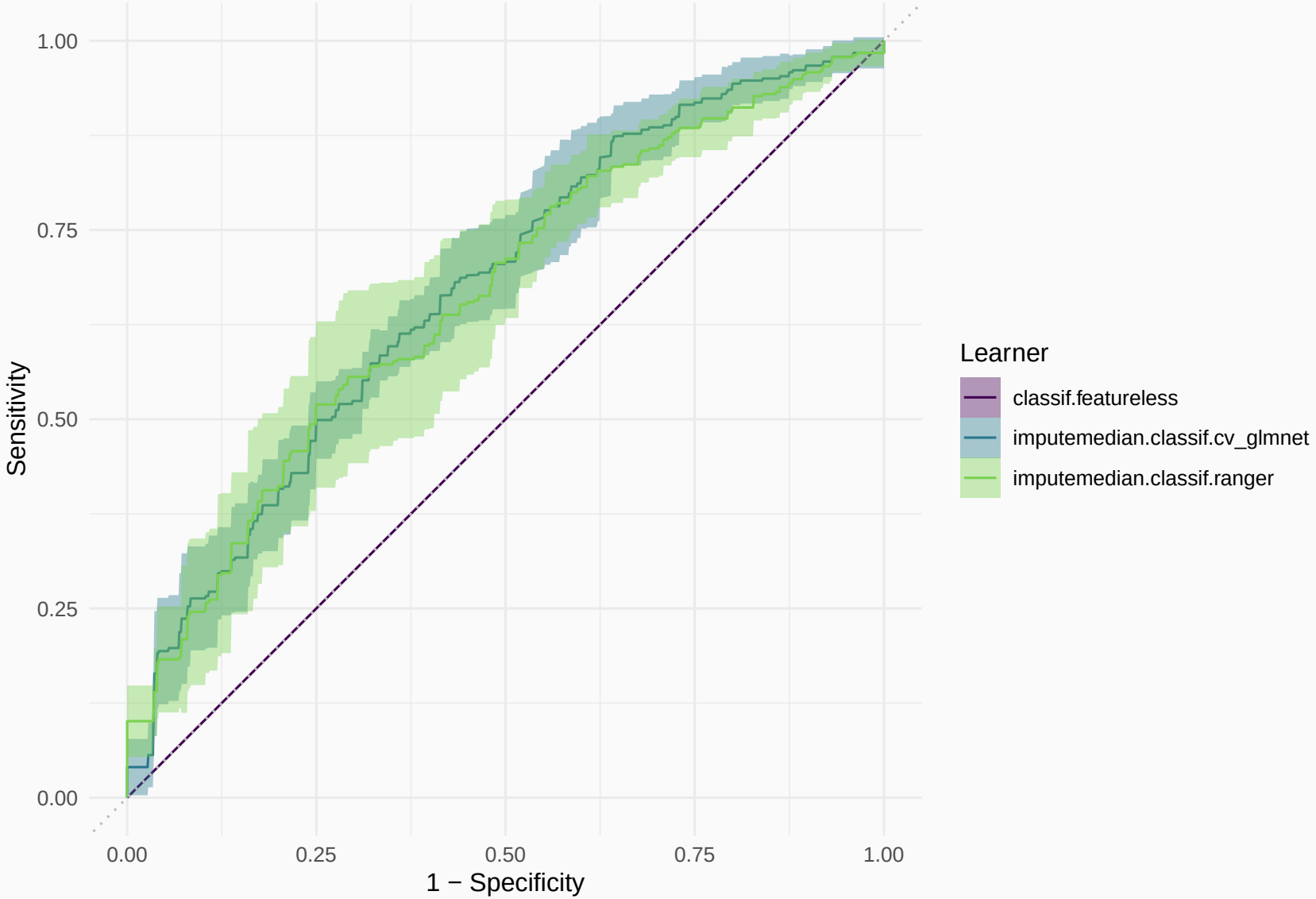
```
bmr_classif <- bm_classif$aggregate(mes_classif)
bmr_classif[, c(4,7:9)]
```

learner_id	classif.ce	classif.tpr	classif.tnr
classif.featureless	0.45	1.00	0.00
imputemedian.classif.cv_glmnet	0.38	0.88	0.32
imputemedian.classif.ranger	0.38	0.75	0.46

```r
plot(bm_classif, measure = msr("classif.auc"))
```



```
plot(bm_classif, type = "roc")
```



Let's Take a Break!

Picture by Kaboompics.com: https://www.pexels.com/de-de/foto/kaffee-tasse-becher-loffel-6398/

Model Interpretation

Goal: Describe model predictions (of any machine learning algorithm) in a humanly understandable way

- *Which* predictors most influence model predictions?
  -> **Variable Importance Measures:**
    - ranking of important predictors (no absolute interpretation!)
    - mostly inspired by measures from random forests
- *How* do individual predictors influence model predictions?
  -> **Effect Plots**:
    - plot possible values of a variable against the (mean) predictions of the model
    - visualize interactions of predictor variables

2 HANDS-ON: Variable Importance & Effect Plots

Interpretable Machine Learning (IML) in the mlr3verse

- We use the **DALEX/DALEXtra** packages (Biecek 2018)
  - **EMA - Book** https://ema.drwhy.ai/
- Alternative: **iml** package (Molnar, Casalicchio, and Bischl 2018)
  - **IML - Book** https://christophm.github.io/interpretable-ml-book/

```
# load packages
library(DALEXtra)
library(ggplot2)

set.seed(123)
rf_regr$train(task_Soci)
```

Construct a new **explainer**.

```
rf_exp <- explain_mlr3(rf_regr,
                       data = phonedata[, c(1:1821,
                              which(colnames(phonedata) == "gender"))],
                       y = phonedata$E2.Sociableness,
                       label = "ranger explainer",
                       colorize = FALSE)
```

```
Preparation of a new explainer is initiated
  -> model label       :  ranger explainer
  -> data              :  620  rows  1822  cols
  -> target variable   :  620  values
  -> predict function  :  yhat.GraphLearner  will be used (  default  )
  -> predicted values  :  No value for predict function target column. (  defau
  -> model_info        :  package mlr3 , ver. 1.1.0 , task regression (  defaul
  -> predicted values  :  numerical, min =  -2.4 , mean =  1.3 , max =  4.4
  -> residual function :  difference between y and yhat (  default  )
  -> residuals         :  numerical, min =  -2.4 , mean =  -0.019 , max =  2.1
  A new explainer has been created!
```

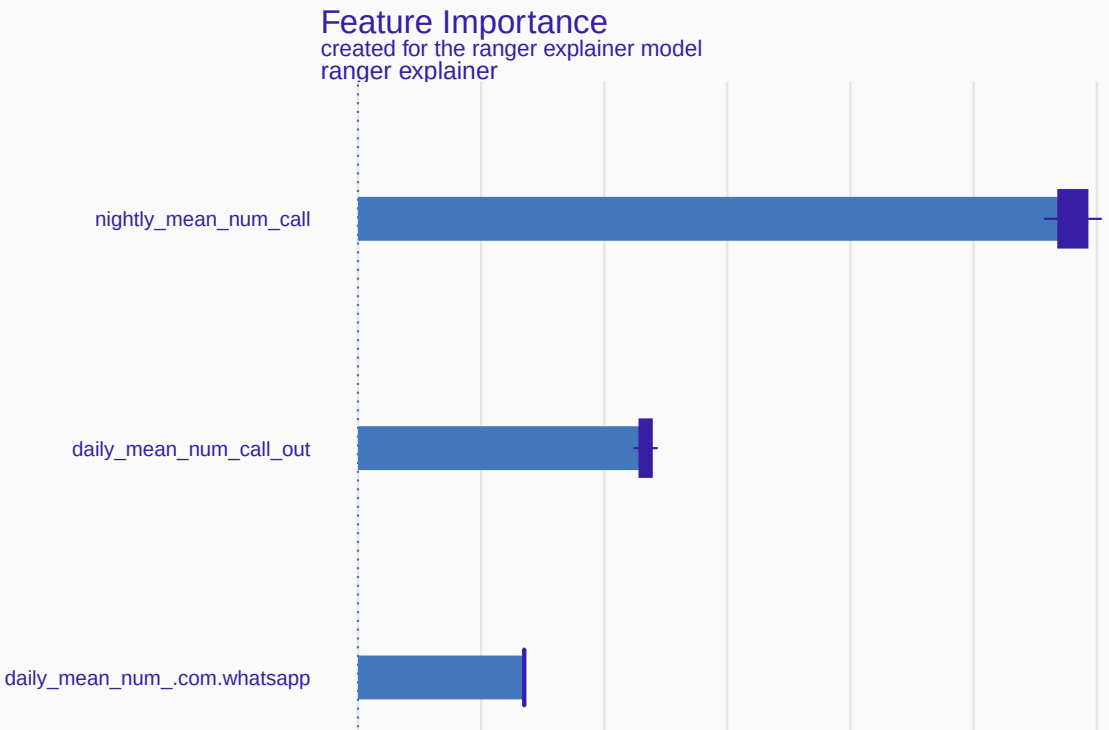
```r
# select a small subset of variables to reduce compute time
# NOTE: in practice you probably want to use ALL variables!

exemplary_features <- c("nightly_mean_num_call",
                        "daily_mean_num_call_out",
                        "daily_mean_num_.com.whatsapp")
```

2 HANDS-ON: Variable Importance & Effect Plots

Permutation Variable Importance

- Measures model performance change after randomly permuting values of a feature

```r
varimp <- model_parts(rf_exp,
                      B = 2,
                      N = nrow(phonedata),
                      #b is the number of permutations
                      #N is the number of obeservations
                      variables = exemplary_features,
                      # do this for our exemplary features
                      type = "difference")
                      # which score is reported

# now plot the results
plot(varimp, show_boxplots = TRUE)
```



Variable Importance

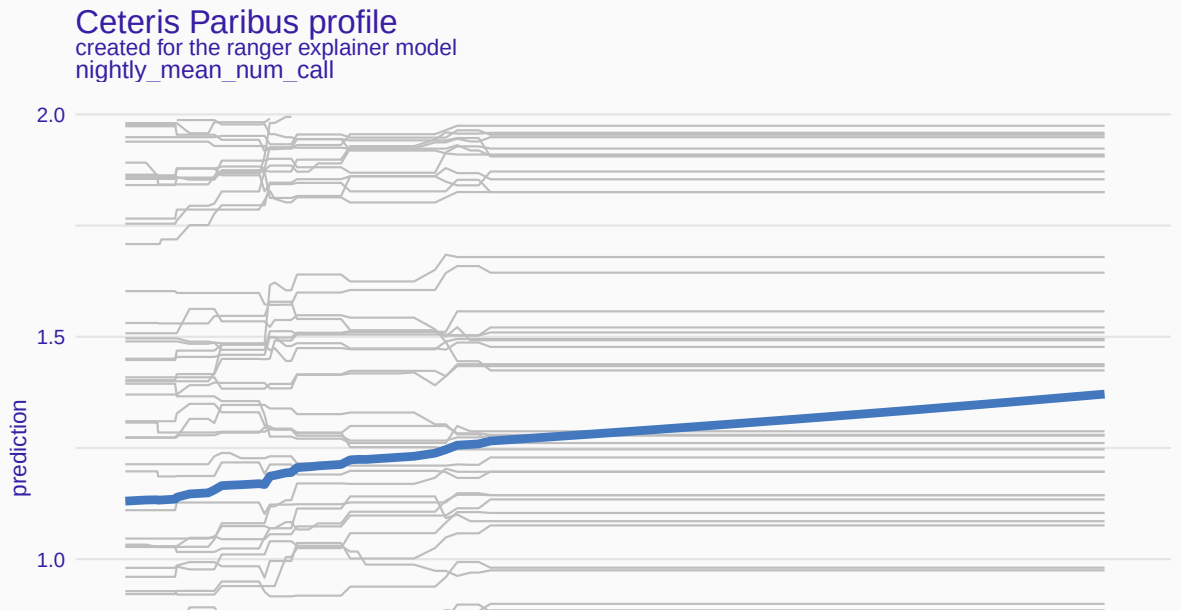
- Random forest specific variable importance measures
  - Random Forests provide specific variable importance measures
  - Some overestimate the importance of variables with many unique values (Strobl et al. 2007) or high correlations with truly important predictors (Strobl et al. 2008)
- Important topics when using variable importance in practice:
  - Different measures lead to different results
  - Interpretations vary
  - Some require alternative RF algorithms (e.g., conditional forests)
  - Computational cost can differ

-> What is most appropriate for your application?

Individual Conditional Expectation (ICE) Profiles

- Shows prediction curve for each individual observation and average overall
- Reveals variation across individuals

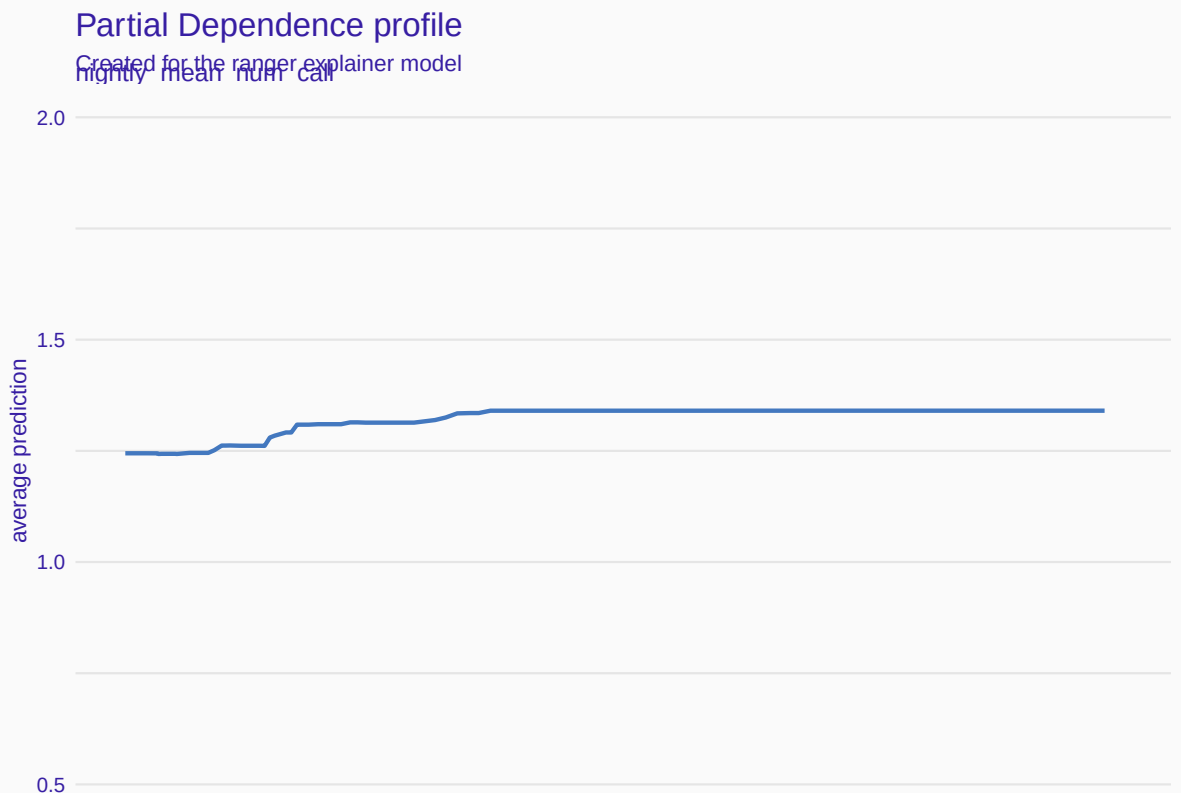
```r
ice <- model_profile(rf_exp,
                     variables = exemplary_features,
                     N = 100,
                     center = FALSE,
                     type = "conditional")
#N -> number of observations used
# center -> should profiles be centered first
#type -> type of profile ("conditional" for ICE)

# create the plot
plot(ice,
     geom = "profiles",
     variables = "nightly_mean_num_call") +
  geom_rug(sides = "b") + xlim(0, 2) + ylim(0.5, 2)
```



Partial Dependence (PD) Plot

- Displays the average effect of a feature on predictions
- Good for global interpretation

```r
pd <- model_profile(rf_exp,
                    variables = exemplary_features,
                     N = 100,
                    center = FALSE,
                    type = "partial") # "partial" for PDP

#  create plot
plot(pd,
     geom = "aggregates",
     variables = "nightly_mean_num_call") +
  geom_rug(sides = "b") + xlim(0, 2) + ylim(0.5, 2)
```



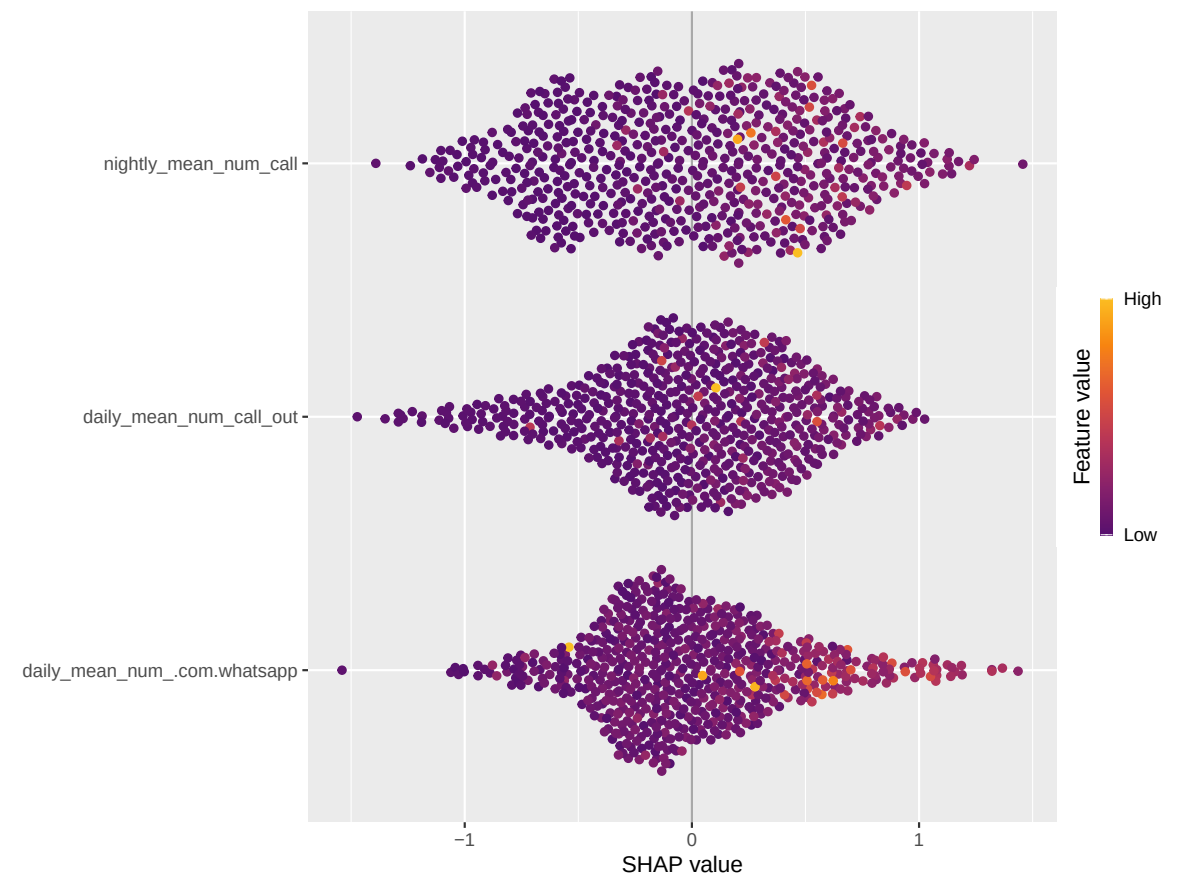
SHapley Additive Explanations (SHAP) Summary Plot

- SHAP is based on cooperative game theory $\rightarrow$ fairly distributes the "prediction difference" among features
- Combines feature importance + feature effects in one visualization
- Each point = one SHAP value for one observation
- Color shows feature value (low $\rightarrow$ high)

```r
# Compute a SHAP summary (global) for a manageable subset of observations

# SHAP can take very long to compute!
# Therefore, we run the analyses on a subset of the data
# to compute SHAP in R, we need to use some other packages

# install.packages(c("treeshap", "shapviz"))  # if needed
library(treeshap)
library(shapviz)
library(ranger)

# 1) Fit a compact RF on our three exemplary features only
# (normally you would take the full model)
rf_model <- ranger(
  formula   = E2.Sociableness ~ .,
  data      = phonedata[, c(exemplary_features, "E2.Sociableness")],
  num.trees = 200,
  mtry      = min(2, length(exemplary_features)),
  importance = "permutation"
)
```

```r
# 2) Unify model with TRAINING FEATURES ONLY (no target)
# this converts tree-based model into a standardized representation.
unified <- ranger.unify(
  rf_model,
  data = phonedata[, exemplary_features, drop = FALSE]
)

# 3) Computes shap values
shap_values <- treeshap(
  unified,
  phonedata[, exemplary_features, drop = FALSE],
  verbose = FALSE # suppress prpgress bar for pdf
)
```

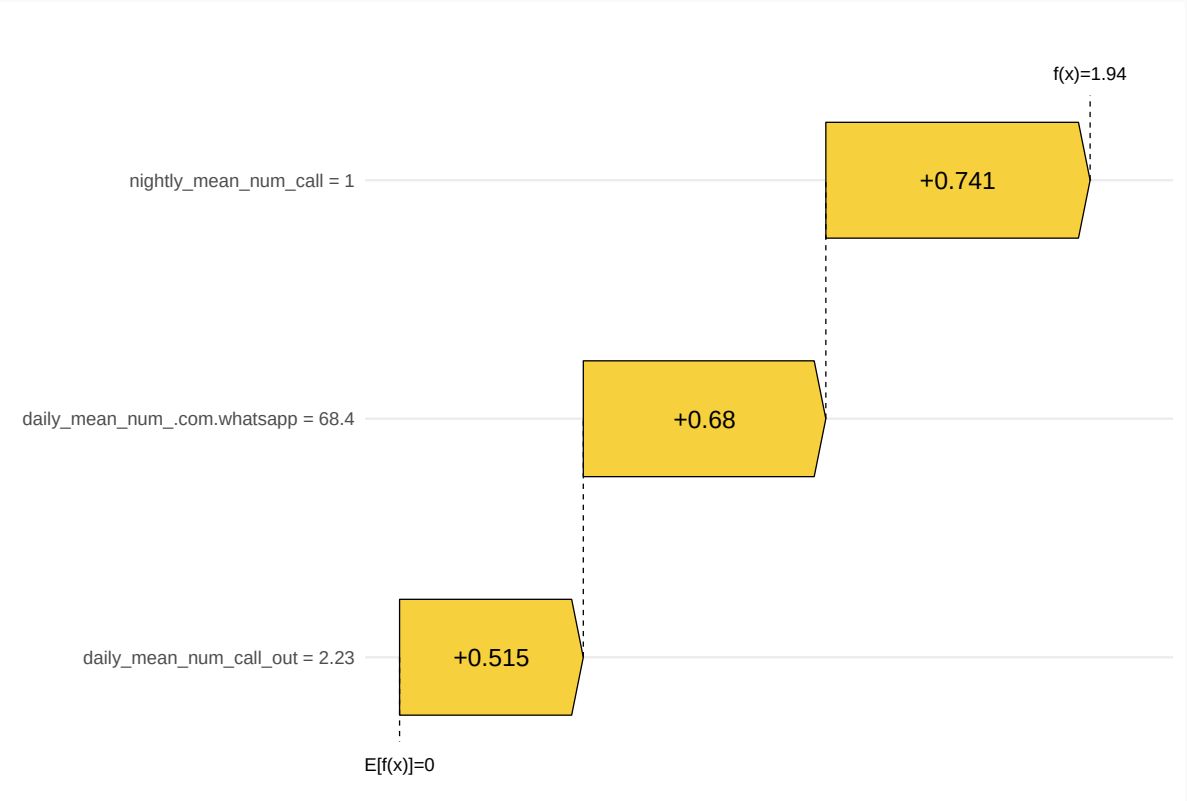
```r
# 4) Visualize: beeswarm with shapviz (global)
sv <- shapviz(shap_values, X = phonedata[, exemplary_features, drop = FALSE])

sv_importance(sv,
              kind = "beeswarm",
              max_display = length(exemplary_features))
```



SHAP Local Explanation (Waterfall Plot)

- Explains one *single* prediction at a time
- Shows how each feature pushes the predicted value up or down
- Helps answer: "Why did the model predict this value for this specific observation?"

```r
## SHAP: Local Explanation for One Participant

# Pick a valid row (avoid NAs)
valid_idx <- which(complete.cases(phonedata[, exemplary_features, drop = FALSE]))

# Local waterfall (prints)
p_wf <- sv_waterfall(sv,
                     row_index = valid_idx,
                     max_display = length(exemplary_features))

print(p_wf) # print the plot
```



CAVEAT: Causal Interpretation & Generalizability

- Do **not** carelessly interpret effect plots **causally**! They show how predictions change, not what causes the outcome
- IML methods only investigate the relationship between feature values and model predictions
- The patterns you see may not represent the true data-generating process (Zhao and Hastie 2021)
- Insights reflect your model + training data and may not hold in other populations or settings

Summary

Take Home Message

When making predictive claims, social scientists should report realistic estimates of predictive performance!

- Correct evaluation allows for safe use of machine learning
- All models (also linear models) should be evaluated adequately:
  - strictly separate training and test sets
  - use cross-validation instead/in addition to in-sample fit
- Repeat all steps from model application during resampling
- Carefully apply IML to learn more about your models inner workings
- Find a **checklist** and **lessons learned** for your own project in Pargent, Schoedel, and Stachl (2023)

Additional Materials for Self Study

- Pargent, Schoedel, and Stachl (2023)
  - https://journals.sagepub.com/doi/full/10.1177/2515245923116255 9
- Personality Computing
  - Henninger et al. (2023)
  - Stachl, Pargent, et al. (2020)
- Books:
  - The mlr3 book; Becker et al. (2021)
  - An Introduction to Statistical Learning; James et al. (2013)
  - Applied Predictive Modeling; Kuhn and Johnson (2013)
- MOOCs:
  - https://introduction-to-machine-learning.netlify.app/
  - https://online.stanford.edu/courses/sohs-ystatslearning-statistical-learning-self-paced

Thank you!

Thrilled to see your studies with machine learning ;-)

Dr. Jan Digutsch
jan.digutsch@unisg.ch
University of St. Gallen

Max Bergmann
maximilian.bergmann@unisg.ch
University of St. Gallen

Dr. Timo Koch
timo.koch@unisg.ch
University of St. Gallen

References i

Arendasy, M., M. Sommer, and M. Feldhammer. 2011. "Manual Big-Five Structure Inventory BFSI." *Schuhfried Gmbh, Mödling*.

Becker, Marc, Martin Binder, Bernd Bischl, Michel Lang, Florian Pfisterer, Nicholas G. Reich, Jakob Richter, Patrick Schratz, and Raphael Sonabend. 2021. "Mlr3 Book." https://mlr3book.mlr-org.com.

Bergmann, Max, Sebastian Müller, Ramona Schoedel, and Clemens Stachl. 2025. "Satisfaction with Life Manifests in Physical Activity Patterns Captured with Smartphones," August. https://doi.org/10.31234/osf.io/g83fn_v1.

Biecek, Przemyslaw. 2018. "DALEX: Explainers for Complex Predictive Models in r." *Journal of Machine Learning Research* 19 (84): 1–5. https://jmlr.org/papers/v19/18-416.html.

References ii

Bischl, Bernd, Olaf Mersmann, Heike Trautmann, and Claus Weihs. 2012. "Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation." *Evolutionary Computation* 20 (2): 249–75.

Breiman, Leo. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.

Breiman, Leo et al. 2001. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statistical Science* 16 (3): 199–231.

Breiman, Leo, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. 1984. *Classification and Regression Trees*. CRC press.

References iii

Digutsch, Jan, Larissa Sust, Ramona Schoedel, Markus Buehner, Timo K. Koch, Maximilian Bergmann, Pietro Alessandro Aluffi, Daniel Racek, and Clemens Stachl. 2025. "Everyday Smartphone Behaviors Predict Life Outcomes." *PsyArXiv*, May. https://doi.org/10.31234/osf.io/x95em_v2.

Eichstaedt, Johannes C, and Aaron C Weidman. 2020. "Tracking Fluctuations in Psychological States Using Social Media Language: A Case Study of Weekly Emotion." *European Journal of Personality* 34 (5): 845–58.

Fernández-Delgado, Manuel, Eva Cernadas, Senén Barro, and Dinani Amorim. 2014. "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?" *Journal of Machine Learning Research* 15: 3133–81. http://jmlr.org/papers/v15/delgado14a.html.

References iv

Gladstone, Joe J, Sandra C Matz, and Alain Lemaire. 2019. "Can Psychological Traits Be Inferred from Spending? Evidence from Transaction Data." *Psychological Science* 30 (7): 1087–96.

Henninger, Mirka, Rudolf Debelak, Yannick Rothacher, and Carolin Strobl. 2023. "Interpretable Machine Learning for Psychological Research: Opportunities and Pitfalls." *Psychological Methods*. https://psycnet.apa.org/doi/10.1037/met0000560.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Springer.

Koch, Timo K., Sanaz Talaifar, Alejandro Hermida Carrillo, Daniel Racek, Jan Digutsch, Pietro Aluffi, Ramona Schoedel, and Clemens Stachl. 2025. "The Digital Authoritarian: Everyday behavioral patterns collected with smartphones predict authoritarianism." *PsyArXiv*, April. https://doi.org/10.31234/osf.io/6mk2a_v1.

References v

Kosinski, Michal, David Stillwell, and Thore Graepel. 2013. "Private Traits and Attributes Are Predictable from Digital Records of Human Behavior." *Proceedings of the National Academy of Sciences* 110 (15): 5802–5.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. Vol. 26. Springer.

Kuhn, Max, and Hadley Wickham. 2018. *Tidymodels: Easily Install and Load the 'Tidymodels' Packages*. https://CRAN.R-project.org/package=tidymodels.

Lang, Michel, Martin Binder, Jakob Richter, Patrick Schratz, Florian Pfisterer, Stefan Coors, Quay Au, Giuseppe Casalicchio, Lars Kotthoff, and Bernd Bischl. 2019. "mlr3: A Modern Object-Oriented Machine Learning Framework in R." *Journal of Open Source Software*, December. https://doi.org/10.21105/joss.01903.

References vi

Matz, Sandra C, Christina S Bukow, Heinrich Peters, Christine Deacons, Alice Dinu, and Clemens Stachl. 2023. "Using Machine Learning to Predict Student Retention from Socio-Demographic Characteristics and App-Based Engagement Metrics." *Scientific Reports* 13 (1): 5705.

Molnar, Christoph, Guiseppe Casalicchio, and Bernd Bischl. 2018. "Iml: An r Package for Interpretable Machine Learning." *Journal of Open Source Software* 3: 786. https://doi.org/10.21105/joss.00786.

Pargent, Florian, and Johannes Albert-Von Der Gönna. 2018. "Predictive Modeling with Psychological Panel Data." *Zeitschrift Fur Psychologie / Journal of Psychology* 226 (4): 246–58. https://doi.org/10.1027/2151-2604/a000343.

References vii

Pargent, Florian, Ramona Schoedel, and Clemens Stachl. 2023. "Best Practices in Supervised Machine Learning: A Tutorial for Psychologists." *Advances in Methods and Practices in Psychological Science* 6 (3): 2515245923116255 9. https://doi.org/10.1177/2515245923116255 9.

Rüegger, Dominik, Mirjam Stieger, Marcia Nißen, Mathias Allemand, Elgar Fleisch, and Tobias Kowatsch. 2020. "How Are Personality States Associated with Smartphone Data?" *European Journal of Personality* 34 (5): 687–713.

Schoedel, Ramona, Fiona Kunz, Maximilian Bergmann, Florian Bemmann, Markus Bühner, and Larissa Sust. 2023. "Snapshots of Daily Life: Situations Investigated Through the Lens of Smartphone Sensing." *Journal of Personality and Social Psychology* 125 (6): 1442–71. https://doi.org/10.1037/pspp0000469.

References viii

Shmueli, Galit. 2010. "To Explain or to Predict?" *Statistical Science* 25 (3): 289–310. https://doi.org/10.1214/10-STS330.

Stachl, Clemens, Quay Au, Ramona Schoedel, Samuel D Gosling, Gabriella M Harari, Daniel Buschek, Sarah Theres Völkel, et al. 2020. "Predicting personality from patterns of behavior collected with smartphones." *Proceedings of the National Academy of Sciences of the United States of America* 117 (30): 17680–87. https://doi.org/10.1073/pnas.1920484117.

Stachl, Clemens, Sven Hilbert, Jiew-Quay Au, Daniel Buschek, Alexander De Luca, Bernd Bischl, Heinrich Hussmann, and Markus Bühner. 2017. "Personality Traits Predict Smartphone Usage." *European Journal of Personality* 31 (6): 701–22. https://doi.org/10.1002/per.2113.

Stachl, Clemens, Florian Pargent, Sven Hilbert, Gabriella M. Harari, Ramona Schoedel, Sumer Vaid, Samuel D. Gosling, and Markus Bühner. 2020. "Personality Research and Assessment in the Era of Machine Learning." *European Journal of Personality* 34 (5): 613–31. https://doi.org/10.1002/per.2257.

Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics* 9 (1): 307. https://doi.org/10.1186/1471-2105-9-307.

Strobl, Carolin, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC Bioinformatics* 8 (1): 25. https://doi.org/10.1186/1471-2105-8-25.

References x

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1): 267–88. http://www.jstor.org/stable/2346178.

Wright, Marvin N, and Andreas Ziegler. 2017. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.

Yarkoni, Tal, and Jacob Westfall. 2017. "Choosing Prediction over Explanation in Psychology: Lessons from Machine Learning." *Perspectives on Psychological Science* 12 (6): 1100–1122. https://doi.org/10.1177/1745691617693393.

Youyou, Wu, Michal Kosinski, and David Stillwell. 2015. "Computer-Based Personality Judgments Are More Accurate Than Those Made by Humans." *Proceedings of the National Academy of Sciences* 112 (4): 1036–40.

Zhao, Qingyuan, and Trevor Hastie. 2021. "Causal Interpretations of Black-Box Models." *Journal of Business & Economic Statistics* 39 (1): 272–81. https://doi.org/10.1080/07350015.2019.1624293.