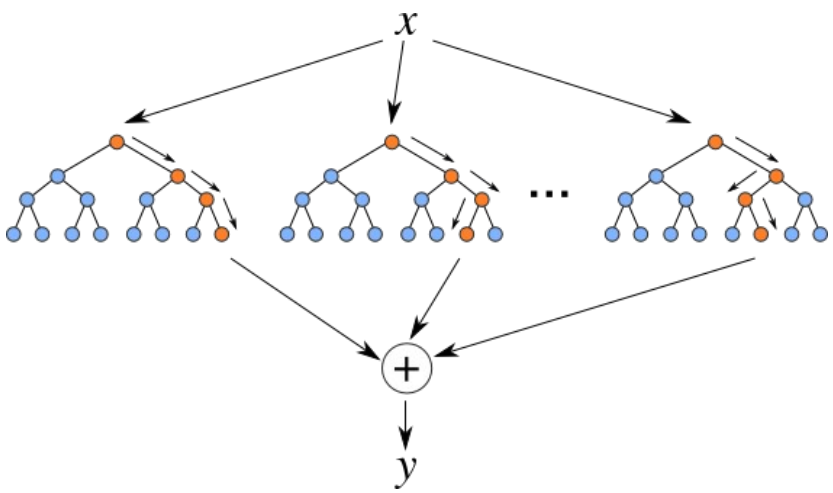
Swiss Paraplegic Research

Workshop:
Machine Learning and Prediction Modelling

Ensemble Methods

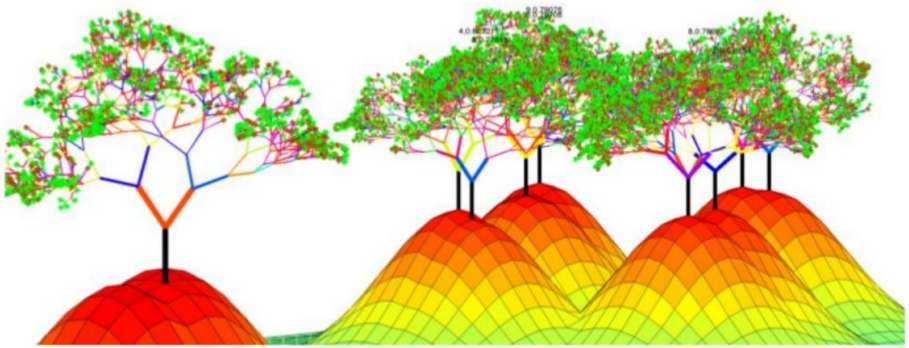


Yannick Rothacher

SPF, HS2025

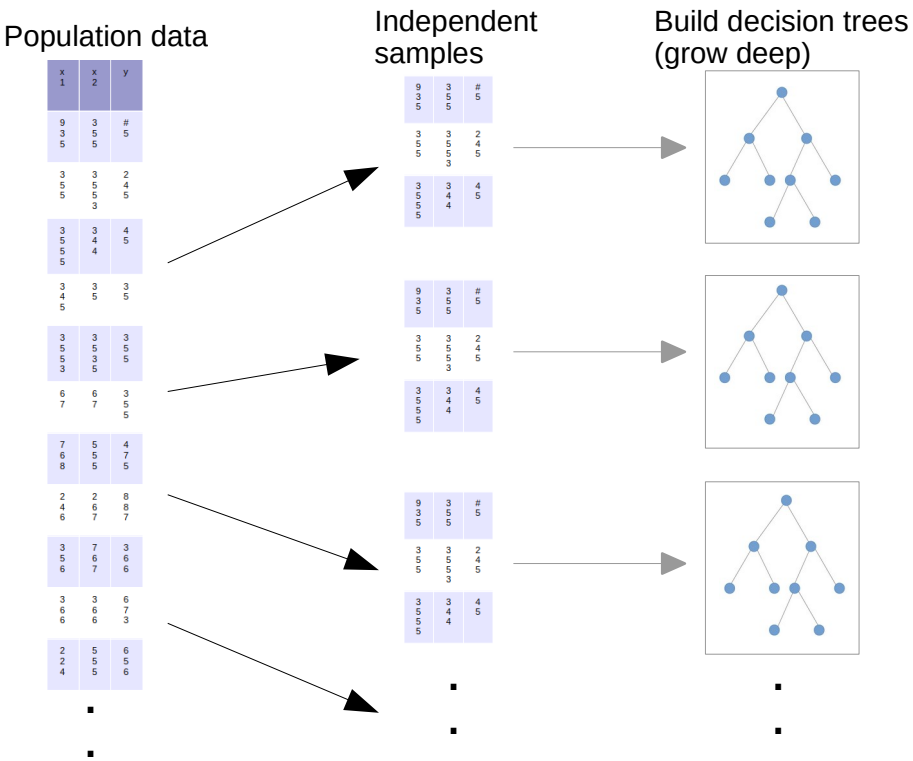
Ensemble methods

- Why not combine multiple models to improve prediction?
- Given some training data and a test-observation:
  - You could apply multiple, different classifiers to the data (logistic regression, KNN, decision tree, neural network, …) and incorporate each classifier's prediction into your final prediction of the test observation (e.g. majority vote)
- **Ensemble methods** are based on a similar intuition: Combine multiple (simple) models of the same type to improve performance
- With regard to decision trees there are two ensemble methods, which are mostly applied:
  - Bagging
  - Random Forest



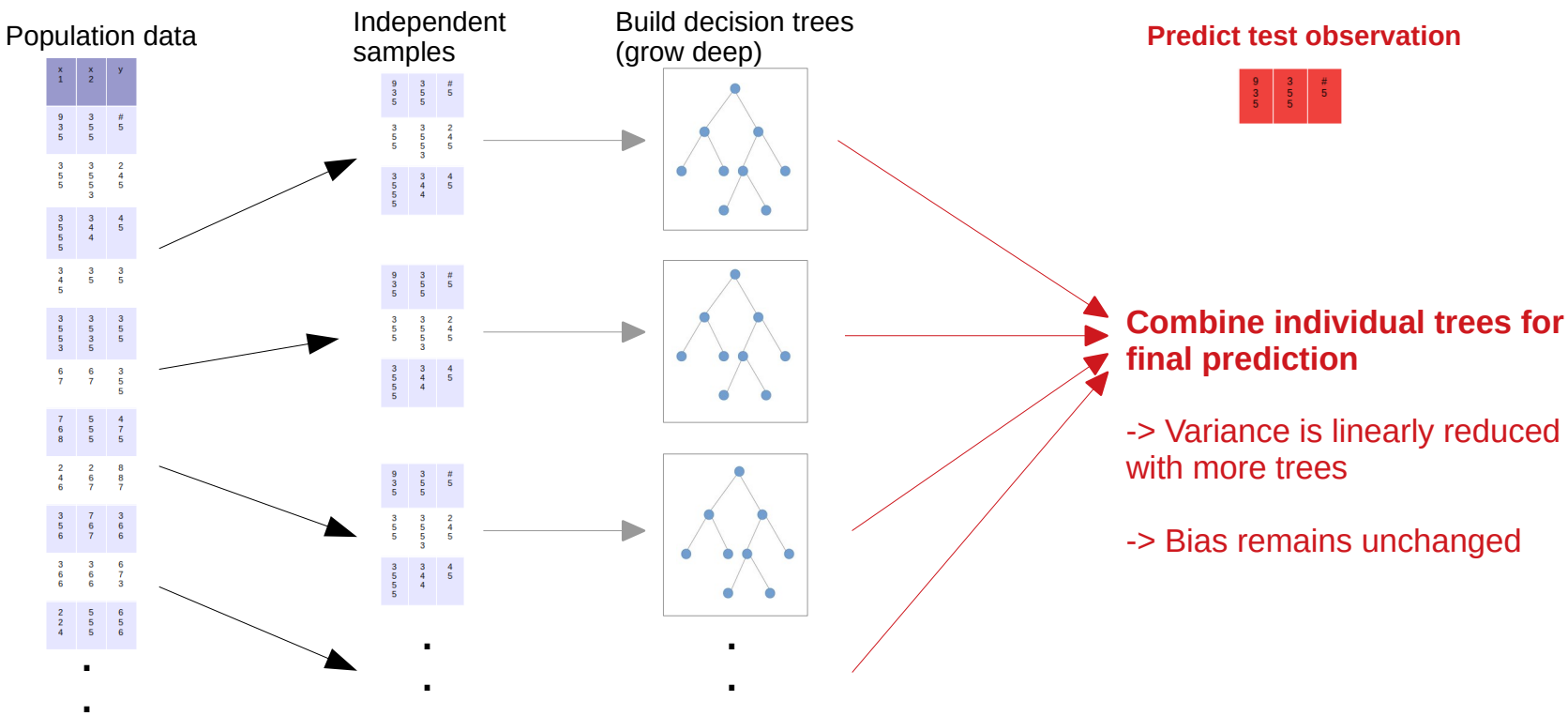
Ensemble methods: Bagging

- **Bagging** (bootstrap aggregating) is an ensemble method typically applied to decision trees (but can be applied to all classifiers)
- Main idea: Fully grown decision trees have **low bias** but **high variance**...
  - ...it would be nice to have multiple independent samples from the same population to **reduce variance** by taking the mean of prediction



Ensemble methods: Bagging

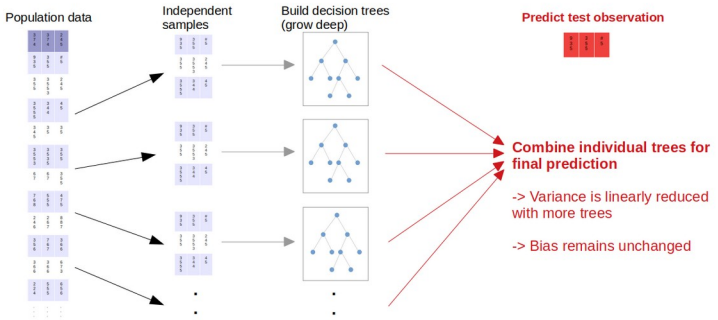
- **Bagging** (bootstrap aggregating) is an ensemble method typically applied to decision trees (but can be applied to all classifiers)
- Main idea: Fully grown decision trees have **low bias** but **high variance**...
  - ...it would be nice to have multiple independent samples from the same population to **reduce variance** by taking the mean of prediction

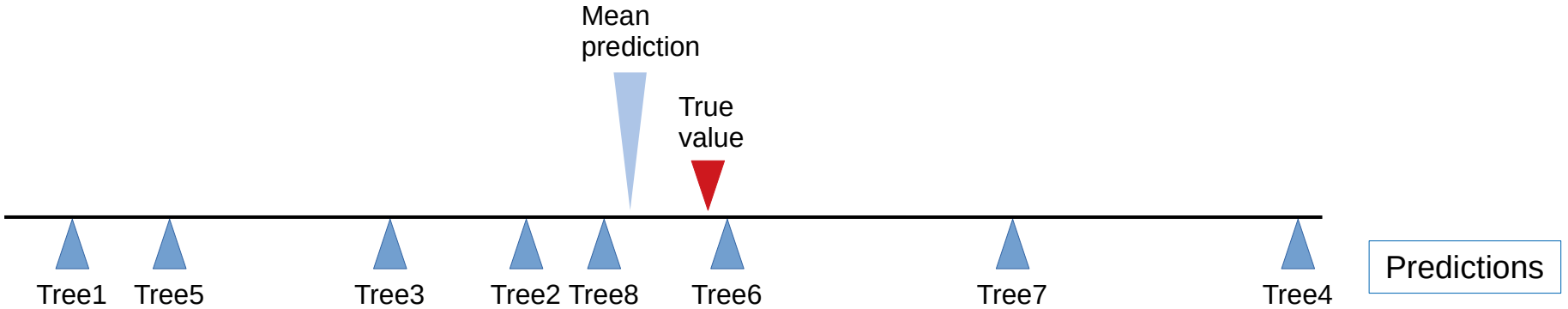


Swiss Paraplegic Research

Ensemble methods: Bagging

- **Bagging** (bootstrap aggregating) is an ensemble method typically applied to decision trees (but can be applied to all classifiers)
- Main idea: Fully grown decision trees have **low bias** but **high variance**...
  - ...it would be nice to have multiple independent samples from the same population to **reduce variance** by taking the mean of prediction
- By averaging the individual trees the variance of the estimation is reduced

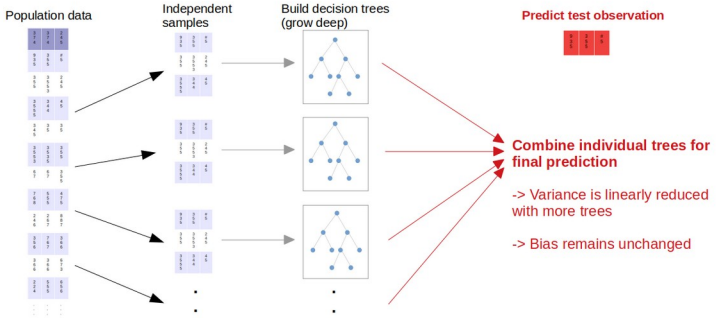


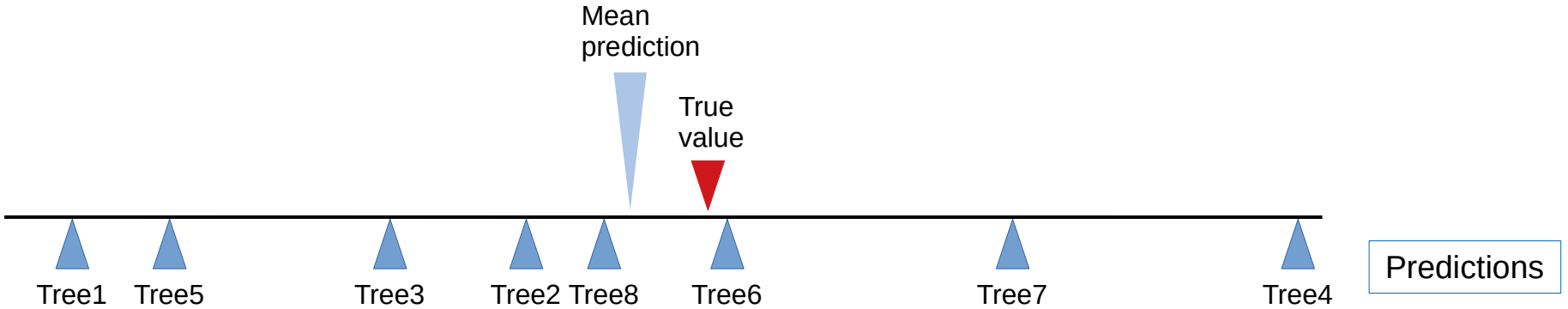


Swiss Paraplegic Research

Ensemble methods: Bagging

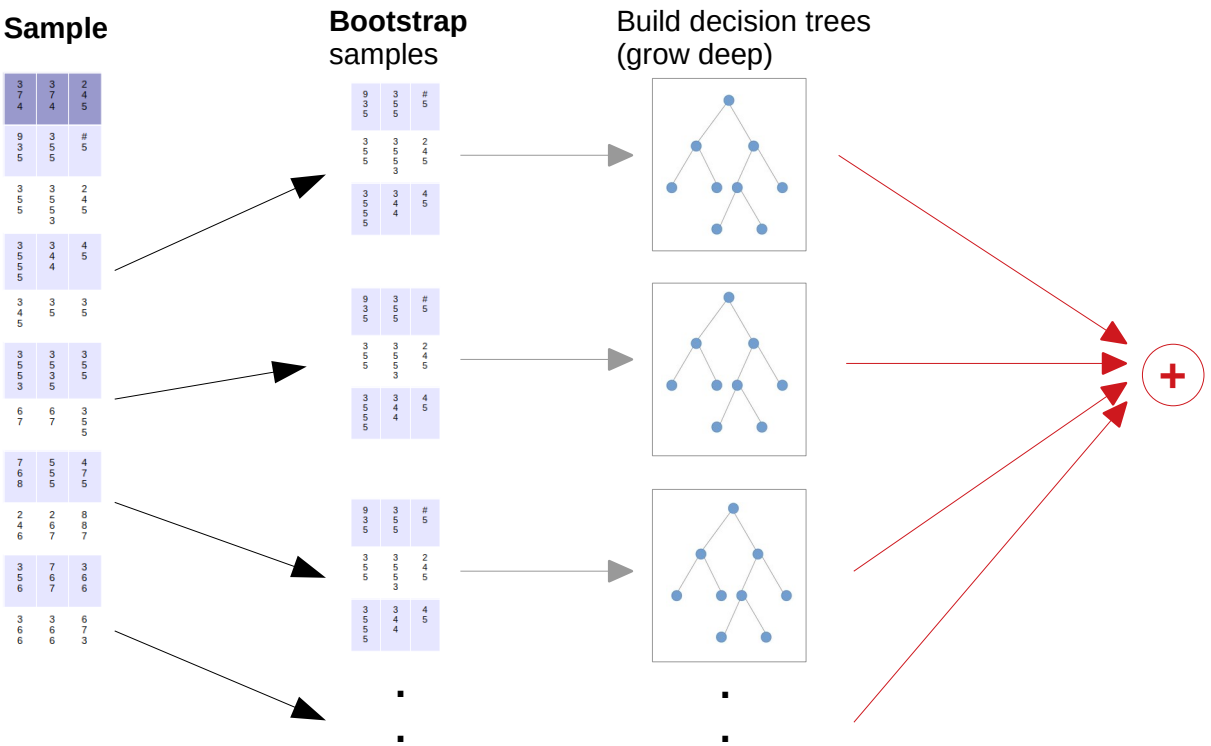
- **Bagging** (bootstrap aggregating) is an ensemble method typically applied to decision trees (but can be applied to all classifiers)
- Main idea: Fully grown decision trees have **low bias** but **high variance**...
  - ...it would be nice to have multiple independent samples from the same population to **reduce variance** by taking the mean of prediction
- By averaging the individual trees the variance of the estimation is reduced
- **However**: In reality we usually only have **one training sample** from the population





Ensemble methods: Bagging

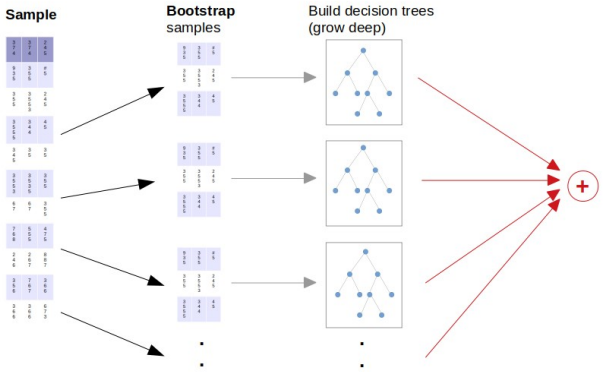
- In reality we usually only have only **one sample** from the population
  - Solution: We now take **bootstrap samples** from our training data (originally suggested by Breiman 1996)
  - Grow deep decision trees on bootstrap samples and combine the trees for prediction



Swiss Paraplegic Research

Ensemble methods: Bagging

- In reality we usually only have only **one sample** from the population
  - Solution: We now take **bootstrap samples** from our training data (originally suggested by Breiman 1996)
  - Grow deep decision trees on bootstrap samples and combine the trees for prediction
- Bootstrap samples are samples drawn with replacement:



Original data

4	5	22	3	99	67	43	1	21	2

Bootstrap samples (same size as original data)

3	5	67	5	43	67	21	2	99	3
2	4	4	67	21	1	1	67	99	3
5	5	21	99	1	2	22	99	67	5

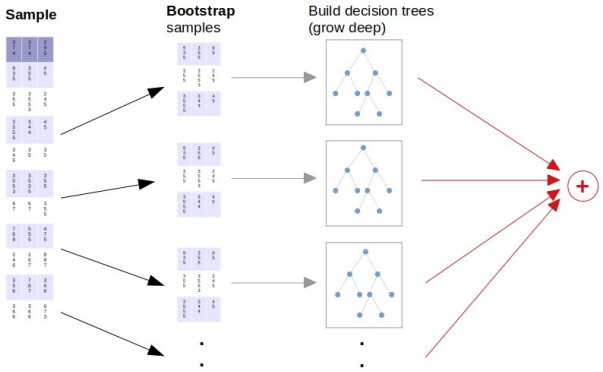
.
.

Breiman L: Bagging Predictors. Machine Learning 1996, 24(2):123-140.

Swiss Paraplegic Research

Ensemble methods: Bagging

- In reality we usually only have only **one sample** from the population
  - Solution: We now take **bootstrap samples** from our training data (originally suggested by Breiman 1996)
  - Grow deep decision trees on bootstrap samples and combine the trees for prediction
- Bootstrap samples are samples drawn with replacement:



Original data

4	5	22	3	99	67	43	1	21	2

Bootstrap samples (same size as original data)

3	5	67	5	43	67	21	2	99	3
2	4	4	67	21	1	1	67	99	3
5	5	21	99	1	2	22	99	67	5

Same observation can appear multiple times!

⋮

Breiman L: Bagging Predictors. Machine Learning 1996, 24(2):123-140.

Swiss Paraplegic Research

Ensemble methods: Bagging

- In reality we usually only have only **one sample** from the population
  - Solution: We now take **bootstrap samples** from our training data (originally suggested by Breiman 1996)
  - Grow deep decision trees on bootstrap samples and combine the trees for prediction
- Bootstrap samples are samples drawn with replacement:

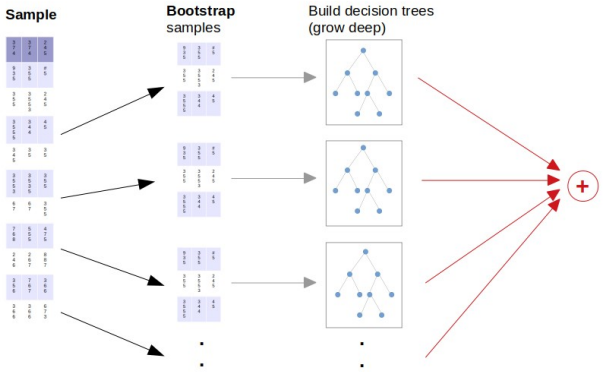
Original data

4	5	22	3	99	67	43	1	21	2

Bootstrap samples (same size as original data)

3	5	67	5	43	67	21	2	99	3
2	4	4	67	21	1	1	67	99	3
5	5	21	99	1	2	22	99	67	5

Same observation can appear multiple times!



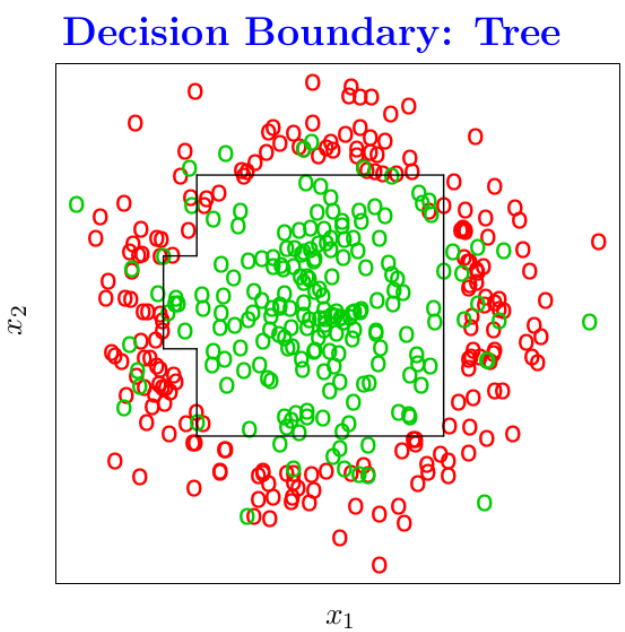
Bagging shows better performance than one decision tree applied to original data (why?)

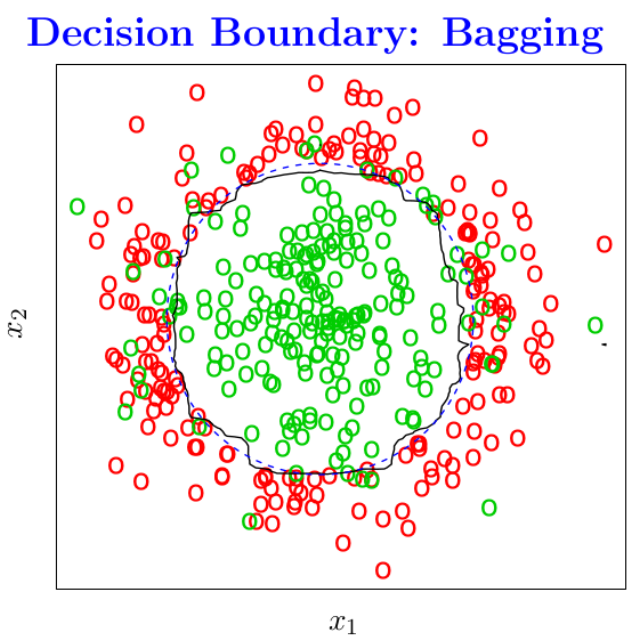
Note: Individual bootstrap samples are not independent!

-> Variance is only reduced sublinearly

Swiss
Paraplegic
Research

Ensemble methods: Bagging

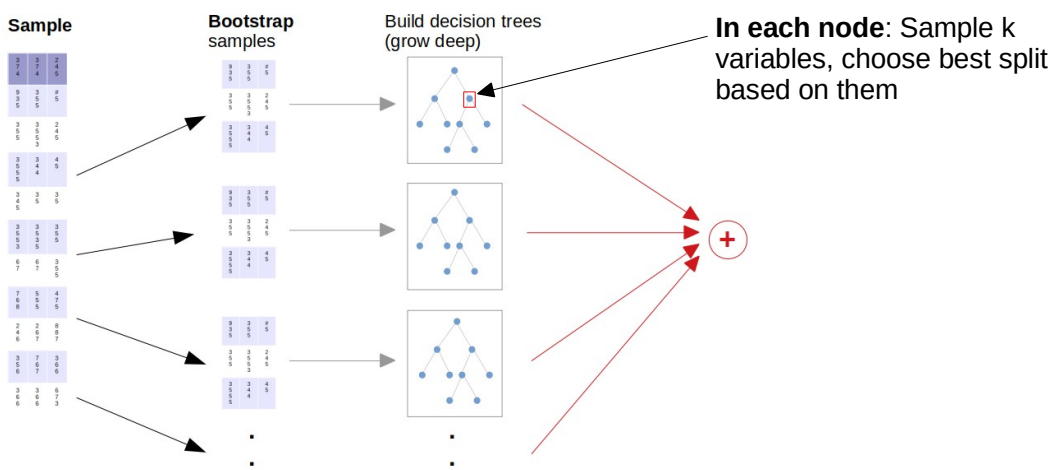




Source: Ji Zhu, University of Michigan

Ensemble methods: Random Forest

- **Random forest** is an extension of bagging
  - Main idea: Do not only sample from the data but also from the **variables**, which are used for splitting
- In random forest the trees are generated in the following way
  - Generate bootstrap samples from original data (same like bagging)
  - Build a decision tree on each bootstrap sample, but...
  - ... at each node of a decision tree, **randomly select k variables**, which are evaluated for splitting. Choose the best split (using only the k variables).

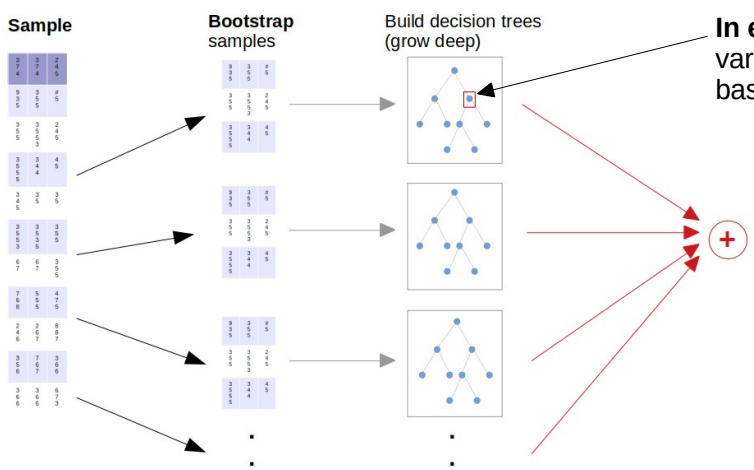


Swiss Paraplegic Research

Ensemble methods: Random Forest

- **Random forest** is an extension of bagging
  - Main idea: Do not only sample from the data but also from the **variables**, which are used for splitting
- In random forest the trees are generated in the following way
  - Generate bootstrap samples from original data (same like bagging)
  - Build a decision tree on each bootstrap sample, but...
  - ... at each node of a decision tree, **randomly select k variables**, which are evaluated for splitting. Choose the best split (using only the k variables).

Can also use **sub-samples** instead of bootstrap samples. Bootstrap sampling can also induce bias in variable selection.

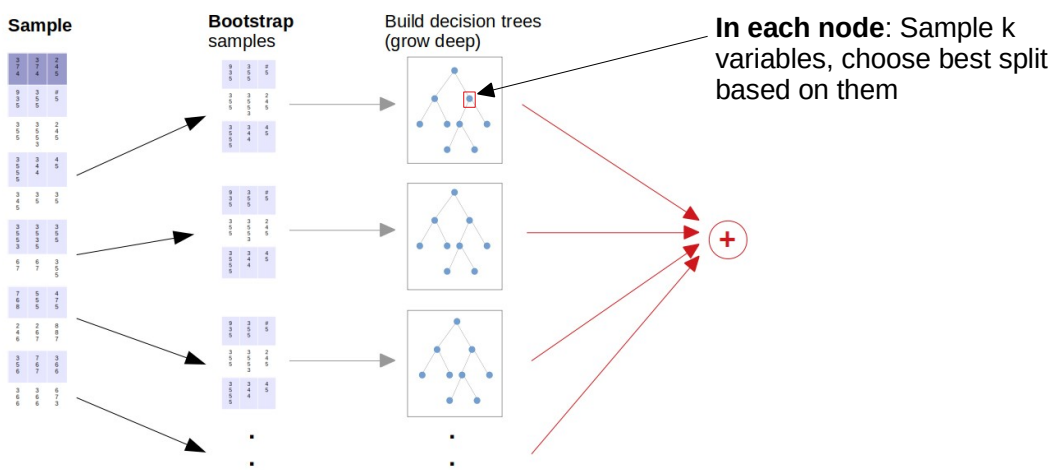


In each node: Sample k variables, choose best split based on them

Bias of bootstap sampling in RF:
Strobl, C., Boulesteix, A., Zeileis, A. et al. Bias in random forest variable importance measures: Illustrations, sources and a solution. BMC Bioinformatics 8, 25 (2007)

Ensemble methods: Random Forest

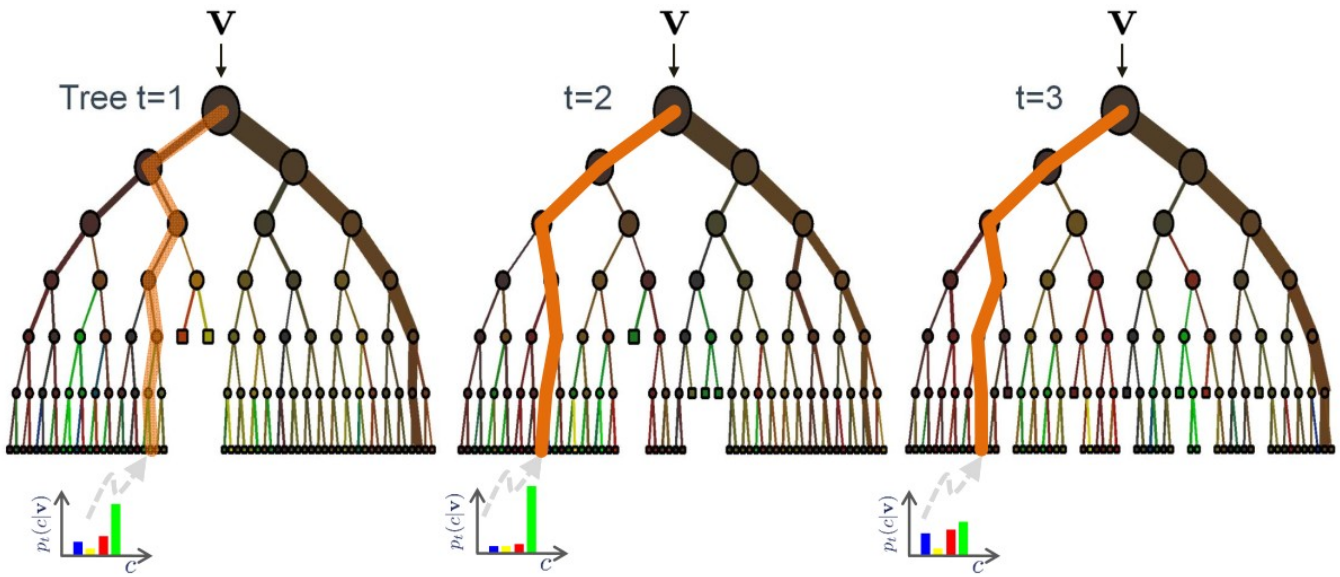
- What is the point of sampling the variables?
  - Sampling variables de-correlates the individual trees, makes them more diverse (this increases variance reduction)
  - **Intuition**: By restricting the variable selection at each node, some variables are incorporated in the analysis which might otherwise never be considered
  - This can reveal interactions in the data, which would otherwise not be detected



Swiss Paraplegic Research

Ensemble methods: Random Forest

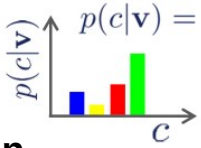
- Prediction of a new observation same like in bagging:
  - Run observation through all trees and incorporate each tree's prediction in the final prediction (e.g. majority vote for classification or mean for regression)
- Example of aggregating the tree's results for **classification**:



Source: Microsoft

Average probability:

$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$$



Predicted category: green

Swiss Paraplegic Research

Random Forest in R

- We will use the **cforest()** function from the package "partykit" (unbiased variable selection)

```
### Cforest with partykit:
library("partykit")
set.seed(123)
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)
### Confusion matrix (OOB predictions):
preds_oob <- predict(rf_smoke, OOB = TRUE)
confT <- table(dat_smoking$intention_to_smoke, preds_oob)
confT
     preds_oob
      no yes
  no  93  20
  yes 25  62
### Calculate (OOB) misclassification rate:
confT_mis <- confT
diag(confT_mis) <- 0
sum(confT_mis)/sum(confT)
0.23
### Make predictions for "new" observations:
predict(rf_smoke, newdata = dat_smoking[1:4,], type = 'response')
  1   2   3   4
 no yes  no yes
Levels: no yes
```

The used "**smoking**" data includes data from teenagers and the target variable is whether a person intends to start smoking (yes/no). The predictor variables are:
lied_to_parents,
alcohol_per_month,
age,
friends_smoke

Swiss Paraplegic Research

Random Forest in R

- We will use the **cforest()** function from the package "partykit" (unbiased variable selection)

ntree: How many trees to grow

mtry: How many variables to sample at each node

```
### Cforest with partykit:
library("partykit")
set.seed(123)
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)
### Confusion matrix (OOB predictions):
preds_oob <- predict(rf_smoke, OOB = TRUE)
confT <- table(dat_smoking$intention_to_smoke, preds_oob)
confT
     preds_oob
      no yes
  no  93  20
  yes 25  62
### Calculate (OOB) misclassification rate:
confT_mis <- confT
diag(confT_mis) <- 0
sum(confT_mis)/sum(confT)
0.23
### Make predictions for "new" observations:
predict(rf_smoke, newdata = dat_smoking[1:4,], type = 'response')
  1   2   3   4
 no yes  no yes
Levels: no yes
```

What is OOB?

The used "**smoking**" data includes data from teenagers and the target variable is whether a person intends to start smoking (yes/no). The predictor variables are:
lied_to_parents,
alcohol_per_month,
age,
friends_smoke

Out-of-bag error (OOB)

- Random Forest comes with its own integrated evaluation tool!
- Each tree is fitted to a bootstrap sample (or sub-sample) of original data
  - Thus, every tree in the forest has only seen a part of the data
- To calculate the **OOB-error**:
  - Predict each observation in the data using only the trees, which have not seen the observation when they were generated

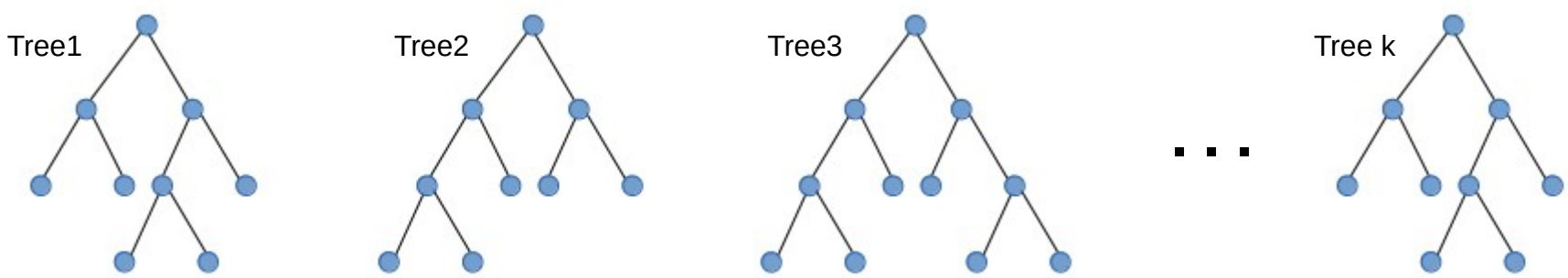
Swiss Paraplegic Research

Out-of-bag error (OOB)

- Random Forest comes with its own integrated evaluation tool!
- Each tree is fitted to a bootstrap sample (or sub-sample) of original data
  - Thus, every tree in the forest has only seen a part of the data
- To calculate the **OOB-error**:
  - Predict each observation in the data using only the trees, which have not seen the observation when they were generated

e.g. observation #24:

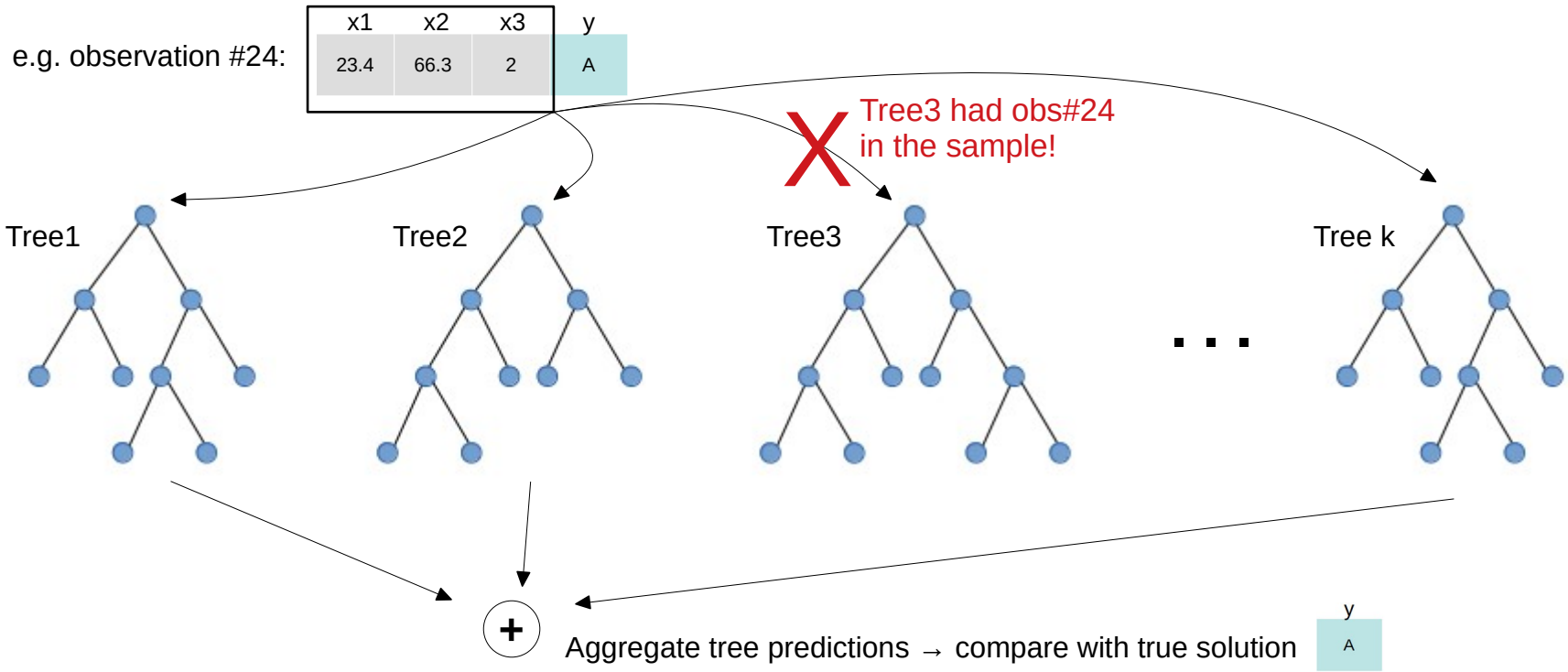
x1	x2	x3	y
23.4	66.3	2	A



Swiss Paraplegic Research

Out-of-bag error (OOB)

- Random Forest comes with its own integrated evaluation tool!
- Each tree is fitted to a bootstrap sample (or sub-sample) of original data
  - Thus, every tree in the forest has only seen a part of the data
- To calculate the **OOB-error**:
  - Predict each observation in the data using only the trees, which have not seen the observation when they were generated



Swiss Paraplegic Research

Out-of-bag error (OOB)

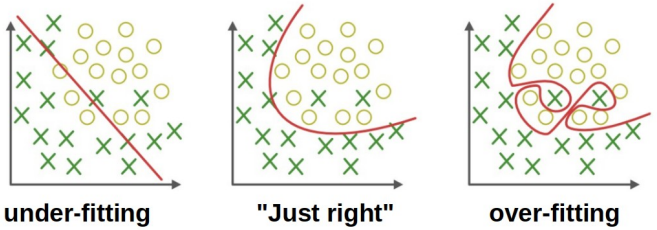
- Create OOB confusion table to look at the OOB-performance (see R-code slide)
- Use predict()-function and set OOB=TRUE, give no "newdata" argument

```
### Fit forest and calculate OOB misclassification rate:
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)
preds_oob <- predict(rf_smoke, OOB = TRUE)
confT <- table(dat_smoking$intention_to_smoke, preds_oob)
confT_mis <- confT
confT
     preds_oob
      no yes
  no  93  20
  yes 25  62
diag(confT_mis) <- 0
sum(confT_mis)/sum(confT)
0.23
```

Swiss Paraplegic Research

Tuning of Random Forests

- In machine learning, we usually tune the hyperparameters of a model to find a setting which is neither under- nor overfitting the data.
  - Use crossvalidation to evaluate model performance

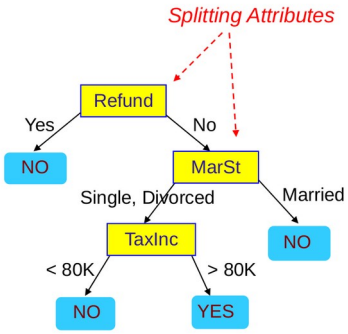


- We will look at model tuning in more detail for neural networks (next topic)
- Random forests are perhaps slightly "special" in this regard, since they are less prone to overfitting
  - Average of ensemble removes variance in predictions (RFs can still overfit in certain scenarios)
  - Random forests are said to work "off-the-shelf" (usually reach good performance with default parameter values)
- The hyperparameter **ntree** cannot be too high (only cost is computational effort)
- Ideally **mtry** (number of variables considered for splitting in each node) should be tuned.
  - There are further hyperparameters, which control the growing of the individual trees, that could be tuned

Swiss Paraplegic Research

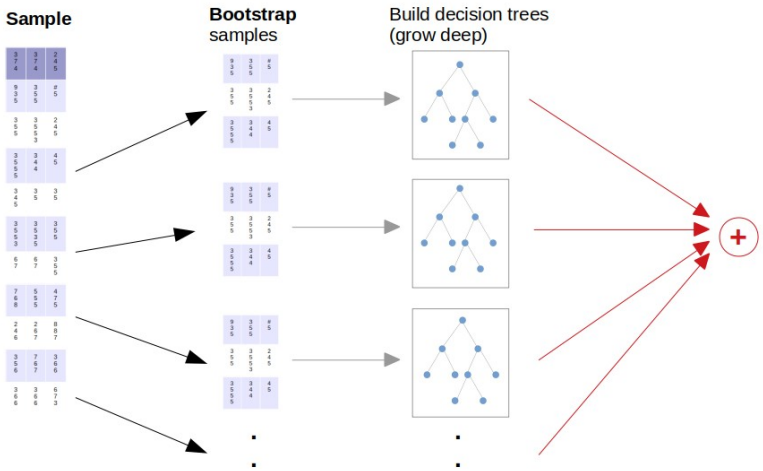
Interpretability of Random Forest

- One advantage of decision trees are their good interpretability
  - Can easily track how each variable affects the prediction
- In Random Forest we somewhat lose the interpretability ("black box")
  - Difficult to track how each variable is active in each of the e.g. 500 trees…
- How can we assess which variables are **important** for prediction?



Model: Decision Tree

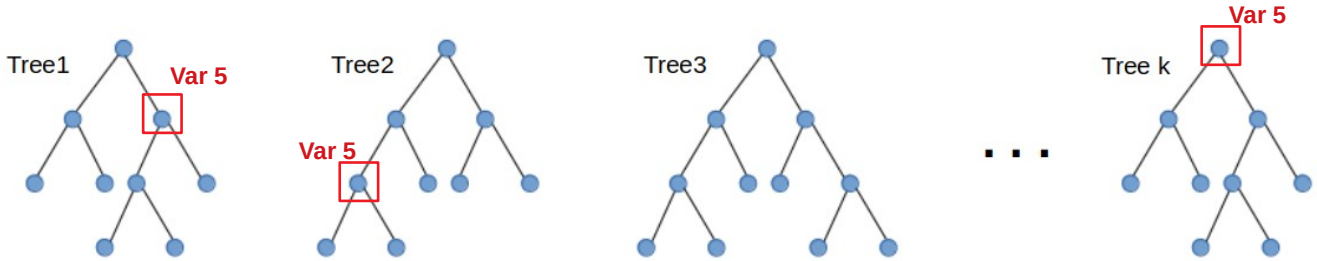
vs.



Swiss Paraplegic Research

Variable importance in RF

- How could variable importance be **scored**?
  - Simple idea: Count the number of times a variable is chosen for splitting throughout all trees
- Two more elaborate approaches:
- **Variable importance 1**: Average score improvement at splits
  - For each node where variable x was used for splitting, record the achieved improvement in the splitting score (e.g. Gini-index -> **Gini-importance score**)
  - Variable x's importance 1 score is the average of all score improvements (can be weighted with the number of data points in each node)



Swiss Paraplegic Research

Variable importance in RF

- **Variable importance 2**: Permutation importance
- **Permutation importance** is a very intuitive importance score
- Main idea: Mix up the values of variable x to break up any meaningful relation between x and the response variable (permutation)
  - Check how much the performance (usually the OOB-error) drops after permutation of x

Original data

Predictor 1	Predictor 2	Predictor 3	Target var
X1	**X2**	**X3**	**Y**
2.3	33.1	67	**A**
5.1	35.8	70.3	**B**
3.3	34.0	96	**A**
2.8	37.7	85	**C**
1.3	38.3	84.9	**A**

vs.

X1 permuted

Predictor 1	Predictor 2	Predictor 3	Target var
X1	**X2**	**X3**	**Y**
5.1	33.1	67	**A**
2.3	35.8	70.3	**B**
2.8	34.0	96	**A**
1.3	37.7	85	**C**
3.3	38.3	84.9	**A**

→ Calculate **OOB-error**

→ Calculate **OOB-error** (using RF fitted to original data)

Permutation importance: Difference in OOB-error

With stronger relation between x1 and target variable, the more should the OOB-error grow

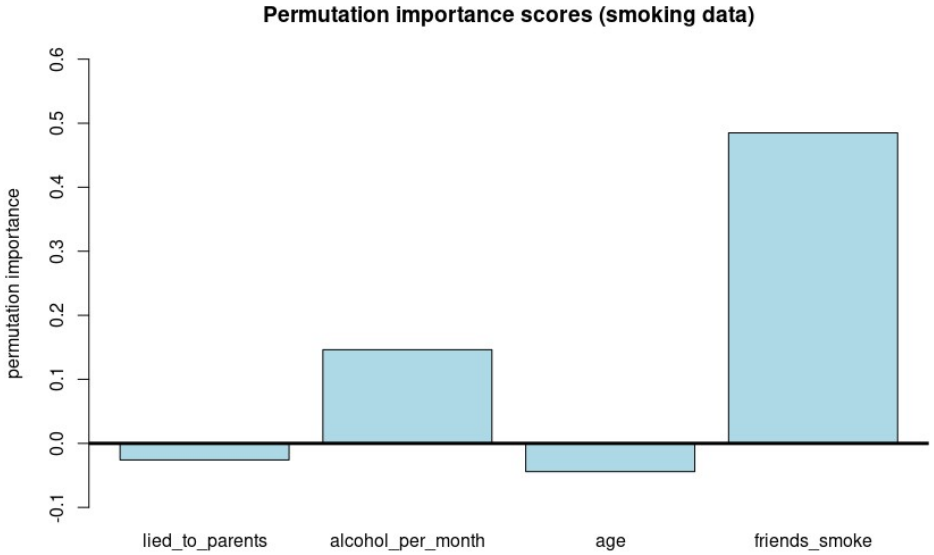
Variable importance with R

- To calculate the variable importance after fitting a Random Forest use the **varimp()**-function (default is permutation importance score)

```
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)

# Calculate variable importance scores:
set.seed(123)
imps <- varimp(rf_smoke)
imps
lied_to_parents   alcohol_per_month           age     friends_smoke
    -0.02591487          0.14622240   -0.04417460        0.48487966

# Create barplot:
barplot(imps, col='lightblue',
        main = 'Permutation importance
                scores (smoking data)',
        ylab = 'permutation importance',
        ylim = c(-0.1, 0.6))
abline(h = 0, col=1, lty=1, lwd=3)
```



Variable importance with R

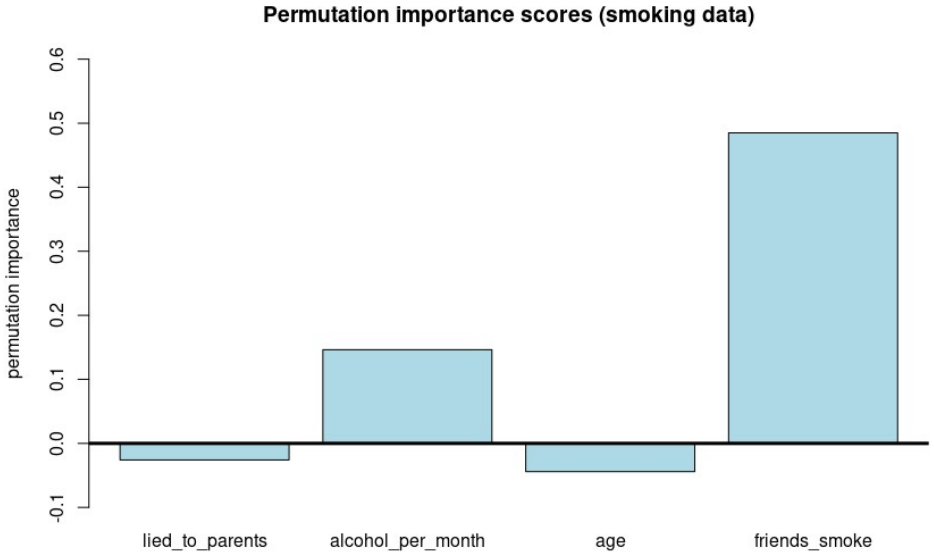
- To calculate the variable importance after fitting a Random Forest use the **varimp()**-function (default is permutation importance score)

```
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)

# Calculate variable importance scores:
set.seed(123)
imps <- varimp(rf_smoke)
imps
lied_to_parents   alcohol_per_month           age     friends_smoke
    -0.02591487          0.14622240   -0.04417460        0.48487966
```

Because importance scores are based on random permutation we should fix the random seed when calculating them

```
# Create barplot:
barplot(imps, col='lightblue',
        main = 'Permutation importance
                scores (smoking data)',
        ylab = 'permutation importance',
        ylim = c(-0.1, 0.6))
abline(h = 0, col=1, lty=1, lwd=3)
```



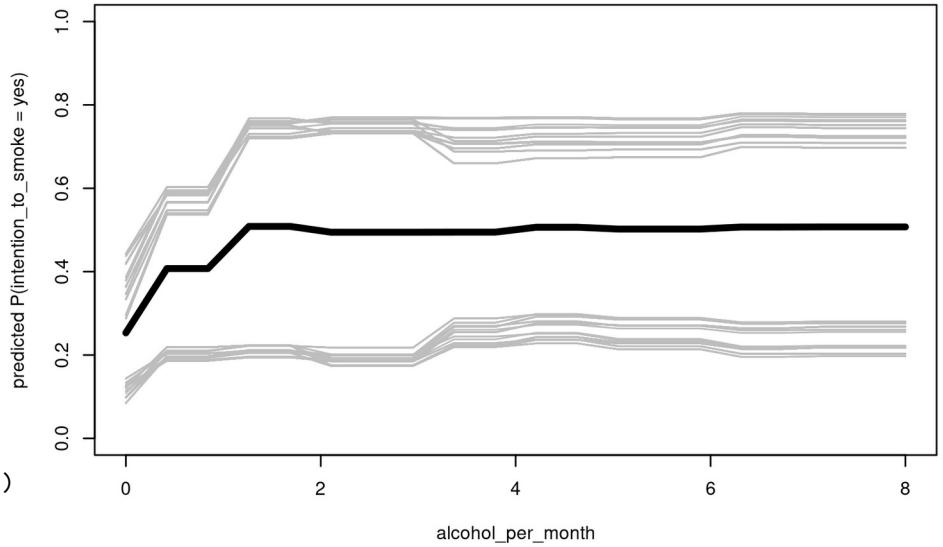
Permutation importance scores can also be negative!
→ Why?

Swiss Paraplegic Research

Partial Dependence (and ICE) Plots

- Partial dependence plots are another method to "bring light" into what is happening in the forest
- Main idea: See how the prediction of the response variable changes when only **one predictor** is shifted in its value
- Example of **partial dependence plot** for the smoking data (**response**: intention_to_smoke; **predictor**: alcohol_per_month):

```
# Self generated:
set.seed(123)
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)
steps <- seq(min(dat_smoking$alcohol_per_month), max(dat_smoking$alcohol_per_month),
             length.out=20)
predic <- matrix(NA, nrow=nrow(dat_smoking), ncol = length(steps))
for(i in 1:nrow(dat_smoking)){
  obs <- dat_smoking[i,]
  obs$alcohol_per_month <- NULL
  for(s in 1:length(steps)){
    obs$alcohol_per_month <- steps[s]
    predic[i,s] <- predict(rf_smoke, newdata=obs, type = 'prob')[2]
    print(paste0('Done with step ', s, ' of observation ', i))
  }
}
plot(NULL, xlim=c(min(dat_smoking$alcohol_per_month),
     max(dat_smoking$alcohol_per_month)),
     ylim=c(0, 1),
     xlab='alcohol_per_month', ylab='predicted P(intention_to_smoke = yes)')
for(l in 1:nrow(predic)){
  lines(x = steps, y = predic[l,], col='grey')
}
lines(x=steps, y=apply(predic, 2, mean), col=1, lwd=4)
```



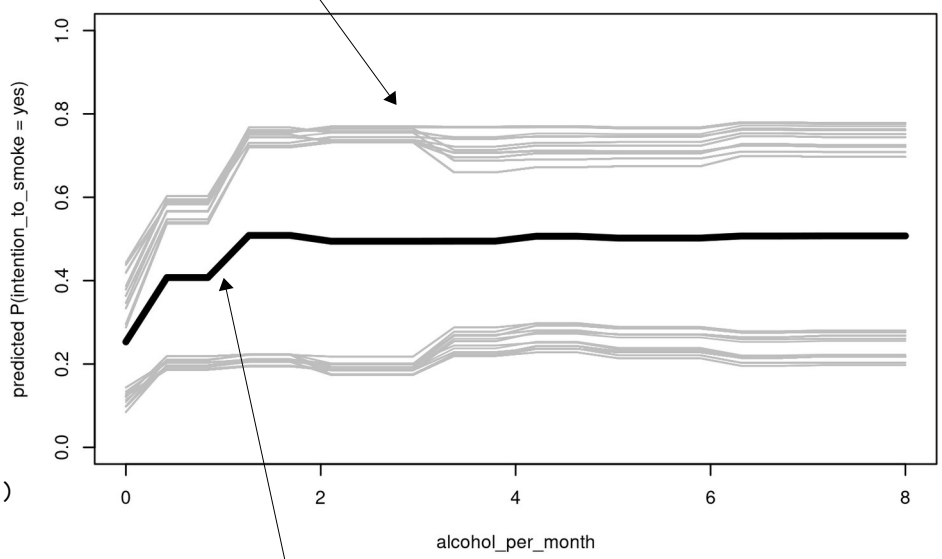
Swiss Paraplegic Research

Partial Dependence (and ICE) Plots

- Partial dependence plots are another method to "bring light" into what is happening in the forest
- Main idea: See how the prediction of the response variable changes when only **one predictor** is shifted in its value
- Example of **partial dependence plot** for the smoking data (**response**: intention_to_smoke; **predictor**: alcohol_per_month):

```
# Self generated:
set.seed(123)
rf_smoke <- cforest(intention_to_smoke ~., data = dat_smoking, ntree=500, mtry=2)
steps <- seq(min(dat_smoking$alcohol_per_month), max(dat_smoking$alcohol_per_month),
             length.out=20)
predic <- matrix(NA, nrow=nrow(dat_smoking), ncol = length(steps))
for(i in 1:nrow(dat_smoking)){
  obs <- dat_smoking[i,]
  obs$alcohol_per_month <- NULL
  for(s in 1:length(steps)){
    obs$alcohol_per_month <- steps[s]
    predic[i,s] <- predict(rf_smoke, newdata=obs, type = 'prob')[2]
    print(paste0('Done with step ', s, ' of observation ', i))
  }
}
plot(NULL, xlim=c(min(dat_smoking$alcohol_per_month),
     max(dat_smoking$alcohol_per_month)),
     ylim=c(0, 1),
     xlab='alcohol_per_month', ylab='predicted P(intention_to_smoke = yes)')
for(l in 1:nrow(predic)){
  lines(x = steps, y = predic[l,], col='grey')
}
lines(x=steps, y=apply(predic, 2, mean), col=1, lwd=4)
```

Grey lines: "**Individual Conditional Expectation**" (ICE) → Predictions for individual cases

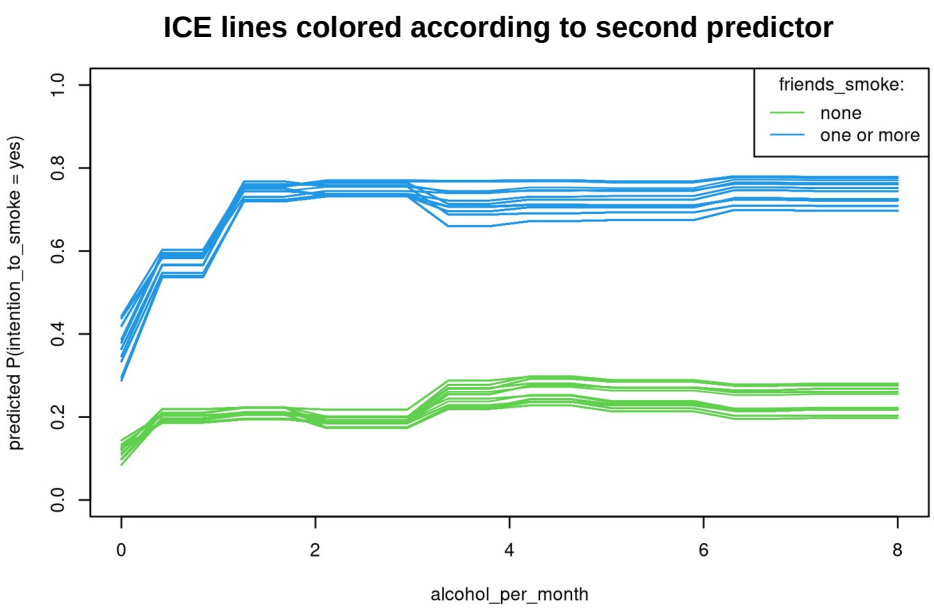


Black line: "**Partial dependence**" → mean of the ICE lines

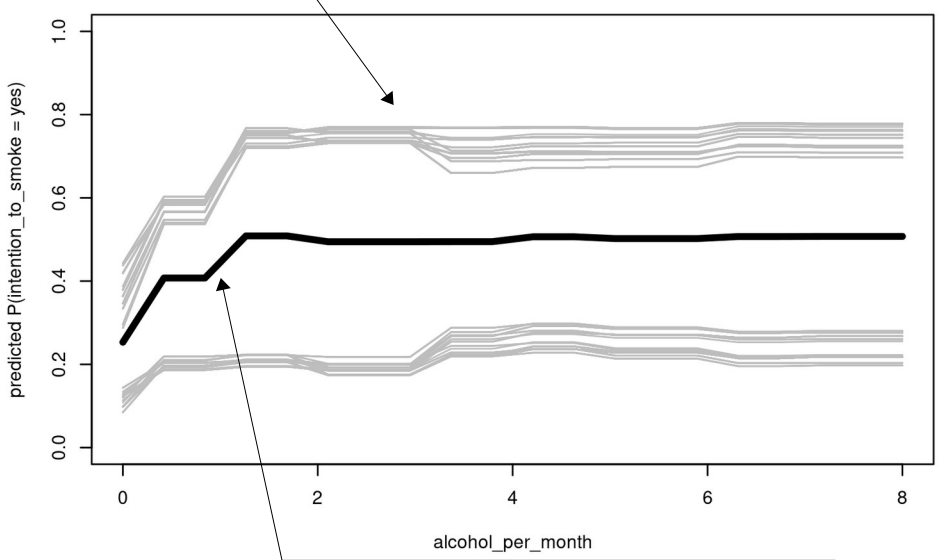
Swiss Paraplegic Research

Partial Dependence (and ICE) Plots

- Partial dependence plots are another method to "bring light" into what is happening in the forest
- Main idea: See how the prediction of the response variable changes when only **one predictor** is shifted in its value
- Example of **partial dependence plot** for the smoking data (**response**: intention_to_smoke; **predictor**: alcohol_per_month):



Grey lines: "I**ndividual Conditional Expectation**" (ICE) → Predictions for individual cases

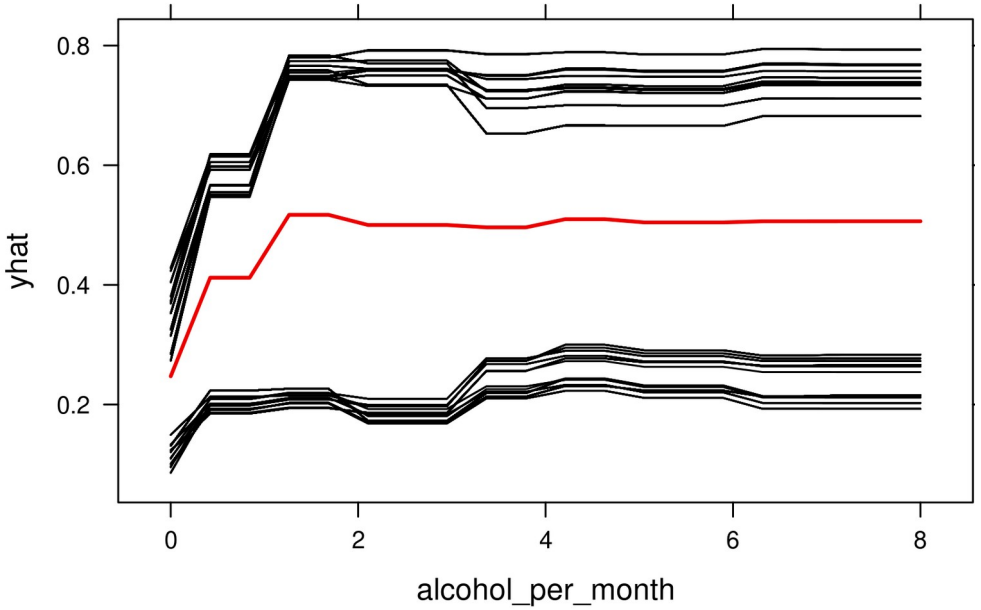


Black line: "**Partial dependence**" → mean of the ICE lines

Partial dependence plots with "pdp"

- "**pdp**" is an R package to construct partial dependence plots
- Can be applied to different Machine Learning methods
  - Also works for Random Forests fitted with cforest()

```
library(pdp)
partial(rf_smoke, pred.var = "alcohol_per_month",
        grid.resolution = 20,   # Resolution of prediction grid
        prob = TRUE,            # Return predicted probability ...
        which.class = 2,        # ...for which class of y
        ice = TRUE,             # Generate ICE curves
        plot = TRUE)            # Create plot
```



- See also the **iml** R package for a collection of interpretation techniques

Summary Random Forest

- **Advantages:**
  - Random Forest is a very effective ML method (good performance in ML-competitions)
  - Is non-parametric, poses no assumption regarding distribution of variables or residuals
  - Comes with included performance evaluation (OOB-error)
  - Random Forests less prone to overfitting
  - Can handle thousands of input variables
  - Can handle lots of noise variables with only few relevant variables
  - Random Forest can detect strong and local interactions
  - Is robust against outliers
  - It can handle imbalanced data (e.g. using weighted sampling)
  - ...

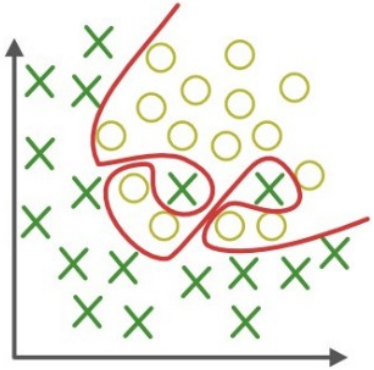
Summary Random Forest

- **Disadvantages:**
  - Less interpretability, feeling of a "black box"
  - Can not extrapolate predictions beyond the range of training data well
  - Takes more time to train than e.g. decision tree

Swiss Paraplegic Research

Ensemble methods: Boosting

Bagging + RF:

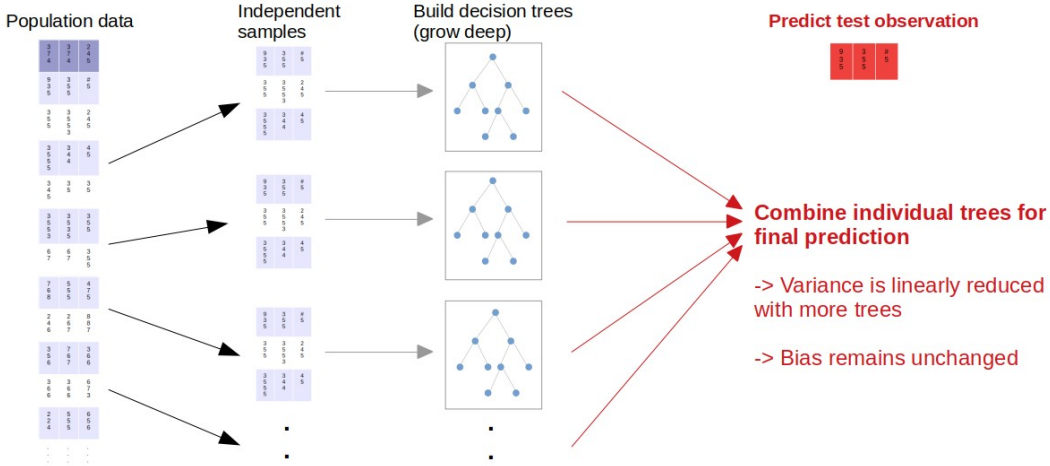


Over-fitting base model (linear model or **trees**)

Avoid weakness of base model by creating an **ensemble** of models

Bagging

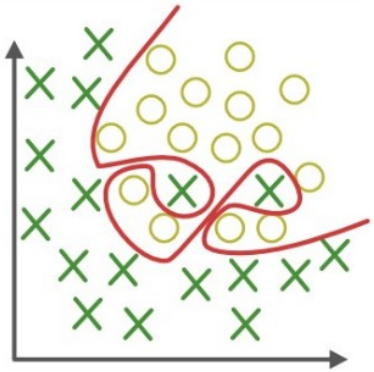
Random Forests



Swiss Paraplegic Research

Ensemble methods: Boosting

Bagging + RF:



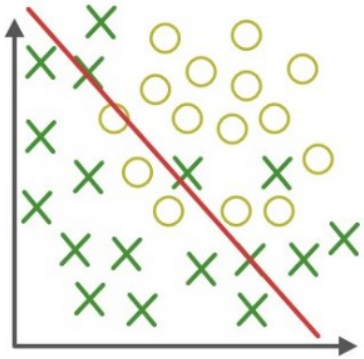
Over-fitting base model (linear model or **trees**)

Avoid weakness of base model by creating an **ensemble** of models

Bagging

Random Forests

Boosting:



Under-fitting base model (linear model or **trees**)

Avoid weakness of base model by creating an **ensemble** of models

Adaptive boosting

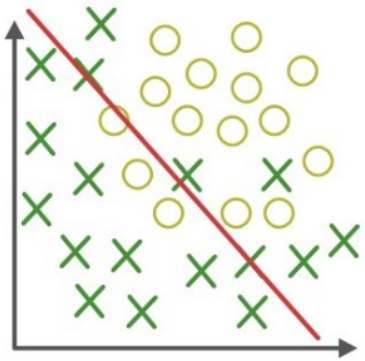
Gradient boosting

Swiss Paraplegic Research

Gradient boosting

- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure...

Boosting:



Under-fitting base model (linear model or **trees**)

Avoid weakness of base model by creating an **ensemble** of models

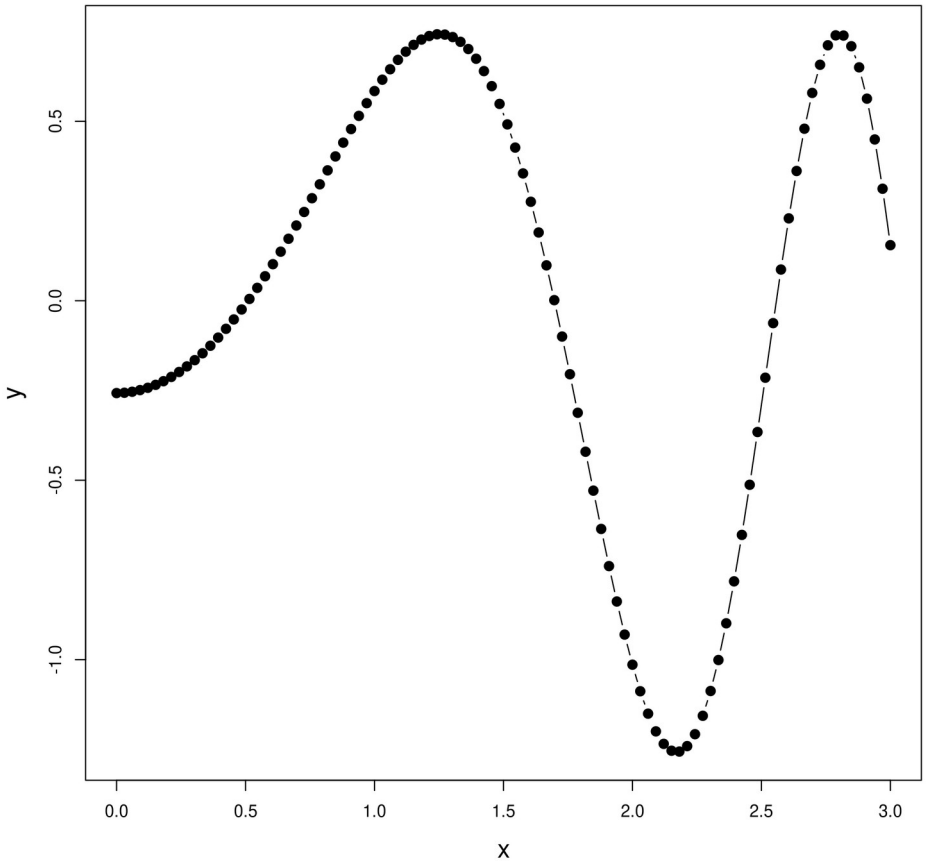
Adaptive boosting

Gradient boosting

Swiss Paraplegic Research

Gradient boosting

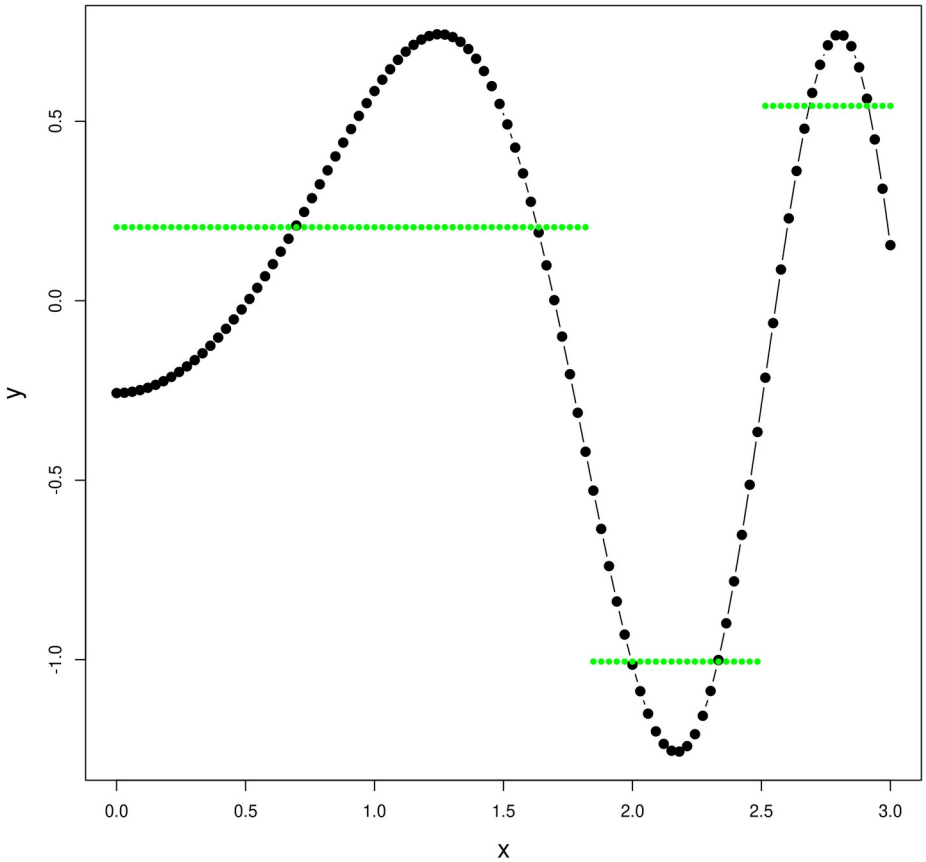
- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure...



Swiss Paraplegic Research

Gradient boosting

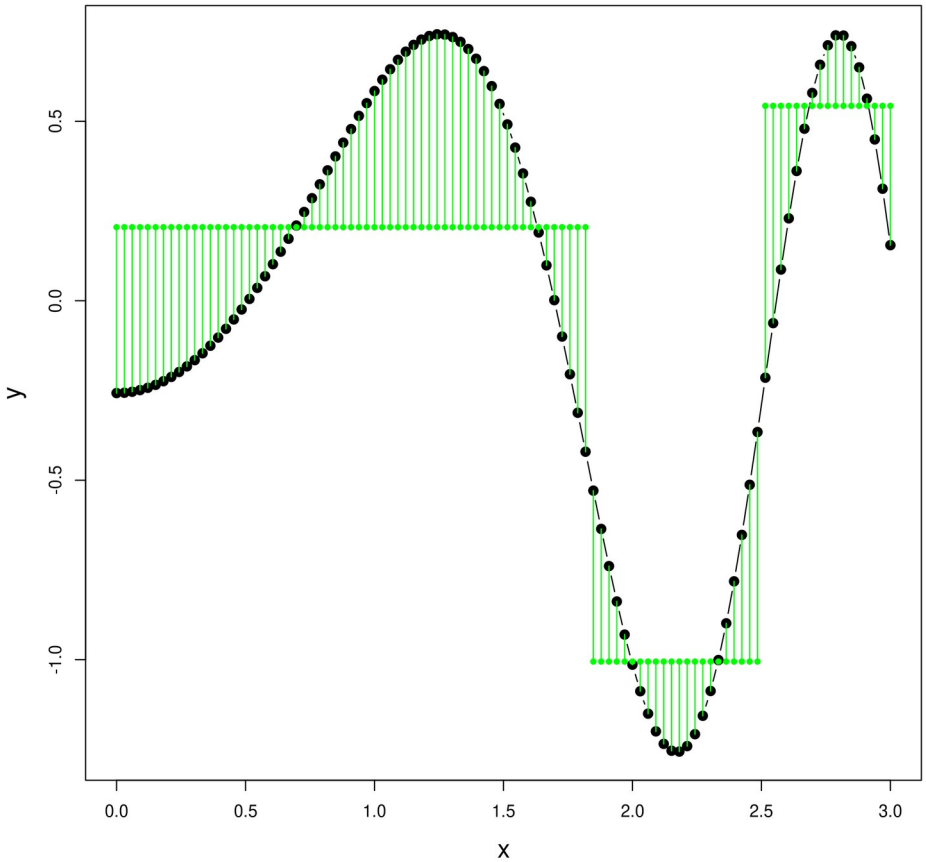
- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure...



Swiss Paraplegic Research

Gradient boosting

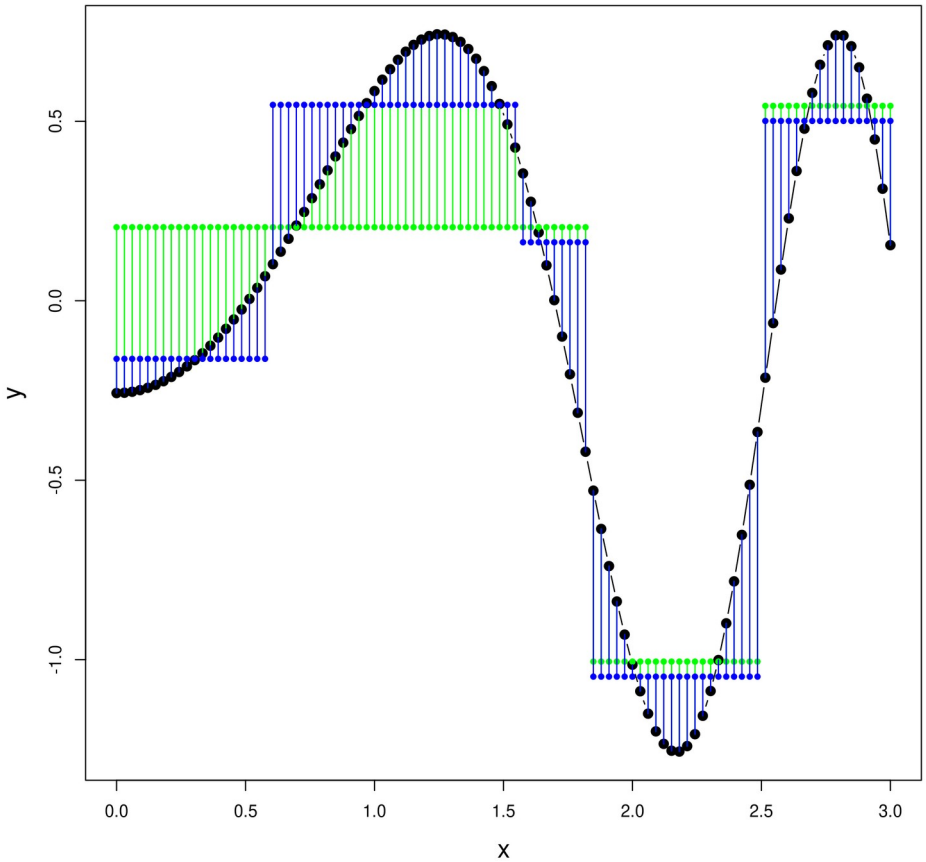
- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure...



Swiss Paraplegic Research

Gradient boosting

- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure...



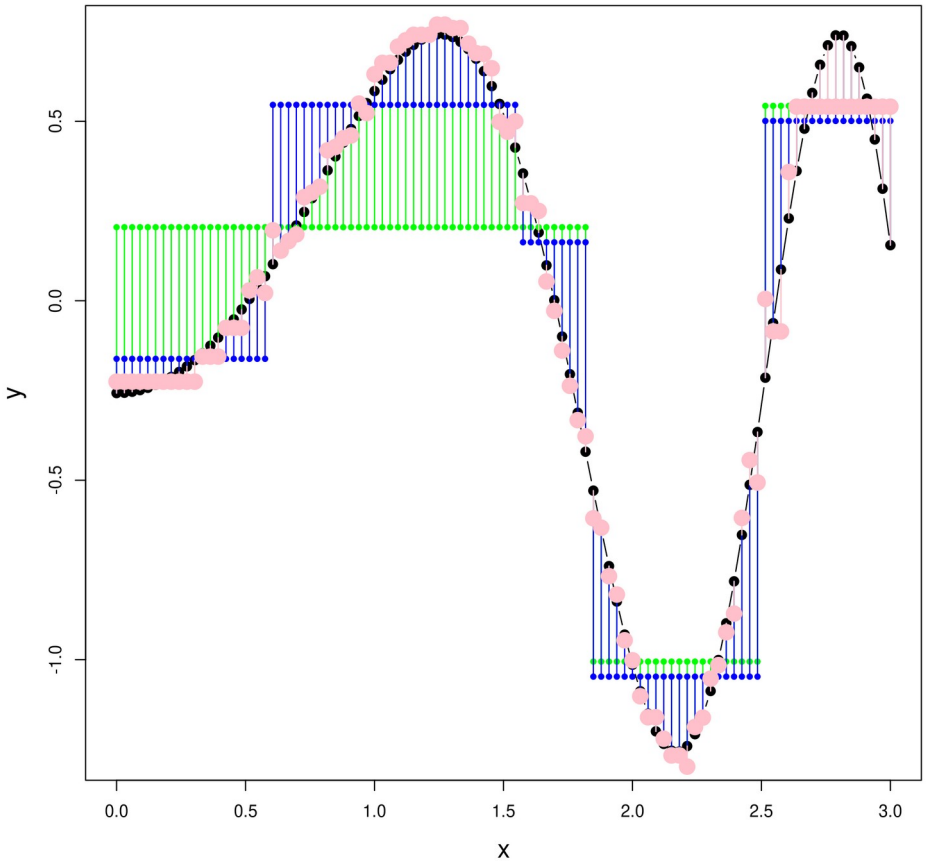
Swiss Paraplegic Research

Gradient boosting

- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure…

$$M = M_1 + \eta \sum \gamma_i T_i$$

The final prediction of the model is updated by adding (scaled) predictions of following trees.
η is the learning rate → determines how much the model is updated with each tree (small values recommended, e.g. 0.1)



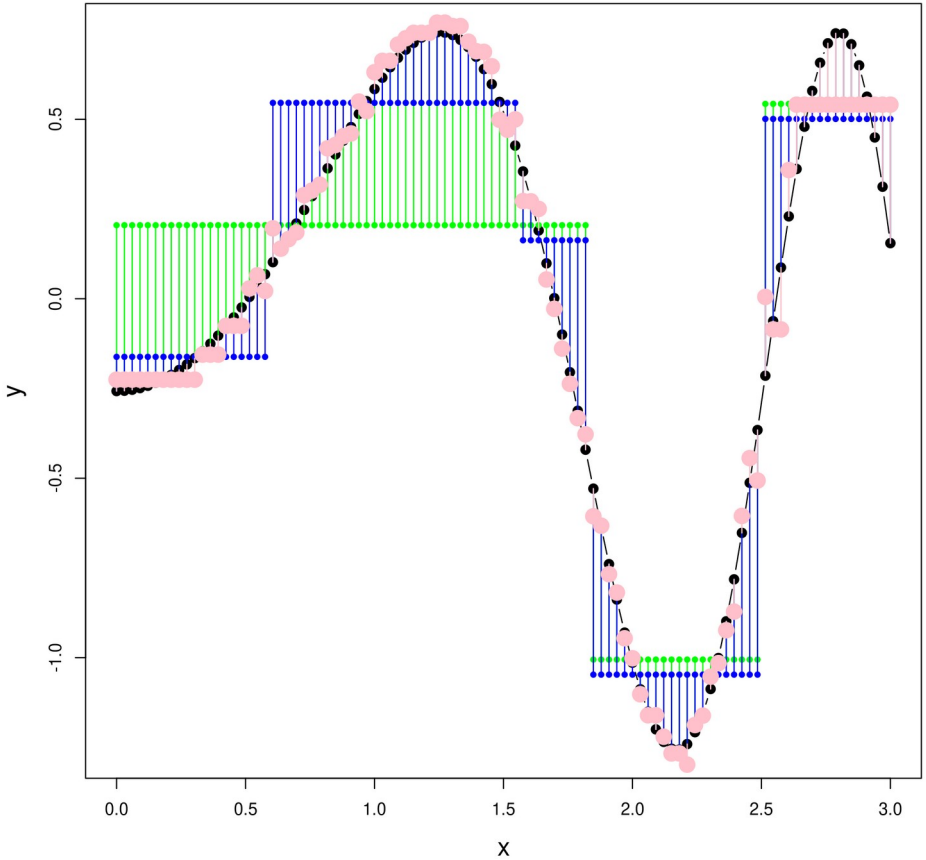
Swiss Paraplegic Research

Gradient boosting

- **General intuition behind gradient boosting:**
  - Fit a weak learner (usually shallow tree) to your data and check training error
  - Express through "residuals" (more specifically gradient of loss, later) the shortcomings of the initial model
  - Fit a new shallow tree **to the "residuals"** of the initial model
  - The predictions of the model now consist of the sum of the predictions from the first and the second tree
  - Add further trees following this procedure…

$$M = M_1 + \eta \sum \gamma_i T_i$$

The final prediction of the model is updated by adding (scaled) predictions of following trees.
η is the learning rate → determines how much the model is updated with each tree (small values recommended, e.g. 0.1)



Show boosting progression (Gif)

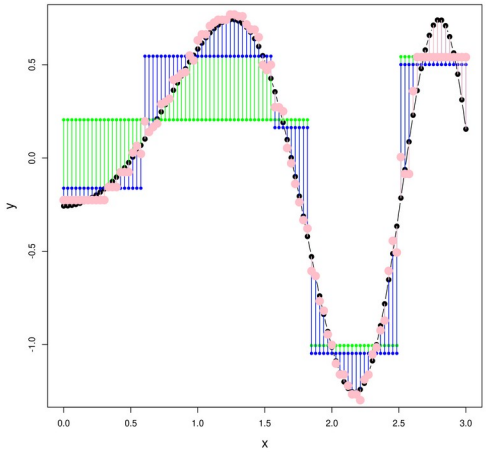
Swiss Paraplegic Research

"Gradient" boosting

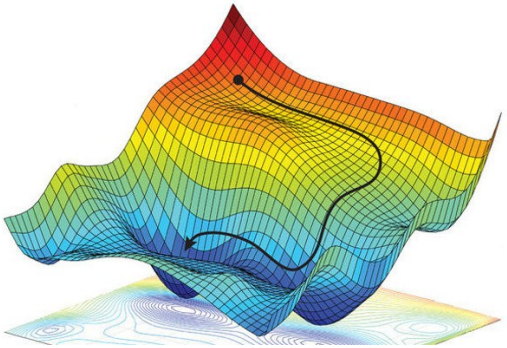
- The "residuals" to which subsequent trees are fitted are the negative gradients of the problem-specific **Loss** (for each data point)

$$-g(x_i) = \frac{-\partial Loss(y_i, M(x_i))}{\partial M(x_i)}$$

- Using this framework allows the application of boosting to various other problems (e.g. multiclass classification)
  - Friedman/Hastie/Tibshirani(2000): Generalization to a variety of loss functions
- Gradient boosting is an iterative **gradient descent** algorithm



$$M = M_1 + \eta \sum \gamma_i T_i$$

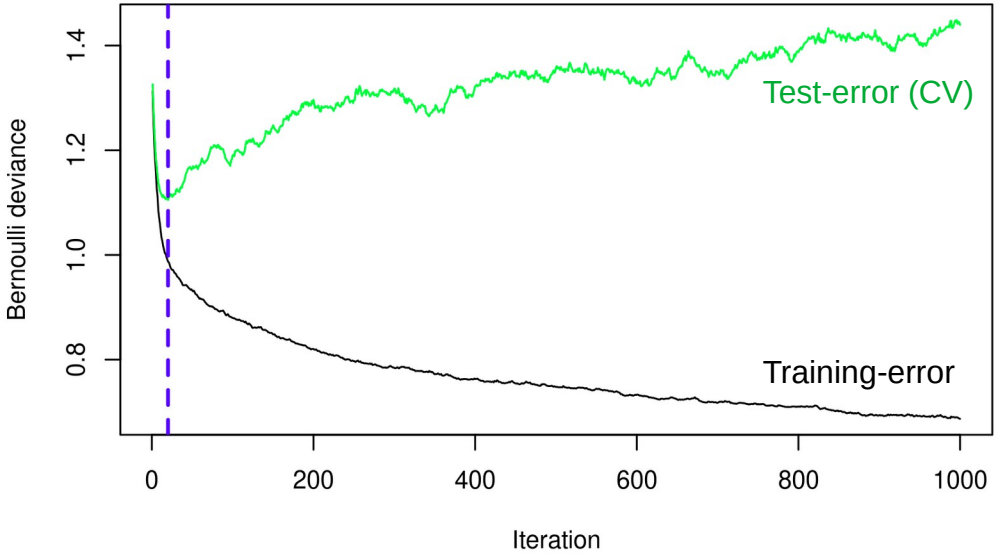


Source: Amini et al. 2018

Swiss Paraplegic Research

Gradient boosting in R (smoking data)

```
### Gradient boosting with gbm:
library("gbm")
### Turn intention to smoke to 0, 1 variable:
d <- dat_smoking
d$intention_to_smoke <- as.numeric(dat_smoking$intention_to_smoke)-1
### Fit gradient boosting ensemble:
boost_smk <- gbm(intention_to_smoke ~., data=d,
                 distribution = 'bernoulli',
                 n.trees = 1000,
                 shrinkage = 0.1,
                 interaction.depth = 3,
                 cv.folds = 10)
### Find best number of iterations, i.e. trees:
gbm.perf(boost_smk, method = 'cv')
[1] 22
### Make predictions (probabilites):
predict(boost_smk, newdata=d[1:3,],
      type = 'response', n.trees = 22)
[1] 0.4046658 0.7258298 0.1103924
```



Swiss Paraplegic Research

Gradient boosting in R (smoking data)

```
### Gradient boosting with gbm:
library("gbm")
### Turn intention to smoke to 0, 1 variable:
d <- dat_smoking
d$intention_to_smoke <- as.numeric(dat_smoking$intention_to_smoke)-1
### Fit gradient boosting ensemble:
boost_smk <- gbm(intention_to_smoke ~., data=d,
                 distribution = 'bernoulli',
                 n.trees = 1000,
                 shrinkage = 0.1,
                 interaction.depth = 3,
                 cv.folds = 10)
### Find best number of iterations, i.e. trees:
gbm.perf(boost_smk, method = 'cv')
[1] 22
### Make predictions (probabilites):
predict(boost_smk, newdata=d[1:3,],
      type = 'response', n.trees = 22)
[1] 0.4046658 0.7258298 0.1103924
```

For binary classification y must be a numeric variable (0,1)

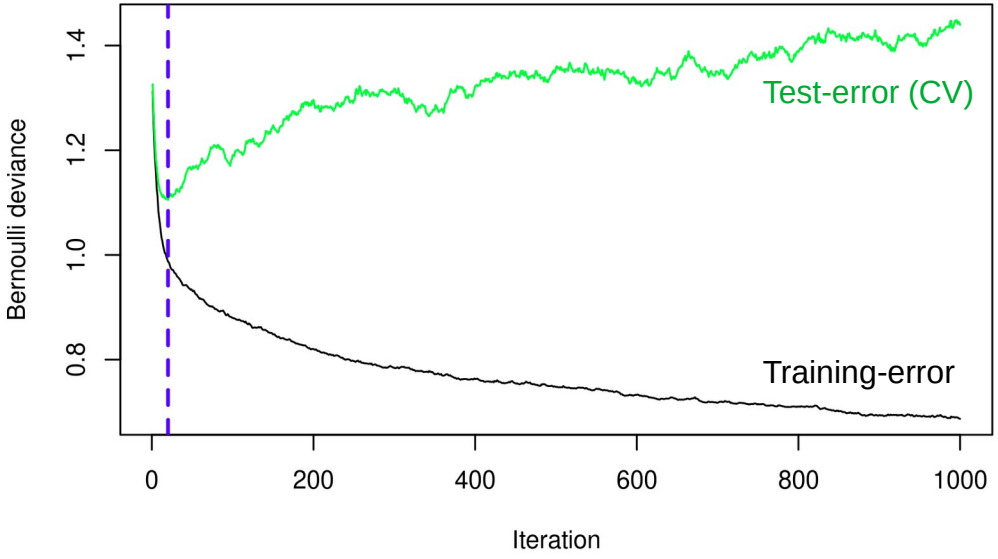
Specify type of y (will be guessed if not supplied)

How many trees

Learning rate

Maximal depth of trees

Run 10-fold crossvalidation for each iteration



Alternative R-package:
xgboost ("extreme" gradient boosting)
→ Specific implementation of gradient boosting
→ Theory similar to gradient boosting
→ Faster implementation

Further reading

- Strobl, C., Malley, J., & Tutz, G. (2009). **An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and Random Forests**. Psychological Methods, 14(4), 323–348. https://doi.org/10.1037/a0016973