FH KREMS
UNIVERSITY OF APPLIED SCIENCES / AUSTRIA

Machine Learning, Artificial Intelligence, and Big Data Analytics (IL, 4th Semester)

Lecture 5

Agenda

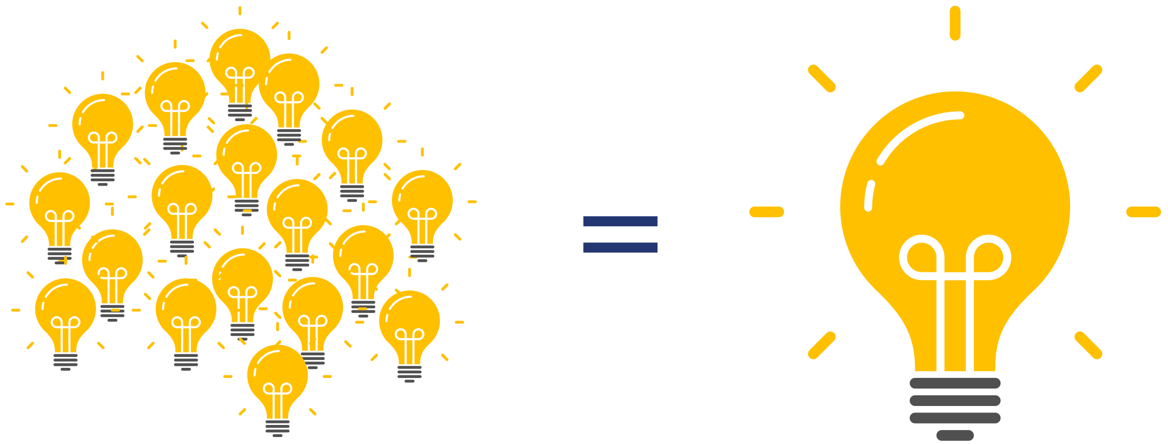
- Lecture 2: Generalization error, bias and variance, data splitting, Cross-validation.
- Lecture 3: Model evaluation
- Lecture 4: Decision Trees – Grow, Splitting and Stopping criteria, Prune, Final Evaluation
- EL: SVM (datacamp)

Today

- Introduction to ensemble models
- Random Forest, ADAboost and XGBoost
- New Assignment
- Coding session

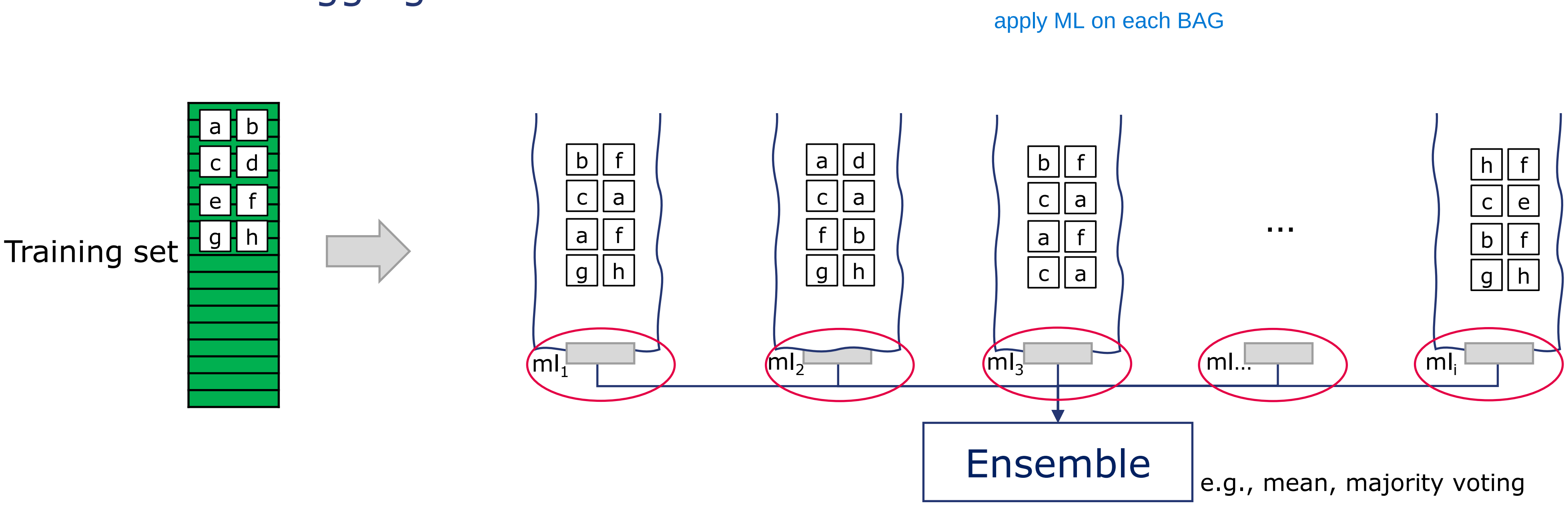
Ensemble models

- A highly accurate model can be obtained by a collection of multiple individual *weak learners*



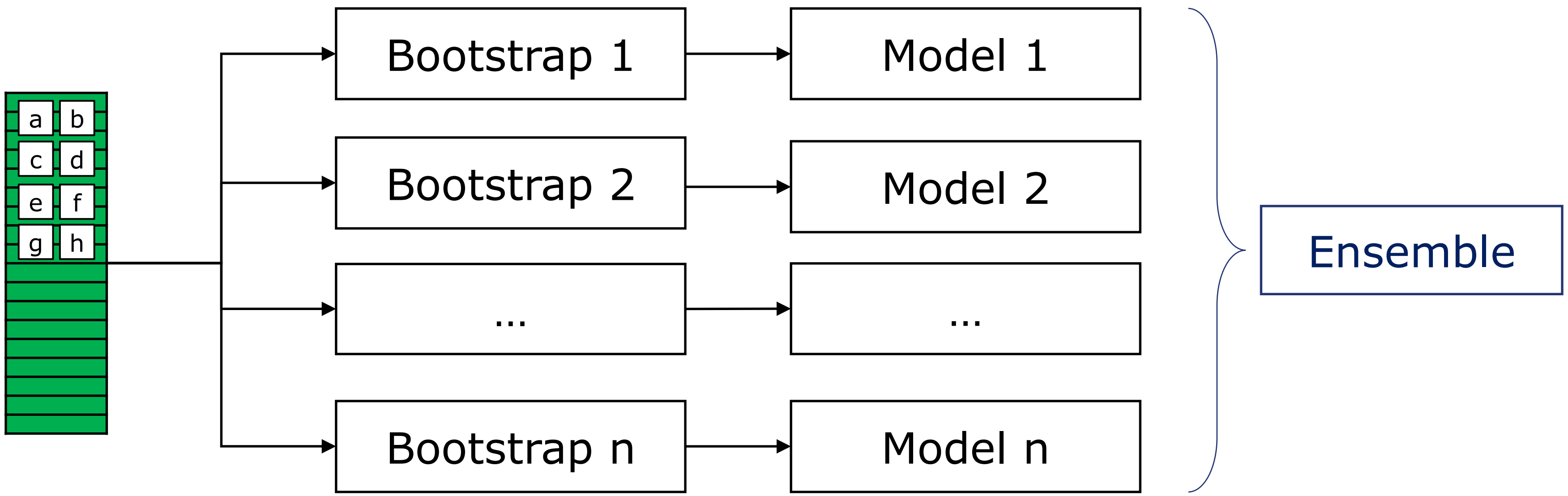
Ensemble models

- Remember bagging?



Ensemble models

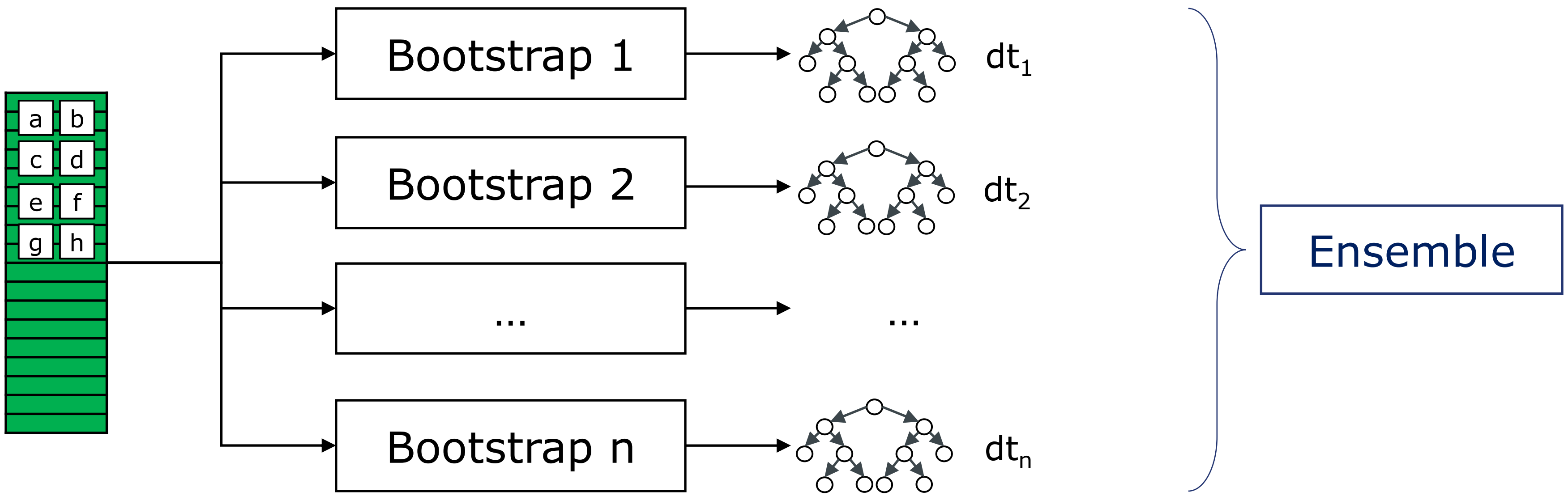
Bagging



Ensemble models

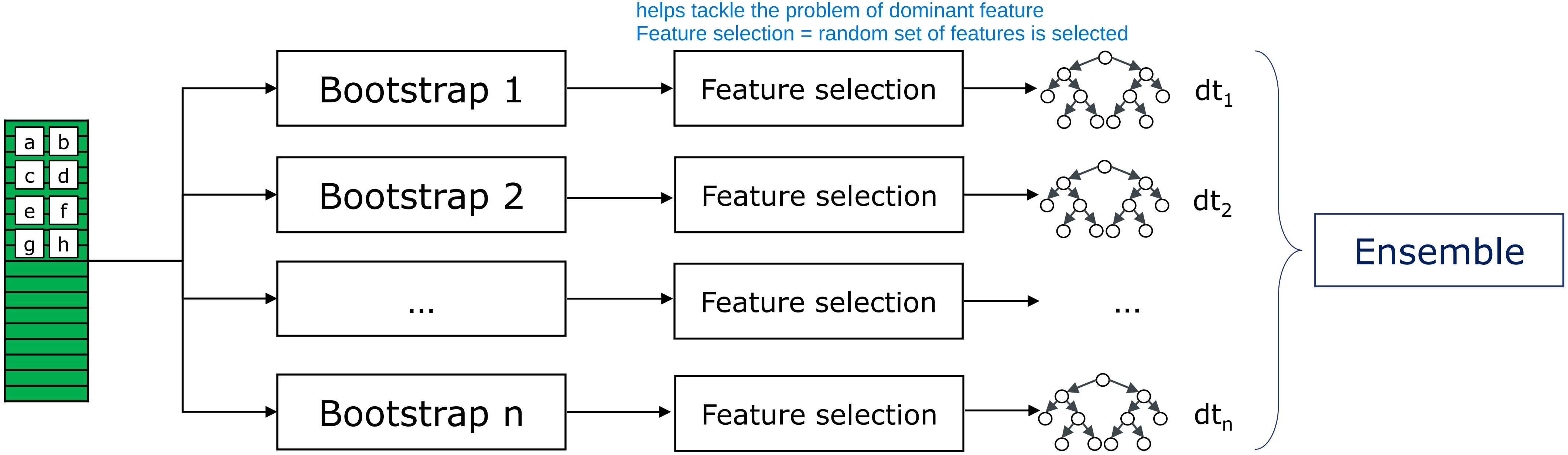
Bagging with trees

PROBLEM: If one feature is dominant the trees will always split based on that feature

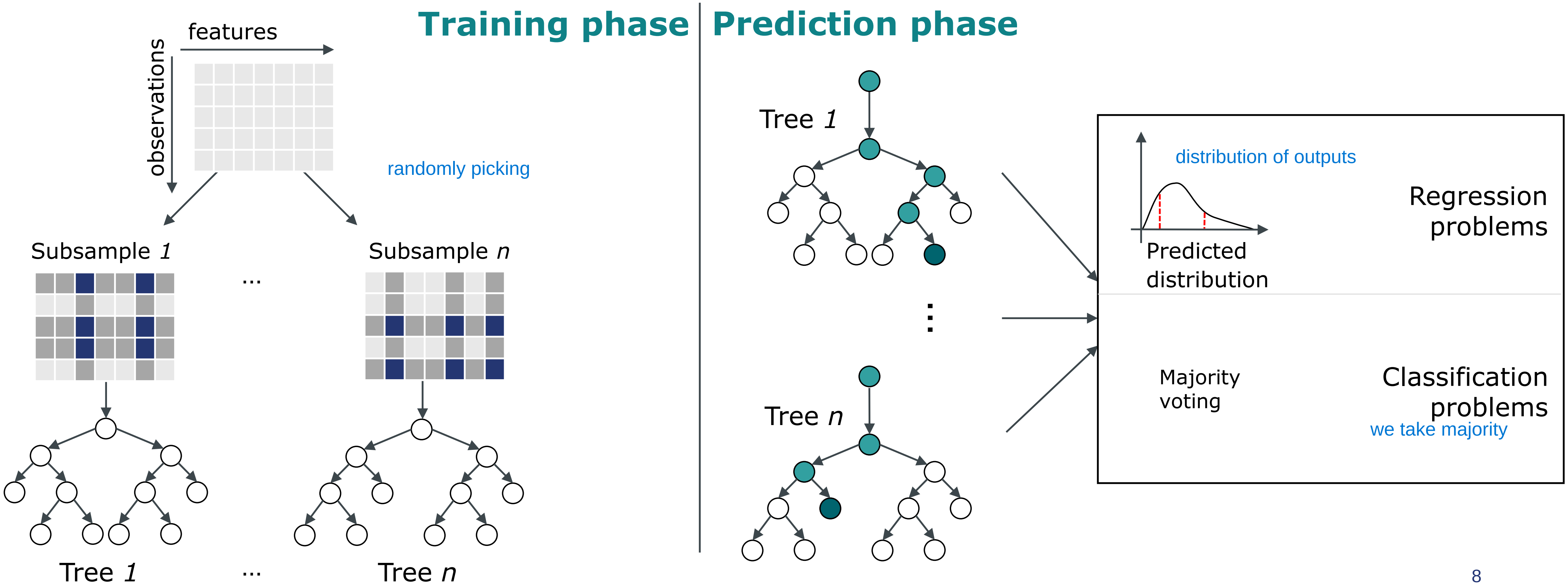


Ensemble models

Bagging + feature selection + trees = RANDOM FOREST



Random Forest – A fast glance!



Building a forest

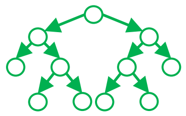
Example

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	F	55	0	66	0
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	M	42	0	74	1

Building a forest

Example

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	F	55	0	66	0
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	M	42	0	74	1



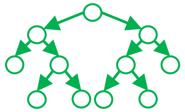
dt_1

- Subsetting
  - A percentage of samples (rows)
  - A number of features (columns)
- Create a decision tree

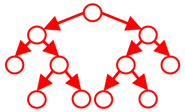
Building a forest

Example

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	F	55	0	66	0
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	M	42	0	74	1



dt_1



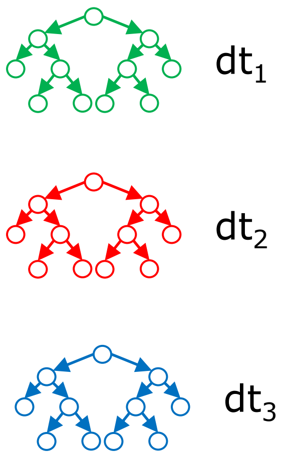
dt_2

- Subsetting
  - A percentage of samples (rows)
  - A number of features (columns)
- Create a decision tree
- Repeat a number of times

Building a forest

Example

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	F	55	0	66	0
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	M	42	0	74	1



- Subsetting
  - A percentage of samples (rows)
  - A number of features (columns)
- Create a decision tree
- Repeat a number of times

- We now have three trees, each with their own understanding of the problem, i.e., each with their own prediction

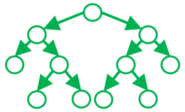
- Mean or majority voting

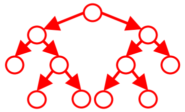
Building a forest

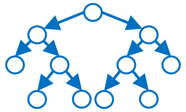
Main parameters - for R: *ranger*, for Py: *sklearn*)

sqrt of number of features

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	F	55	0	66	0
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	M	42	0	74	1


dt_1


dt_2


dt_3

R: *sample.fraction (0.63)*
Py*: max_samples (None)*

R: *mtry (sqrt)*
Py: *max_features (sqrt)*

min.node.size (1)
max.depth (unlim)
…
(see decision trees)

- Subsetti…
  - A percent…ge of sampl…
  - A number of features
- Create a decision tree
- Repeat a number of times

R: *num.trees (500)*
Python: *n_estimators (100)*

…e trees, each with their own understanding of the problem, i.e., each with their own prediction

- Mean or majority voting

FH KREMS
UNIVERSITY OF APPLIED SCIENCES / AUSTRIA

All - Used by tree.color = Out Of Bag

most of the data will than be out of the bag

Building a forest

Two more things

we use Out Of Bag samples to evaluate the model

- OOB Score

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	**F**	**55**	**0**	**66**	**0**
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	**M**	**42**	**0**	**74**	**1**

These observation not picked. Should we waste them?
No! Out-Of-Bag (OOB) samples. Can be used to evaluate the model! For free!

This is an oversimplification. What really happens is:

- For every tree, keep track of which samples were OOB.
  - **In our example**: 1 is OOB for red and blue, 2 is OOB for green, 3 is OOB for all trees, etc.
- At the end, a prediction is done by passing each sample through all trees for which they were OOB.
  - **In our example:** 1 predicted by using only red and blue, 2 predicted by using only green, 3 predicted by using the entire forest
- The OOB score is the number of correctly predicted rows from the OOB sample

Building a forest

Two more things

- Feature importance

	Gender	Age	Smoke	BMI	Heart Disease
1	M	28	0	46	0
2	M	39	1	116	1
3	**F**	**55**	**0**	**66**	**0**
4	M	18	0	46	0
5	F	87	1	88	0
6	F	58	1	135	1
7	F	77	0	65	0
8	M	60	0	116	1
9	**M**	**42**	**0**	**74**	**1**

Mean Decrease impurity (specific to Random Forest)

We created trees from subset of features!
We could see how much each feature contributed to the decrease in purity when they were selected! For free!

Feature Permutation importance (generic)

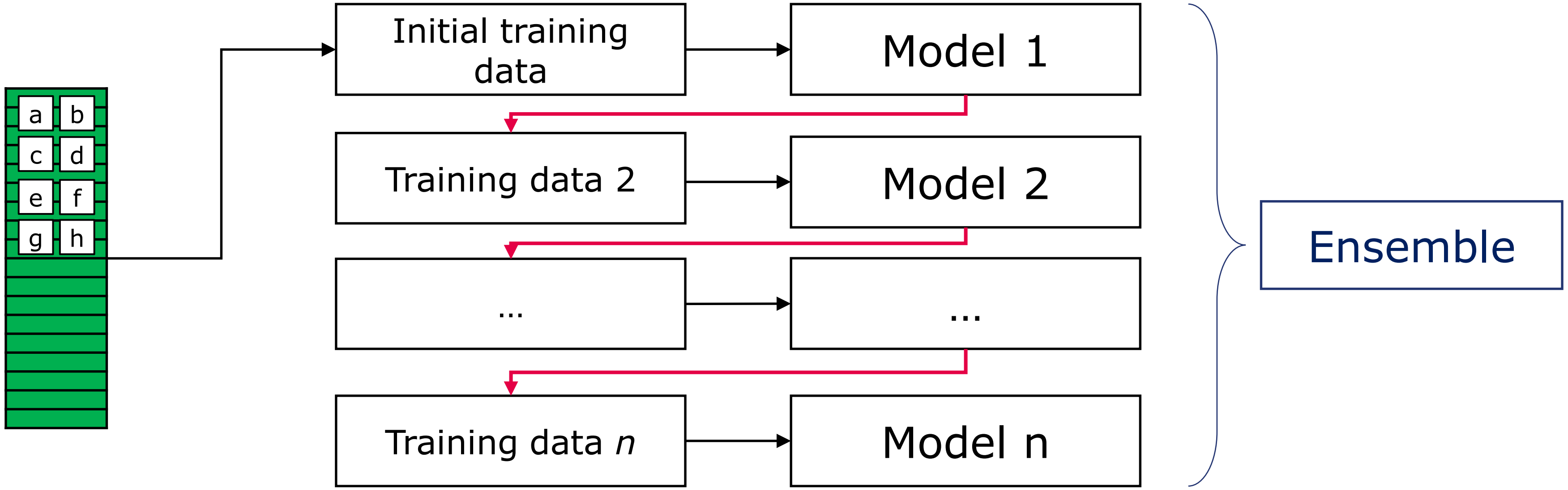
We take one feature and we randomly reshuffle it. Then we see how this reschuffling impacted the prediction error (the larger the impact the more important the feature). Not for free, but more robust.

Take one feature, reshuffle, predict, the more impact the more important the feature is

Ensemble models

Boosting

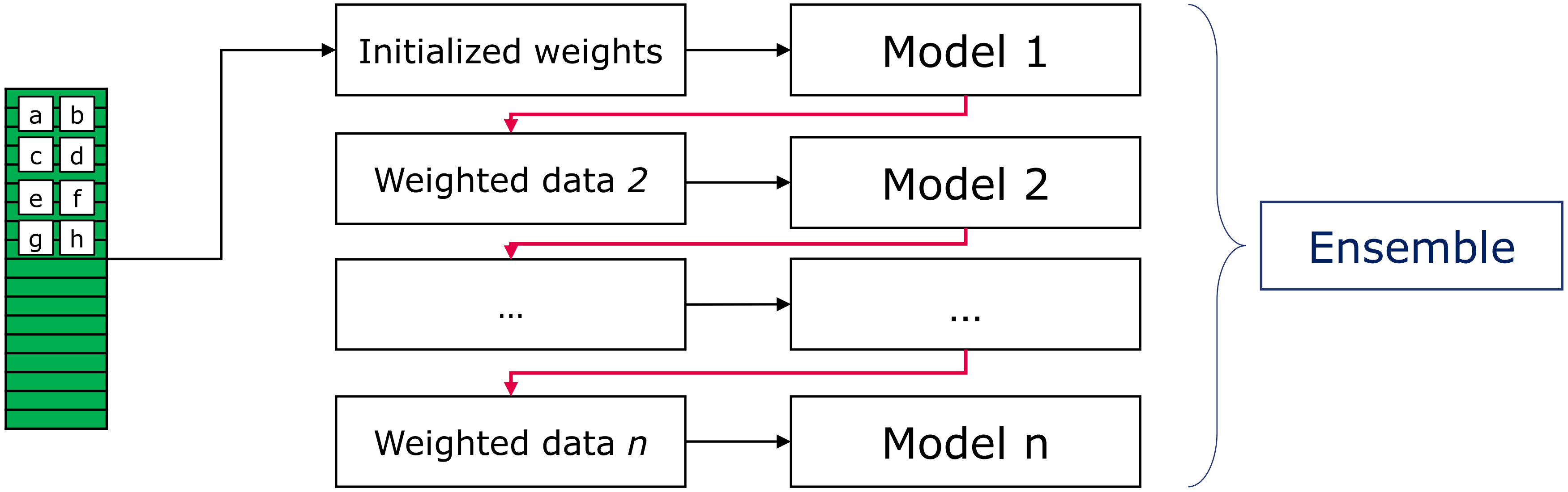
- Differently from bagging, boosting is not parallel. The models are built in sequence, each one considering the output of the previous one.



Ensemble models

Adaboost

- In AdaBoost, the output of one model is used to weight the input data of the next model

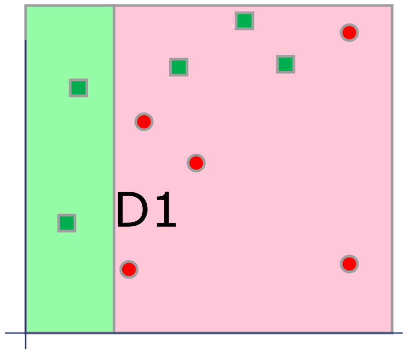


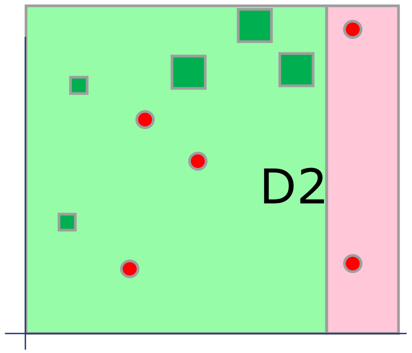
Ensemble models

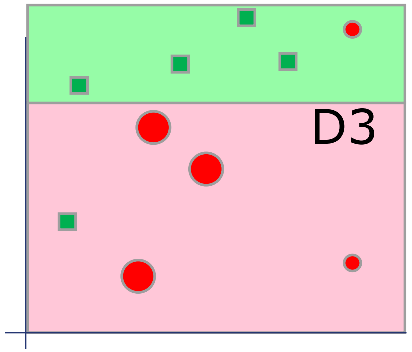
Adaboost Example

uses the previous knowledge and tries again

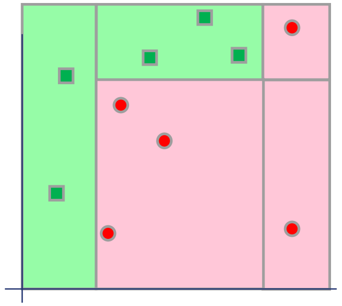
find issue so uses the knowledge and tries the safe option







tries the first 2 realizes it done correctly so it remembers and uses it in the second model



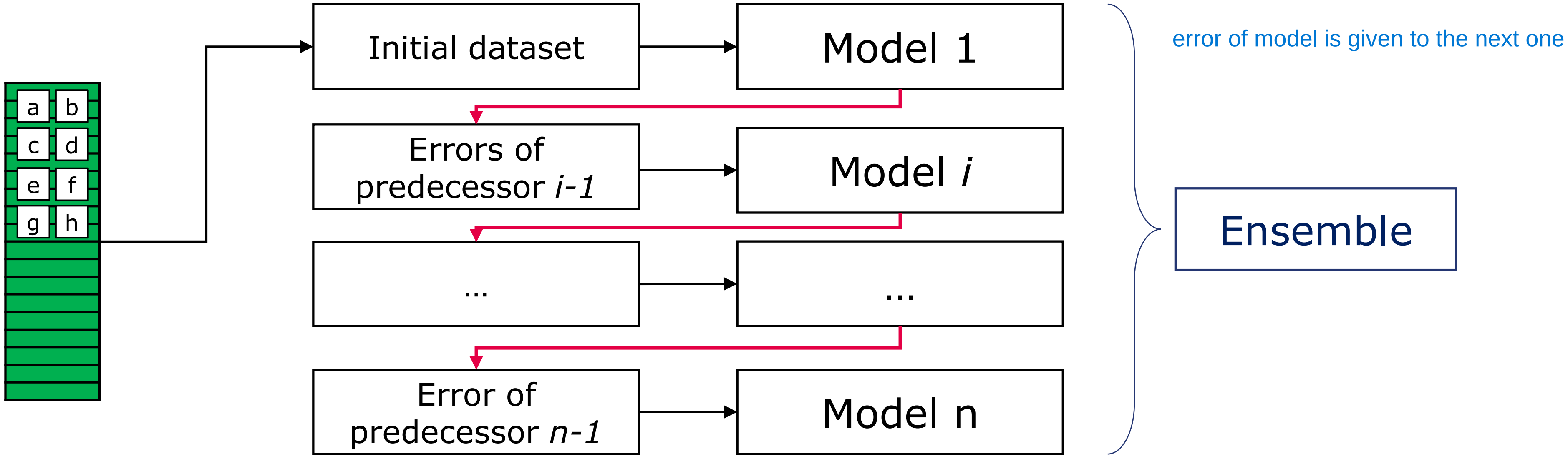
AdaBoost model final that takes the features from all the models

random forest is quicker than AdaBoost

Ensemble models

Gradient Boosting

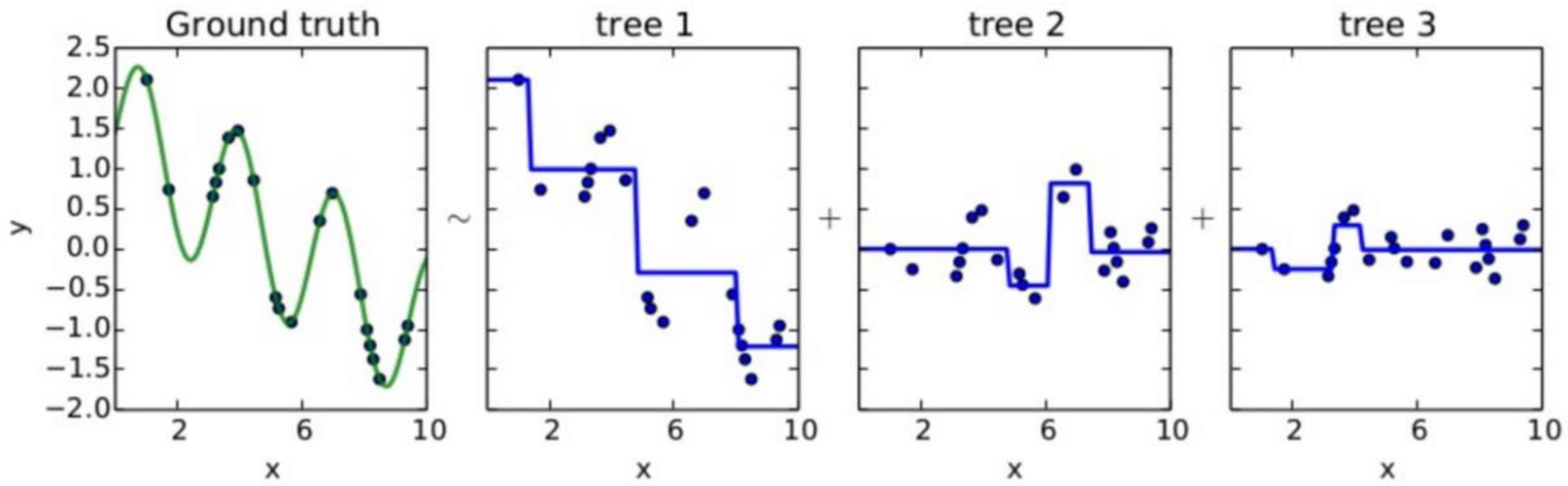
- In Gradient Boosting, each model fits the residual error of the previous model



Ensemble models

Gradient Boosting Example

random forest runs in parallel
Adaboost/XG boost works in sequence
Ada boost waits for the next
XGBoost uses the error as input for the next models



Taken from the web (source unknown)

Gradient Boosting

Example of parameters

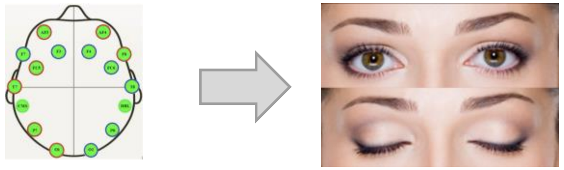
- Shrinkage:
  - **Iterations (Py: *num_round*, R: *nrounds*)**: num of boosting iterations to perform (i.e., n. of trees). The more trees the better but at higher computational costs. It is the main parameter to control model performance. Typical value: Iterations > 100.
  - **Learning rate (Py: *eta*, R: *eta*)**: Scales the contribution of each iteration. Small learning rates lead to higher number of iterations. Typical value: [0.01,…0.1,…,0.5], default = 0.3.
  - **Min_split_loss (Py: gamma, R: gamma)**: Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma, the more conservative the algorithm. Like purity gain in decision trees.
- + all the familiar ones from decision trees (maxdepth, min node size, etc.)
- Super documentation!!!
  - https://xgboost.readthedocs.io/en/latest/R-package/index.html
  - https://xgboost.readthedocs.io/en/latest/python/index.html

Ensemble models from decision trees

Summary

- Ensemble
  - Combining multiple (weak) learners
  - Random Forest:
    - Bagging + feature selection + Decision Tree
  - AdaBoost
    - Boosting (each model is trained on a dataset weighted depending on previous models) + Decision Tree
  - Gradient Boosting:
    - Boosting (each model is trained based on the residuals of the previous model) + Decision Tree
    - XGBoost is an implementation of Gradient Boosting

Exercise 2 – **Due on 04.04.2021 23.59 CET**



- Import the EEG data from the files section in Teams
  - 16 EEG numerical features
  - 1 label (eyes closed or open)
  - Train and test are already split (*eeg_training.csv*, *eeg_test.csv*)
- Create a classifier that can detect if the patient has open or closed eyelid
- Try Decision Trees, Random Forest, AdaBoost, and XGBoost
- **Attention: This is not the same data that can be found online. Do not copy!**
- Hints: Are there NAs? Is it balanced? Are there useless features? Are there outliers?

Appendix I

Strategies for preprocessing

- Missing values
  - Omit the observations containing NAs -> If there are only fews
  - Omit the features containing NAs -> If a feature contains mostly NAs
  - Impute the NAs -> Manual (e.g. median), Manual by group of similars, model-based (eg. kNN)
- Imbalanced data
  - Undersampling -> Drop samples from the majority class
  - Oversampling -> Duplicate samples from the minority class
  - Synthetic generation of new data points (e.g. *ROSE*)

Appendix II

- One hot encoding

Color
Red
Green
Blue

is_red	is_green	is_blue
1	0	0
0	1	0
0	0	1

- Dummy encoding

blue is just the rest

Color
Red
Green
Blue

is_red	is_green
1	0
0	1
0	0