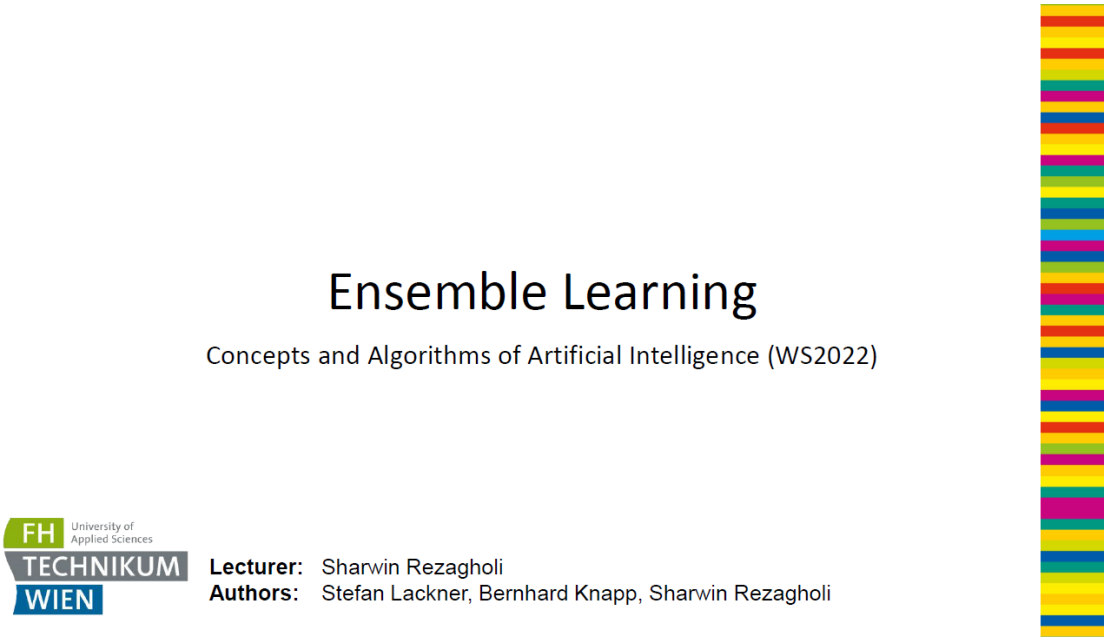
EnsembleLearning

June 21, 2025



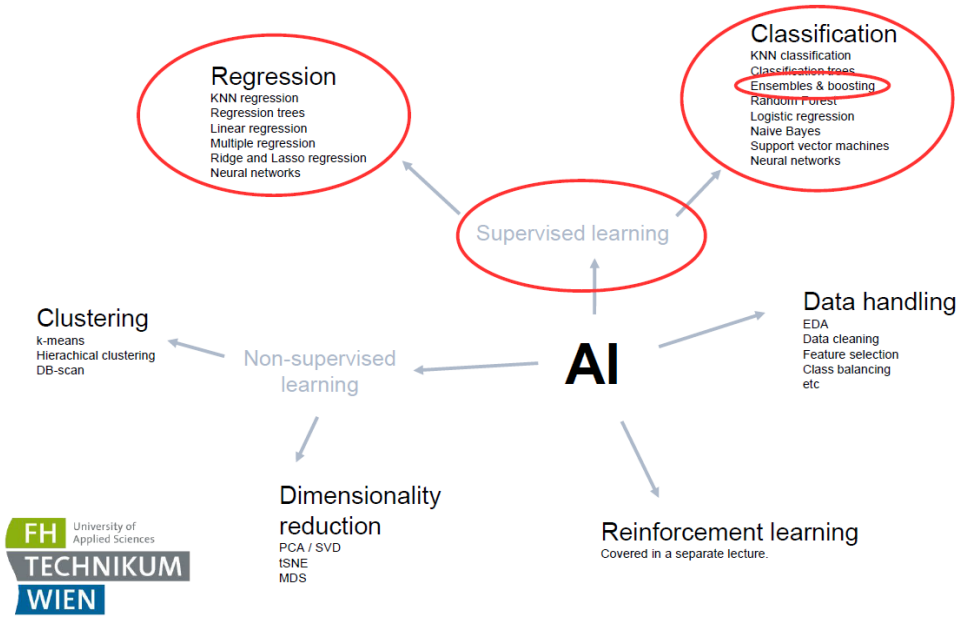
Slide 1 – Title Slide

"Good [morning/afternoon], everyone.

Today, I'll be presenting the topic of **Ensemble Learning**, as part of the course *Concepts and Algorithms of Artificial Intelligence* from the Winter Semester 2022.

This lecture was given by **Sharwin Rezagholi**, and the material was prepared by **Stefan Lackner**, **Bernhard Knapp**, and **Sharwin Rezagholi**.

Let's dive into the core ideas behind ensemble methods and see how combining multiple models can often outperform a single model."



Slide 2 – Positioning Ensemble Learning in AI

"This slide gives us a broad overview of the major fields within Artificial Intelligence.

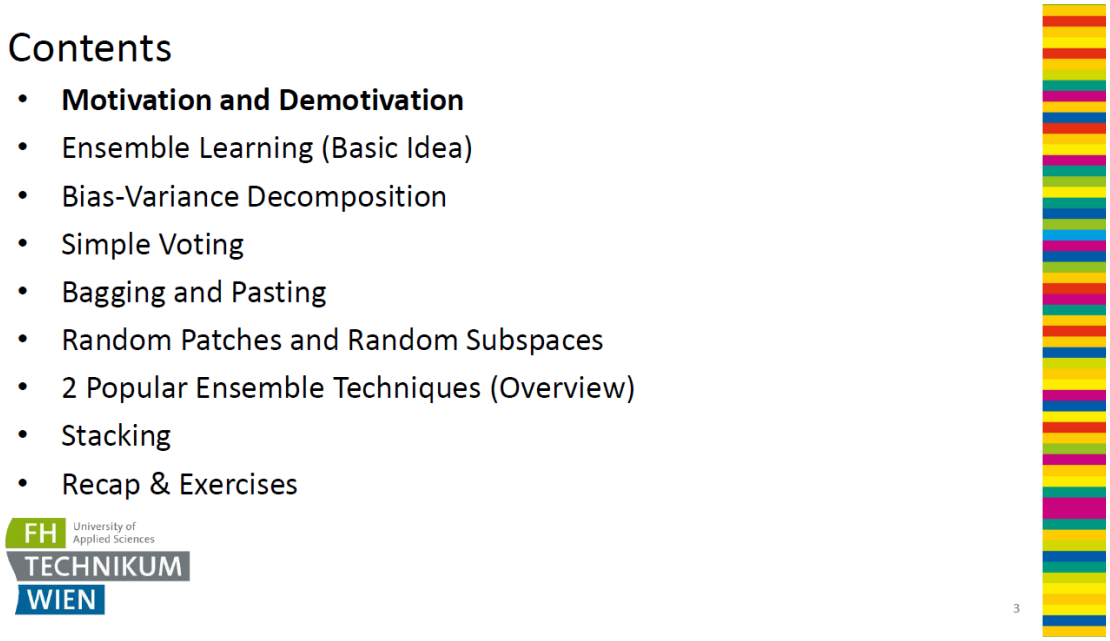
We can see that AI is divided into several branches. In the top part of the diagram, we have **Supervised Learning**, which is further split into **Regression** and **Classification** tasks. Under classification, we find our topic of interest: **Ensembles and Boosting**.

These methods belong to supervised learning because they rely on labeled data to train models.

Other important branches include:

- **Unsupervised Learning**, which covers techniques like **clustering** and **dimensionality reduction**,
- **Reinforcement Learning**, which is covered separately in this course,
- And **Data Handling**, which is a critical foundational step that includes tasks like data cleaning, feature selection, and class balancing.

This visual helps us understand where ensemble learning fits in the broader context of machine learning and AI."



Slide 3 – Contents

"Before we begin, here's a brief overview of the topics we'll be covering in this session on Ensemble Learning.

We'll start with the **Motivation and Demotivation** behind ensemble methods — why they work, but also when they might not.

Then, we'll introduce the **basic idea** of ensemble learning, followed by a look at the **bias-variance decomposition**, which is key to understanding why combining models can reduce error.

Next, we'll discuss practical ensemble techniques:

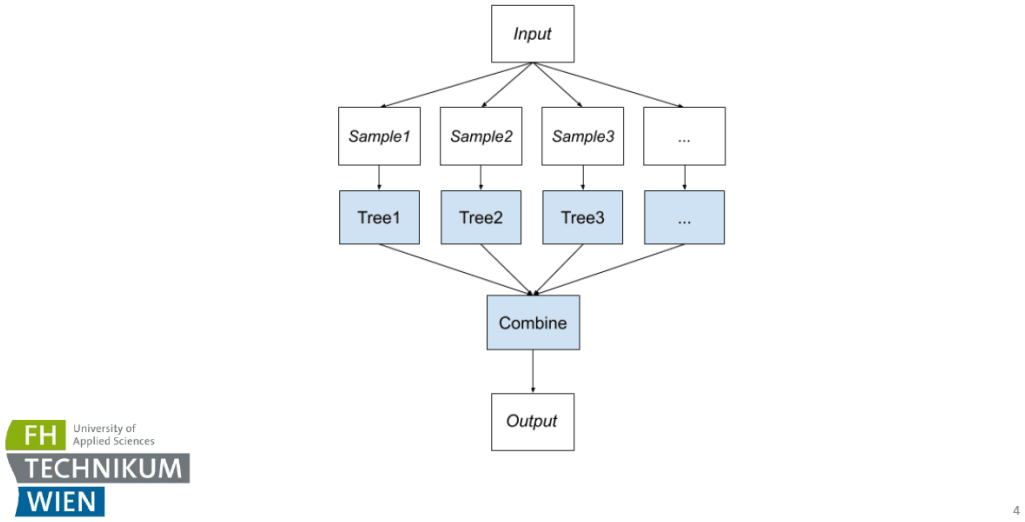
- **Simple Voting**, the most straightforward way to combine models,
- **Bagging and Pasting**, which involve training models on different subsets of the data,
- **Random Patches and Random Subspaces**, which further diversify our ensembles.

After that, we'll take a closer look at **two popular ensemble techniques**, likely Random Forest and Boosting.

We'll also cover **Stacking**, a more advanced approach where a second-level model learns how to best combine the base models.

Finally, we'll wrap up with a **Recap and Exercises** to reinforce the concepts.

Let's get started with the motivation behind ensemble learning."



Slide 4 – Ensemble Learning: The Basic Idea

"This diagram shows the core concept behind ensemble learning.

We start with a common **input dataset**, and from it, we generate multiple **samples** — these can be bootstrapped samples, random subsets, or other variations depending on the method used.

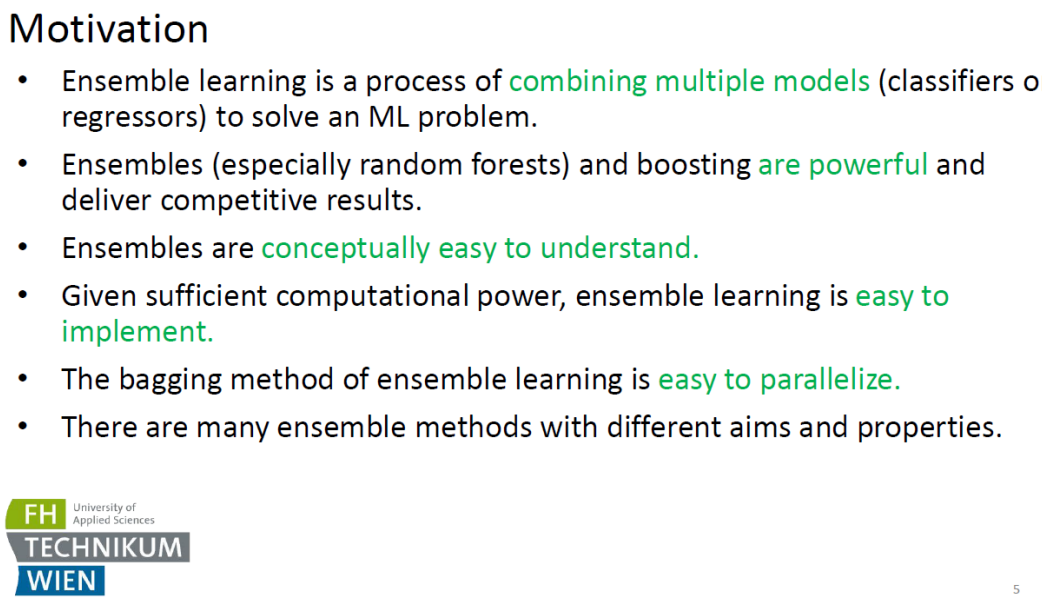
Each sample is then used to train a different model — in this case, multiple decision **trees**.

These individual models, often called **base learners**, operate independently. Each one may perform only moderately well, but they capture different patterns or aspects of the data.

Finally, we have a **combination step**, where the outputs of all these models are aggregated to produce a final **prediction**.

This combination could be done through majority voting, averaging, or even using a meta-model, as we'll see later with stacking.

The key idea is: **a group of weak models can be combined to form a strong one**."



Slide 5 – Motivation

”Why should we care about ensemble learning?

Well, first of all, **ensemble learning** is a method where we combine multiple models — classifiers or regressors — to solve a machine learning problem more effectively.

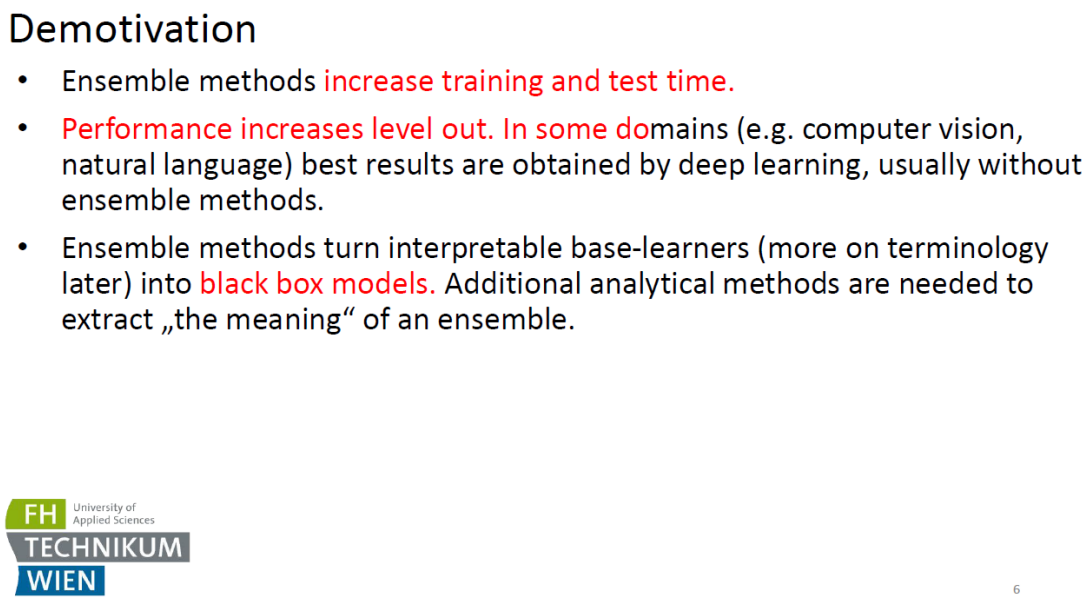
These techniques, particularly **random forests** and **boosting**, are **very powerful** and often produce **state-of-the-art performance** across many tasks.

Another big advantage is that ensembles are **conceptually simple** — even though we’re combining several models, the idea is quite intuitive: many models working together can correct each other’s weaknesses.

Also, ensemble methods are usually **easy to implement**, especially if we have sufficient computational resources.

Techniques like **bagging** can be efficiently **parallelized**, which makes them suitable for large-scale problems or distributed environments.

Finally, there’s a wide variety of ensemble methods — each with its own strengths — so we have a rich toolkit to choose from depending on the specific problem at hand.”



Slide 6 – Demotivation

”Although ensemble learning has many advantages, it also comes with some downsides.

First, **ensemble methods typically increase both training and test time**. Since we are running many models instead of just one, the computational cost is higher.

Second, while ensembles can improve performance, these improvements can **plateau**. In some domains — especially **computer vision** and **natural language processing** — state-of-the-art performance is usually achieved by deep learning models alone, without using ensembles.

And third, ensembles can turn otherwise **interpretable models into black boxes**. For example, a single decision tree is easy to visualize and understand, but an ensemble of hundreds of trees, like in a Random Forest, is not so easy to interpret. This loss of transparency can be a disadvantage in fields where explainability is crucial.

So, while ensembles are powerful, we must be aware of their limitations too.”

Contents

- Motivation and Demotivation
- **Ensemble Learning (Basic Idea)**
- Bias-Variance Decomposition
- Simple Voting
- Bagging and Pasting
- Random Patches and Random Subspaces
- 2 Popular Ensemble Techniques (Overview)
- Stacking
- Recap & Exercises

FH University of Applied Sciences TECHNIKUM WIEN

7

Slide 7 – Contents (Checkpoint)

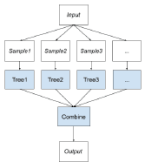
"Here's a quick checkpoint to show where we are in the presentation.

So far, we've covered the **motivation and demotivation** behind ensemble learning, and we've introduced its **basic idea**.

Now, we're moving on to the next core concept: **bias-variance decomposition**, which is fundamental to understanding why ensemble methods often lead to better generalization."

Ensemble Learning

- The **combination of different models** to obtain a final result.
- "Different models" may refer to
  - **different algorithms** (e.g. trees, SVMs, ...),
  - **same algorithm trained differently** (e.g. trees trained on different subsets of the data, or on differently weighted data).
- The **output of all models** is used to obtain a final result.
- Single models are called **base-learners**.
- **Applicable to many problems**: Regression, classification, and clustering (consensus-clustering).



FH University of Applied Sciences TECHNIKUM WIEN

8

Slide 8 – Ensemble Learning

”Let’s take a closer look at what ensemble learning actually is.

At its core, ensemble learning is about the **combination of different models** to produce a final prediction.

Now, what do we mean by *different models*? This can refer to:

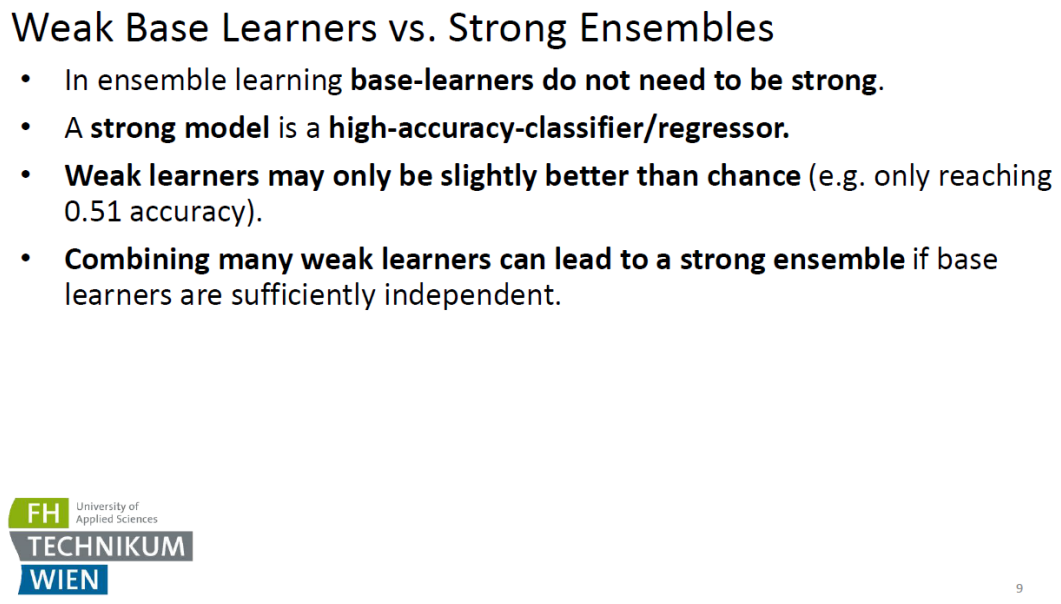
- **Different algorithms**, like combining decision trees, support vector machines, and neural networks.
- Or using the **same algorithm trained differently** — for example, multiple decision trees trained on different data subsets or with different weightings.

The **output of all the models**, or *base learners*, is then combined — through voting, averaging, or other strategies — to reach a final decision.

These base learners can be weak individually, but together they often achieve significantly better performance.

What’s powerful about ensemble learning is that it’s **applicable to many types of problems**: regression, classification, and even clustering — such as in consensus clustering where multiple clustering results are merged.

As shown in the diagram, we take various samples, train multiple models, and combine them to get our final output.”



Slide 9 – Weak Base Learners vs. Strong Ensembles

”One of the most fascinating insights in ensemble learning is that **base learners do not need to be strong**.

A **strong model** is one that has high accuracy — something we usually aim for in machine learning. But ensemble learning flips this idea on its head.

Even **weak learners** — models that perform only slightly better than random guessing, for example around **51% accuracy** — can still be useful.

The trick is in combining them. If these weak models are **independent enough** and make different kinds of errors, then **their combination can become a strong ensemble**.

This is the foundation of methods like **boosting**, where many weak learners, such as shallow decision trees, are combined to create a high-performance predictor.”

Slide 10 – Many Weak Learners Together Can Do a Great Job

”This cartoon is a great analogy for ensemble learning.

Each of the blindfolded individuals is examining a different part of the elephant and coming to their own conclusion:

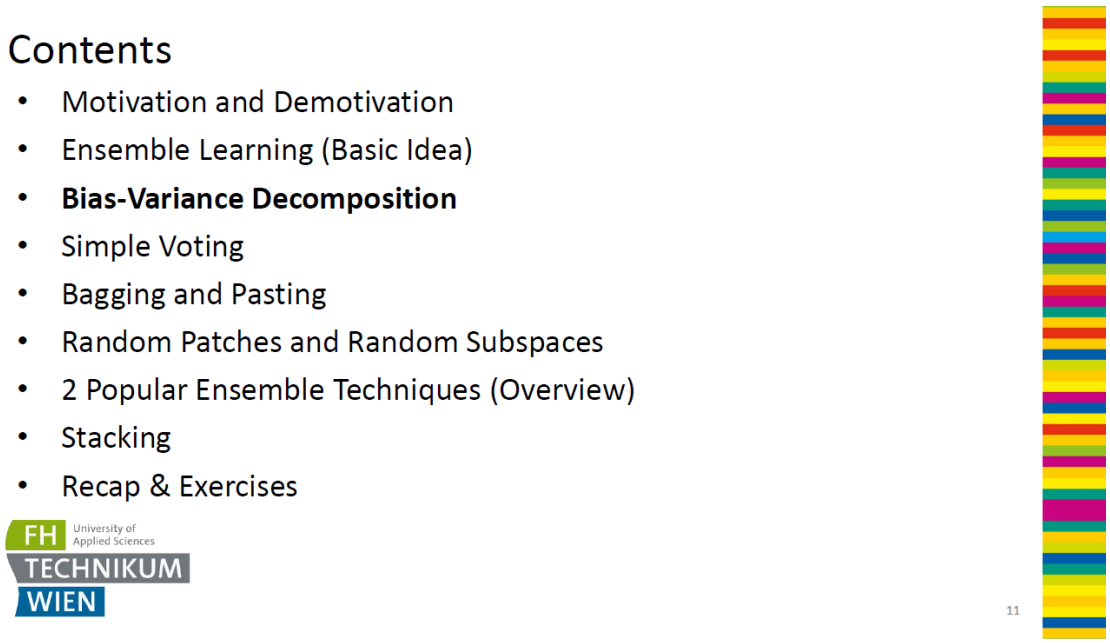
- One thinks it’s a **snake**,
- Another says it’s a **tree stump**,
- One says it feels like **leather**,
- And another describes it as a **furry mouse**.

Individually, they are all **partially correct**, but also **incomplete** and **misleading**.

However, if we were to **combine their observations**, we’d get a much more accurate understanding of the full picture — just like in ensemble learning, where combining multiple weak learners

leads to a strong model.

This is the key idea: **the sum of weak learners can outperform even a single strong learner**, especially when they contribute different perspectives."



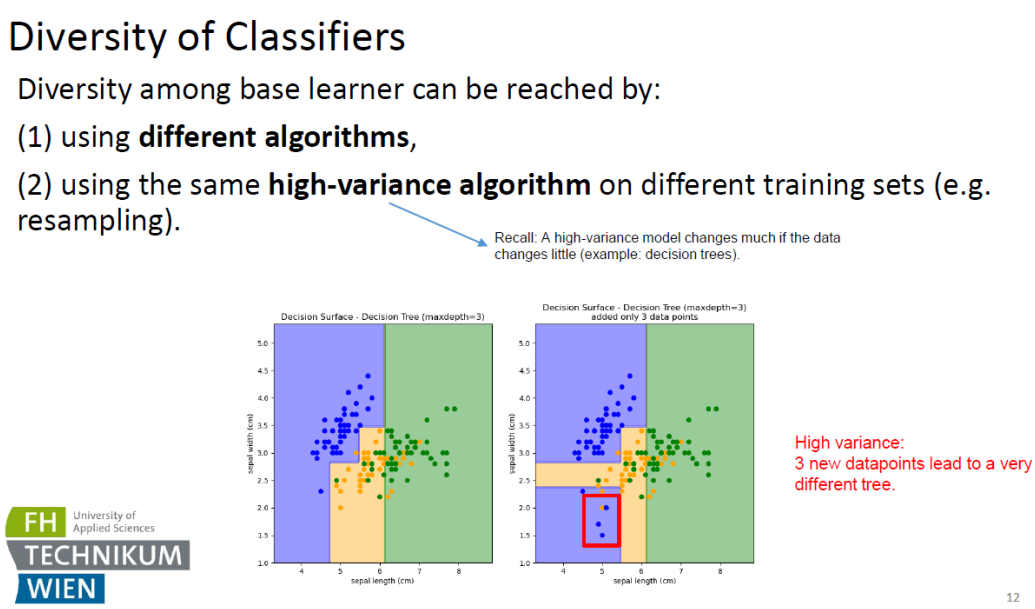
Slide 11 – Transition: Bias-Variance Decomposition

"Let's take another quick look at the contents to see where we are.

We've already covered the motivation behind ensemble learning and introduced the basic concepts.

Now we move on to a crucial theoretical foundation — **Bias-Variance Decomposition** — which helps us understand **why ensembles can reduce error** and improve generalization performance.

Let's dive into that next."



Slide 12 – Diversity of Classifiers

"A critical ingredient in ensemble learning is **diversity** among the base learners.

We can achieve this diversity in two main ways:

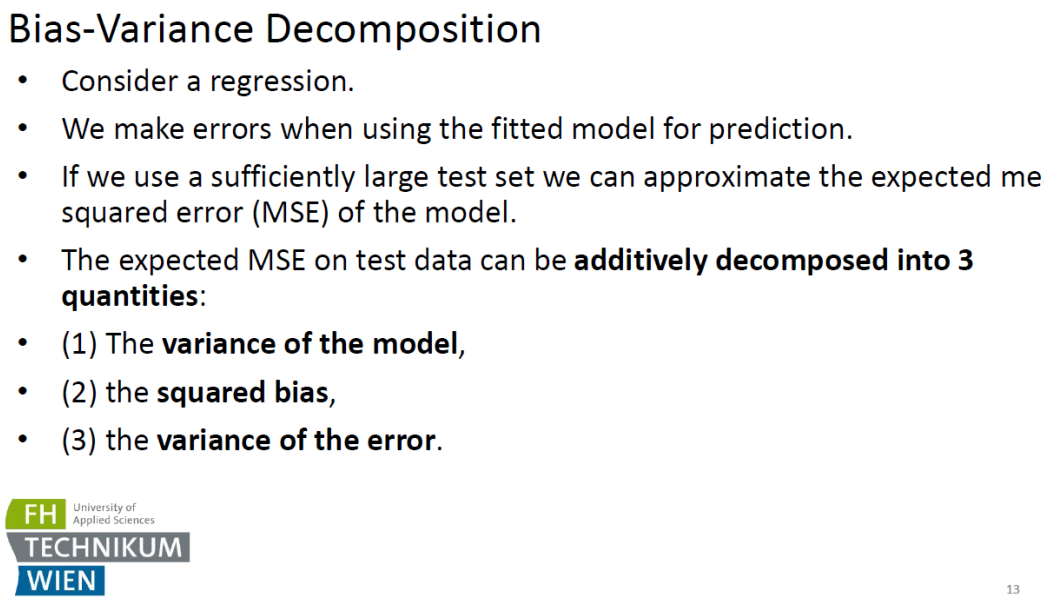
1. By using **different algorithms**, such as combining decision trees, support vector machines, and k-nearest neighbors.
2. Or by using the **same high-variance algorithm**, like decision trees, but trained on different subsets of the data — for example, through **resampling** techniques.

As shown in the two plots below:

- Both use decision trees with a maximum depth of 3.
- But in the right-hand plot, we've added just **3 new data points**.
- That tiny change causes the decision boundaries to shift dramatically.

This is a hallmark of a **high-variance model** — it's very sensitive to the training data.

And this sensitivity is actually beneficial in ensembles, because it helps generate a wide variety of base learners, which improves the overall performance when we combine them."



Slide 13 – Bias-Variance Decomposition

"Let's now talk about **Bias-Variance Decomposition**, which is central to understanding model errors — and how ensemble methods help reduce them.

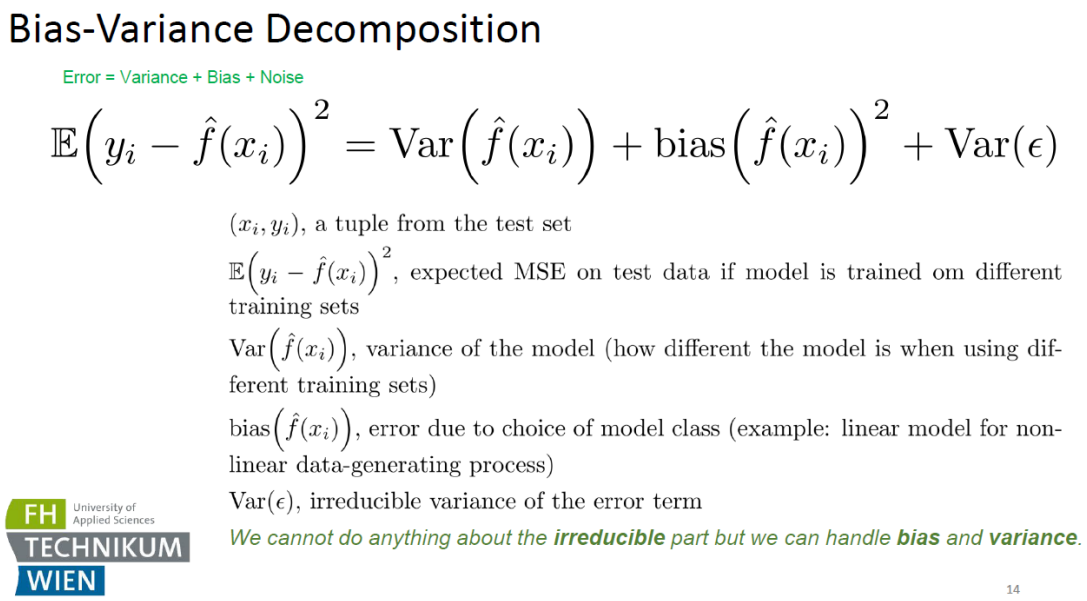
Let's consider a **regression problem**. Whenever we make predictions using a trained model, we naturally make **errors**.

If we use a sufficiently large test set, we can approximate the **expected mean squared error** (MSE) of our model — that's the average squared difference between predicted and actual values.

Importantly, this expected MSE can be **decomposed into three parts**:

1. The **variance of the model** — this reflects how sensitive the model is to variations in the training data. High variance means the model may overfit.
2. The **squared bias** — this measures how far off the model's predictions are from the true values on average. High bias indicates underfitting.
3. And finally, the **variance of the error**, which is essentially the noise inherent in the data — this part cannot be reduced by any model.

Ensemble methods help by **reducing variance**, without necessarily increasing bias — a sweet spot we'll explore more shortly."



Slide 14 – Bias-Variance Decomposition (Formula)

"This slide formalizes what we discussed conceptually in the last one.

We're looking at the **expected squared error** between the true output y_i and the prediction $\hat{f}(x_i)$. This is the **mean squared error**, and it breaks down into three parts:

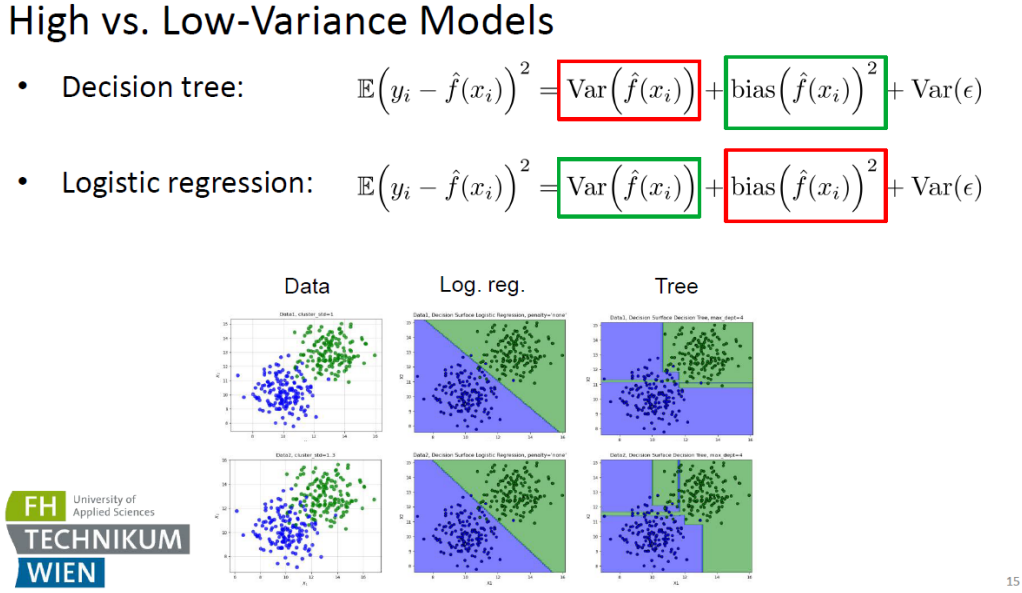
$$\mathbb{E}\left(y_i - \hat{f}(x_i)\right)^2 = \mathrm{Var}\left(\hat{f}(x_i)\right) + \mathrm{bias}^2\left(\hat{f}(x_i)\right) + \mathrm{Var}(\epsilon)$$

Let's unpack this:

- $\mathrm{Var}\left(\hat{f}(x_i)\right)$: This is the **variance** of our model. It shows how much the prediction would change if we trained on different datasets.
- $\mathrm{bias}^2\left(\hat{f}(x_i)\right)$: This is the **squared bias**. It reflects the error from wrong assumptions — for example, using a linear model when the real relationship is nonlinear.
- $\mathrm{Var}(\epsilon)$: This is the **irreducible error** — the noise inherent in the data that no model can fix.

So in practice, we **cannot eliminate the noise**, but we can try to **reduce bias and variance** — and that's where ensemble methods shine.

For example, **bagging** is especially effective at reducing variance, while **boosting** can help reduce both bias and variance."



Slide 15 – High vs. Low-Variance Models

"This slide illustrates how different models contribute differently to the bias-variance tradeoff.

Let's compare two models:

- A **decision tree**,
- And **logistic regression**.

Both models' performance can be decomposed using the same formula:

$$\mathbb{E}(y_i - \hat{f}(x_i))^2 = \mathrm{Var}(\hat{f}(x_i)) + \mathrm{bias}^2(\hat{f}(x_i)) + \mathrm{Var}(\epsilon)$$

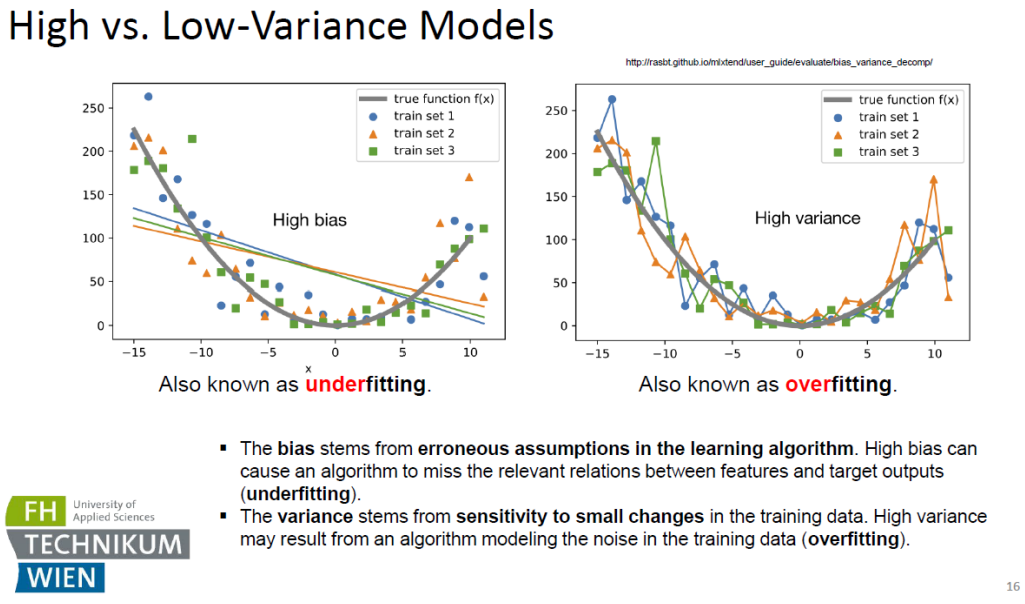
For **decision trees**, the **variance is high** — that's marked in red. The model is very flexible and changes a lot with different datasets, which can lead to overfitting. However, the **bias is low**, because it can fit complex patterns.

In contrast, **logistic regression** has **low variance** — marked in green — meaning it's stable across datasets. But it comes with **higher bias**, because it can only model linear boundaries, and may underfit if the true relationship is nonlinear.

The images at the bottom illustrate this:

- The **Logistic Regression** decision boundaries are linear and consistent.
- The **Decision Trees** show more flexible but also more dataset-sensitive boundaries.

The key insight is that **ensemble methods can help reduce variance** — making high-variance models like decision trees more robust by combining them, as in Random Forest."



Slide 16 – High vs. Low-Variance Models

"This slide gives us a **visual comparison** between high-bias and high-variance models — or in other words, **underfitting** versus **overfitting**.

On the **left**, we see a **high-bias model**:

- It fits a straight line through data that clearly follows a curve.
- It **misses the true relationship**, and all models — regardless of training data — give similar, poor fits.
- This is classic **underfitting**, caused by **overly simplistic assumptions** in the model.

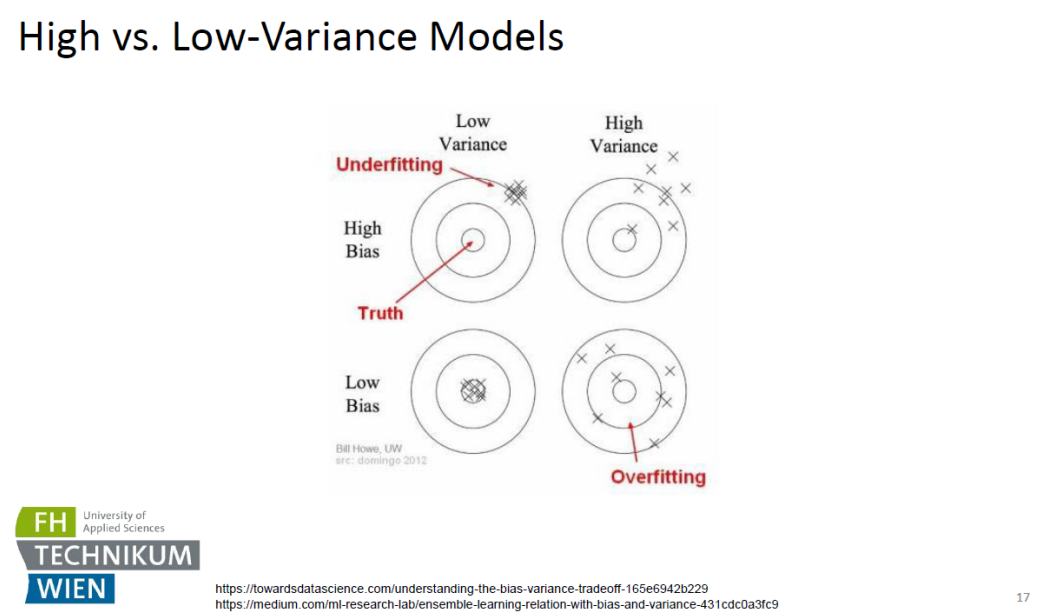
On the **right**, we have a **high-variance model**:

- Here, each training set leads to a completely different, wiggly curve.
- The model captures **noise instead of the true pattern**.
- This is **overfitting**, where the model is too flexible and too sensitive to small changes in the data.

To summarize:

- **Bias** is caused by **wrong assumptions** — like fitting linear models to nonlinear data.
- **Variance** is caused by **too much sensitivity to data** — the model overreacts to minor changes in the training set.

Ensemble methods — especially **bagging** — help reduce variance by averaging out this kind of instability."



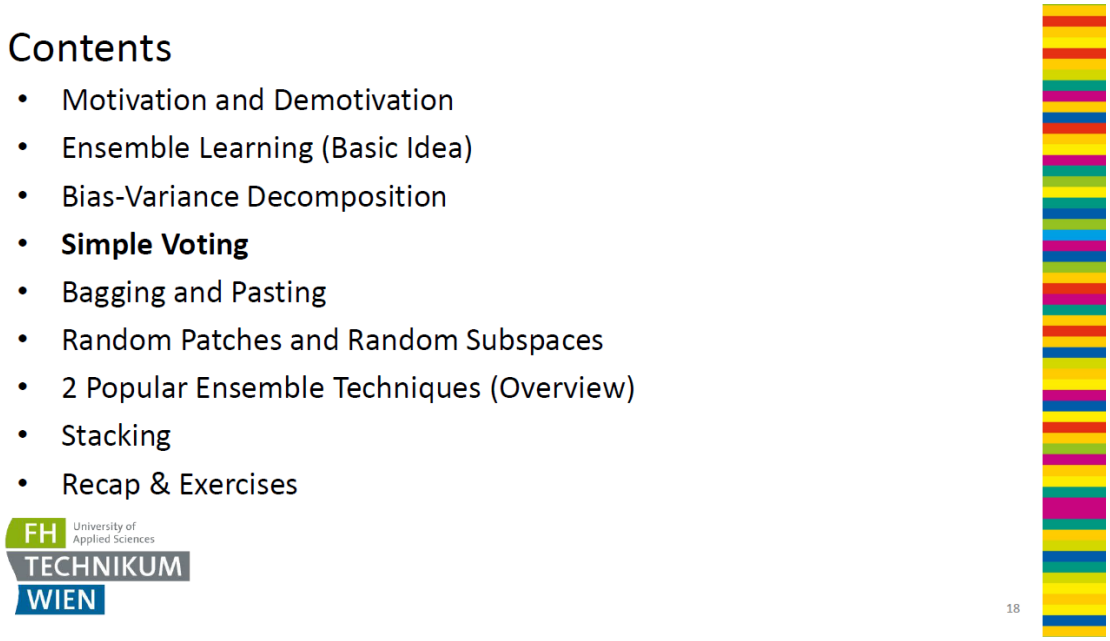
Slide 17 – High vs. Low-Variance Models (Target Analogy)

"This diagram uses a **target board** analogy to help us intuitively understand the bias-variance tradeoff.

Each cross represents a model prediction, and the **center of the target is the true value** we're trying to predict.

- In the **top-left**, we see **high bias, low variance**: all the predictions are tightly clustered, but far from the target. This is typical **underfitting** — the model is too simple to capture the pattern, and always makes the same mistake.
- In the **top-right**, we have **high bias and high variance** — predictions are both **inaccurate and inconsistent**, showing poor learning and over-sensitivity to the data. This is the worst case.
- The **bottom-left** shows the ideal case: **low bias, low variance** — accurate and consistent predictions close to the truth.
- Finally, the **bottom-right** shows **low bias but high variance** — predictions are centered around the truth but scattered. This is typical **overfitting**, where the model learns the training data too well but doesn't generalize.

The key takeaway is that **good models aim to balance bias and variance** — and ensemble methods help us get closer to that bottom-left quadrant."



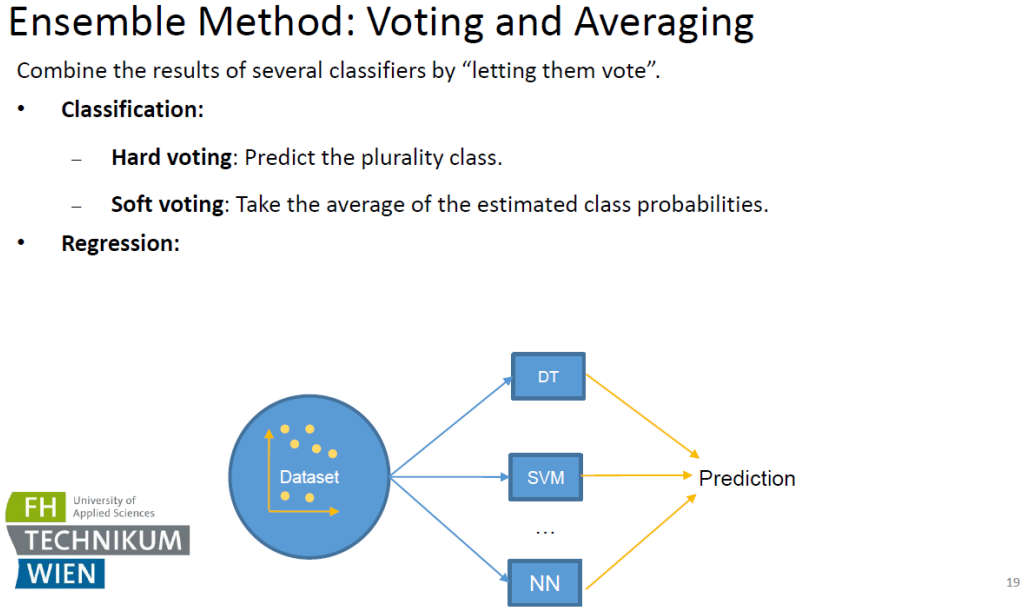
Slide 18 – Contents (Checkpoint: Simple Voting)

"Let's take another quick look at our progress through the lecture.

We've just finished exploring the **bias-variance decomposition** and how it relates to different model behaviors — underfitting, overfitting, and generalization.

Now, we move into more practical territory, starting with one of the simplest ensemble strategies: **Simple Voting**.

This method is intuitive but surprisingly powerful. Let's take a closer look."



Slide 19 – Ensemble Method: Voting and Averaging

"One of the simplest ensemble techniques is called **voting** — where we let multiple classifiers make predictions and then **combine their results**.

Let's break it down:

For **classification**, we have two main types:

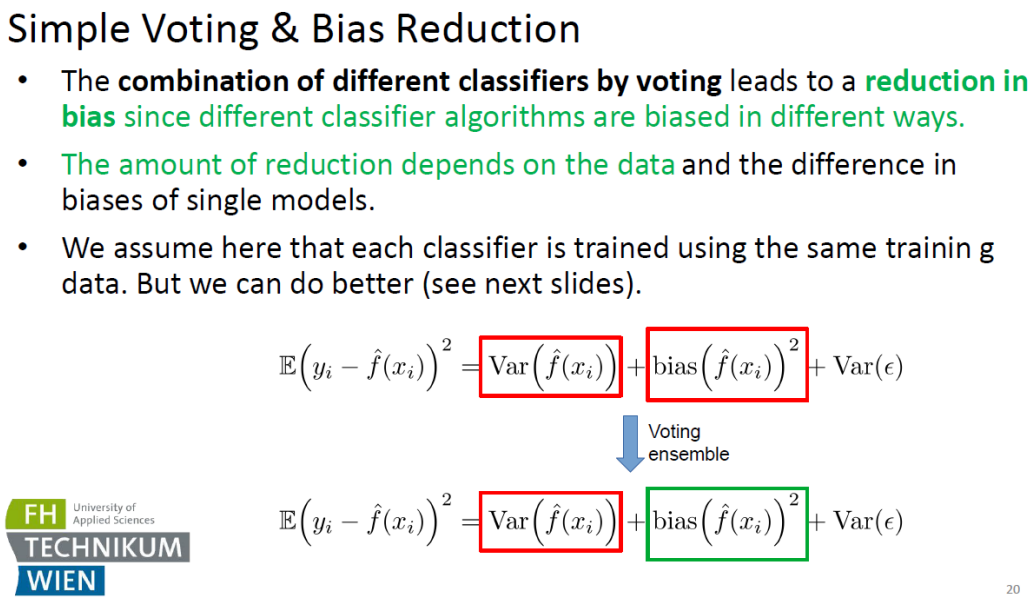
- **Hard voting**: Each classifier makes a prediction — like a class label — and the final output is the class with the most votes. This is like a majority vote.
- **Soft voting**: Instead of choosing directly, each model outputs a probability for each class. We **average these probabilities** and choose the class with the highest average. This usually works better, especially if the classifiers are well-calibrated.

For **regression**, which isn't shown in detail here, we typically just **average the predictions** of all the models.

The diagram at the bottom illustrates this setup:

- We start with a common **dataset**.
- It is passed to multiple different models — for example, a **decision tree (DT)**, a **support vector machine (SVM)**, and a **neural network (NN)**.
- Their outputs are combined to make a final prediction.

Voting and averaging are especially useful when we want to combine diverse models that each bring different strengths."



Slide 20 – Simple Voting & Bias Reduction

"Now let's look at how **simple voting can help reduce bias**.

When we **combine different classifiers** — for example, a decision tree, an SVM, and a neural network — each one may be biased in a different way. But when we vote across them, these biases can **cancel out**, leading to a **lower overall bias**.

That's what the first point is highlighting: combining classifiers by voting leads to a **reduction in bias**, especially if the models are diverse.

The **amount of bias reduction** depends on two things:

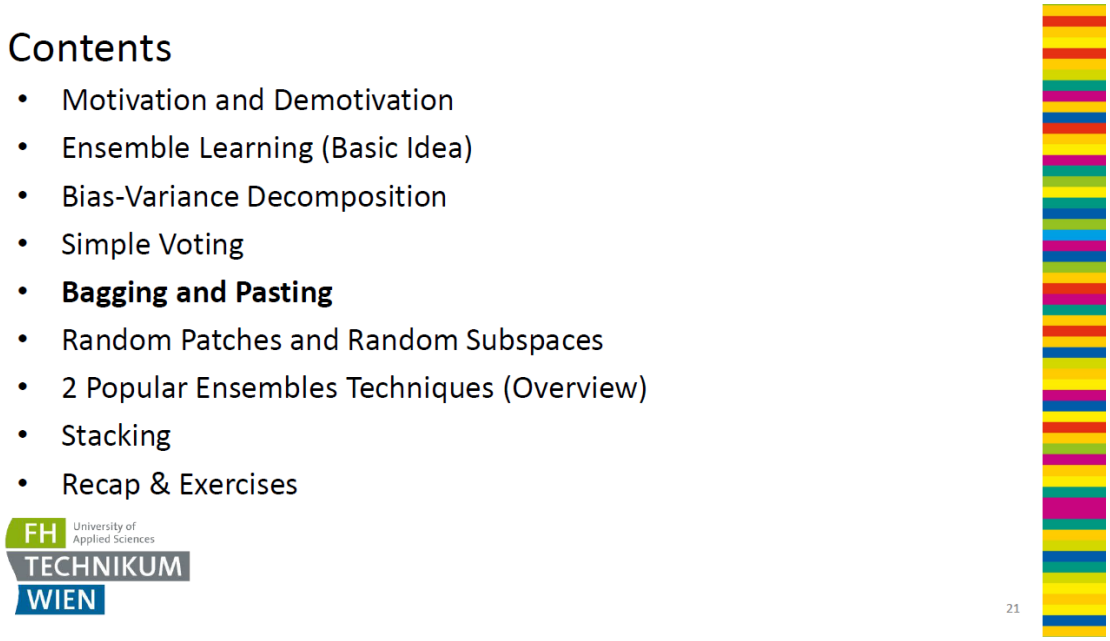
- The **data itself**, and
- How **differently biased** the individual models are.

In the lower part of the slide, we see the familiar bias-variance decomposition:

- The top formula shows the bias and variance **before voting**.
- After applying a **voting ensemble**, we typically see a **reduction in bias** — as shown by the green box — while variance remains the same or may even slightly increase.

The key idea is this:

- Voting ensembles help to **balance out model-specific weaknesses**, particularly their bias.
- And in upcoming slides, we'll see how to do even better — for instance, by training models on different data subsets."



Slide 21 – Contents (Checkpoint: Bagging and Pasting)

”Let’s take another brief look at our agenda.

We’ve just seen how **simple voting** helps reduce bias by combining the predictions of diverse models.

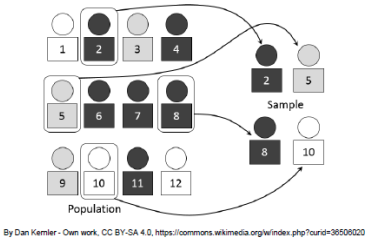
Now, we’re moving on to a more advanced ensemble technique: **Bagging and Pasting**.

Both methods build multiple versions of a model by training on different subsets of the data — and they’re particularly effective at **reducing variance**.

Let’s explore how these techniques work and how they differ.”

Bagging and Pasting

- We do not use all the training data at once.
- **Bagging** and **Pasting** are **resampling techniques:**
  - **Bagging** (**b**ootstrap **agg**regat**ing**): Repeatedly draw random samples from a training set **with replacement**.
  - **Pasting**: Draw random samples **without replacement** (each observation in the sample can be used once). Requires a sufficiently large dataset.
- Resampling techniques are used to **reduce variance.**



By Dan Kernler - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=36506020

22

Slide 22 – Bagging and Pasting

"Now we come to two important ensemble techniques: **Bagging** and **Pasting**.

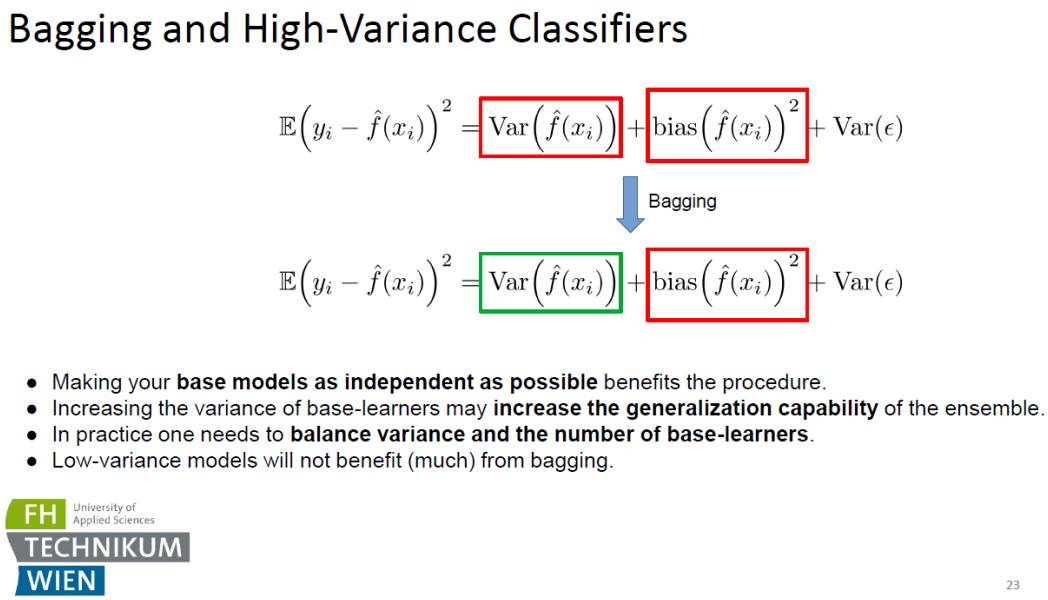
The main idea is that we don't use all the training data at once. Instead, we create **multiple subsets** of the data and train a separate model on each one.

These are both **resampling techniques**:

- **Bagging**, short for *bootstrap aggregating*, draws random samples **with replacement** from the training set. This means some data points might appear multiple times in a single sample.
- **Pasting** draws random samples **without replacement** — each data point appears only once in each subset. This requires a large enough dataset so the subsets are still informative.

The purpose of both methods is to **reduce variance**. By training many models on slightly different datasets, we can smooth out the noise that any one model might overfit to.

The diagram at the bottom illustrates this: from a larger population, we sample smaller subsets, which are then used to train different models in the ensemble."



Slide 23 – Bagging and High-Variance Classifiers

"Now let's see how **bagging** impacts the bias-variance tradeoff — especially when using **high-variance base learners**.

At the top, we have the standard error decomposition:

- The model has **high variance** and potentially some bias.

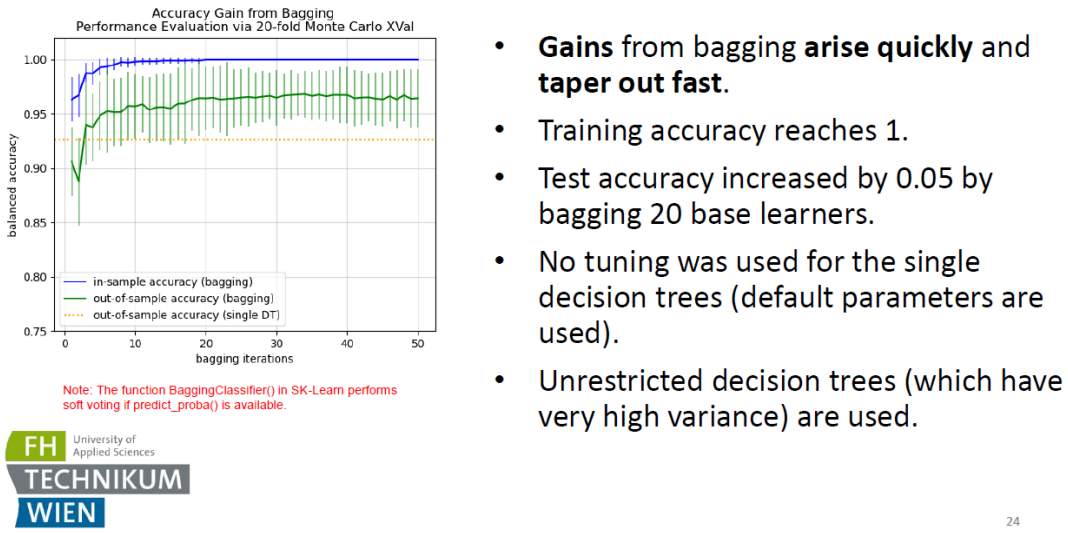
When we apply **bagging**, as shown in the second equation:

- The **variance is reduced** — highlighted in green — because averaging over many diverse models smooths out the fluctuations.
- The **bias generally stays the same** — or may even slightly increase — because we're using the same base algorithm.

Now look at the bullet points:

- It's important that your base models are **as independent as possible** — the more uncorrelated their errors, the better the averaging works.
- Interestingly, even **increasing the variance** of individual base learners may help if it boosts their **diversity** — which is beneficial for ensemble performance.
- But of course, we need to **balance** the variance of base learners with the **number of models** used. Too much noise can't always be canceled out.
- And finally, **low-variance models like linear regression** don't benefit much from bagging — there's just not enough variance to reduce.

This is why **decision trees**, which are naturally high-variance, are such a perfect match for bagging techniques like **Random Forest**."



Slide 24 – Increased Generalization Capability due to Bagging

"This slide shows the practical effect of bagging on model performance — especially how it boosts **generalization**.

Let's break down the chart:

- On the **x-axis**, we have the number of bagging iterations — how many models were used in the ensemble.
- On the **y-axis**, we see **balanced accuracy**, both for training and testing.

The three lines represent:

- **Blue**: In-sample (training) accuracy — which quickly reaches 1.
- **Green**: Out-of-sample (test) accuracy — which steadily improves and then **stabilizes**.
- **Orange dotted**: Test accuracy of a single decision tree — lower and not improving.

The key takeaway:

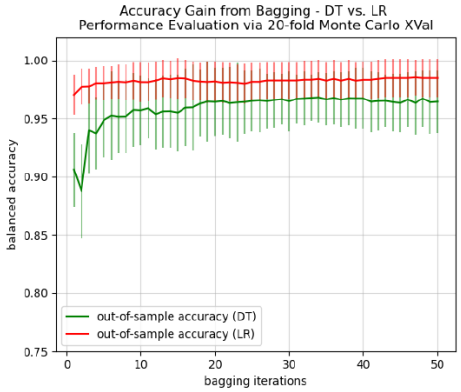
- **Bagging significantly improves test accuracy**, but the gains **taper off** quickly — most of the benefit comes from the **first 10 to 20 models**.
- In this experiment, **20 decision trees** improved accuracy by around **5 percentage points** over a single tree.

Also important to note:

- No hyperparameter tuning was done — just **default decision trees** were used.
- These trees were **unrestricted**, meaning they had **high variance** — ideal for bagging to reduce that variance through averaging.

So: bagging is **simple, powerful**, and works best with **high-variance base models**."

Bagging: Decision Trees vs. Logistic Regressions



- Log. reg. performs better than bagged decision trees on this data.
- **Log. reg. is a low variance model**, hence it does not gain much from bagging.
- However: Be careful and try different things on your own datasets.

FH University of Applied Sciences TECHNIKUM WIEN

*James, Witten, Hastie, Tibshirani (2017), p. 37.

25

Slide 25 – Bagging: Decision Trees vs. Logistic Regressions

"This slide compares the performance of bagging when applied to **decision trees** versus **logistic regression**.

Let's interpret the graph:

- The **x-axis** shows the number of bagging iterations — how many models were used.
- The **y-axis** is again **balanced accuracy** on test data.

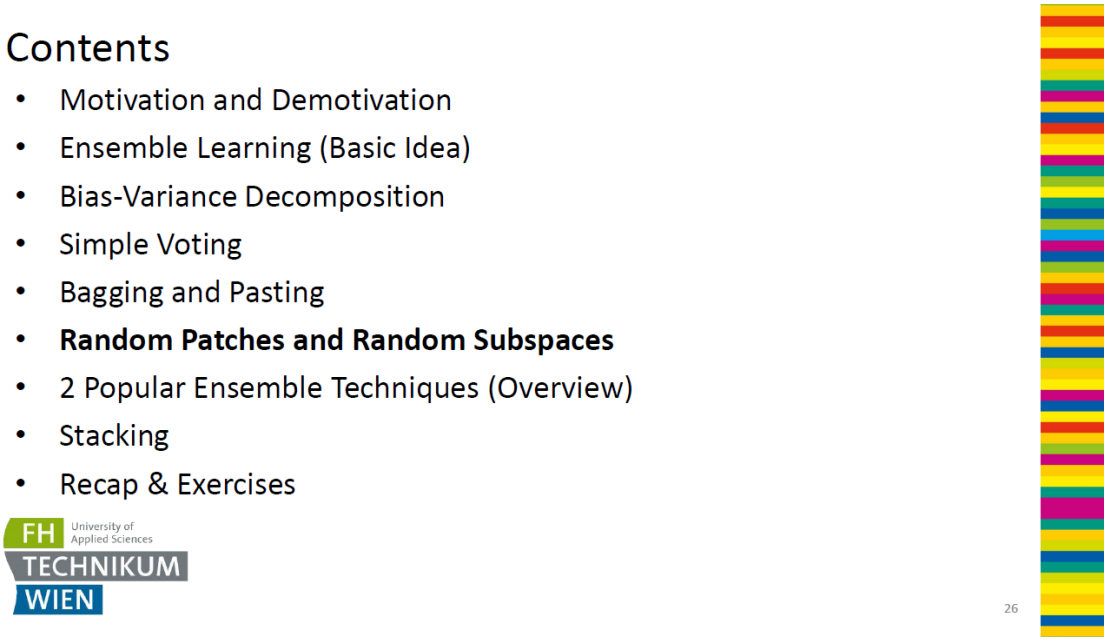
We see two curves:

- The **green curve** is for bagged decision trees. As we saw earlier, the performance improves with more models, but eventually **levels off**.
- The **red curve** shows logistic regression, which already performs **very well**, and does **not benefit much** from bagging.

Why is this?

- **Logistic regression is a low-variance model**. Since bagging reduces variance, there's just not much to gain here.
- On the other hand, decision trees are **high-variance** models, so bagging helps them generalize better.

But a word of caution: while these results are consistent on this dataset, **you should always try different approaches on your own data**, because the optimal strategy can vary depending on the problem."



Slide 26 – Contents (Checkpoint: Random Patches and Random Subspaces)

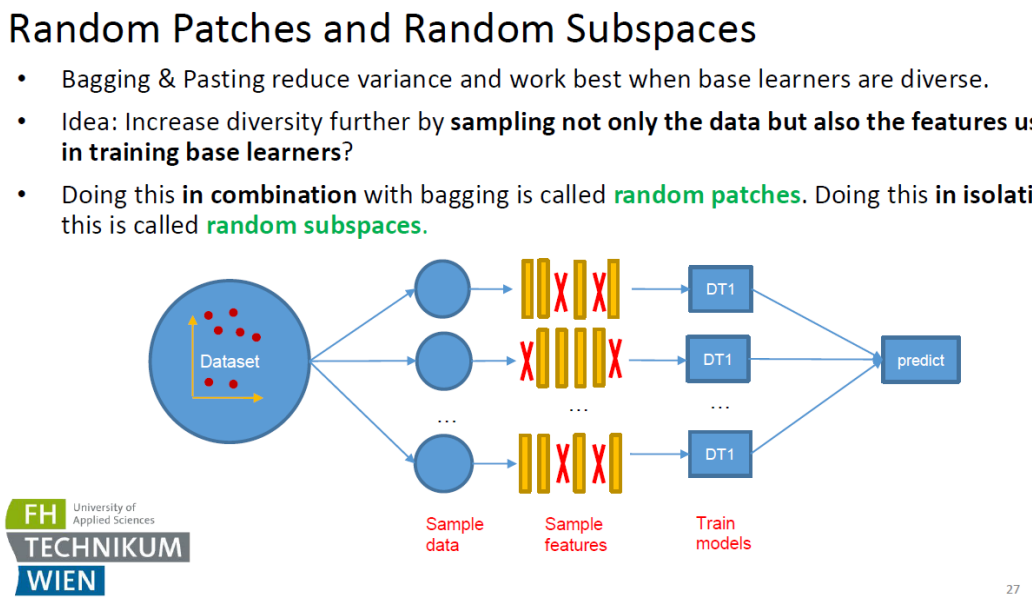
"At this point, we've covered the foundations of ensemble learning — from simple voting to bagging and pasting.

Now, we move to the next important idea: **Random Patches and Random Subspaces**.

These are techniques used to further increase the **diversity** among the base learners by not only sampling different data points — but also sampling **different features**.

This is especially relevant in algorithms like **Random Forest**, which rely on feature-level randomness to reduce correlation among trees.

Let's dive into how that works in practice."



Slide 27 – Random Patches and Random Subspaces

"We already saw that **bagging and pasting** reduce variance and perform well when base learners are diverse.

So how can we **increase diversity even more**?

The idea is: let's not only sample different data points — let's also sample **different sets of features**.

In other words, each model sees:

- A different subset of rows **and**
- A different subset of columns.

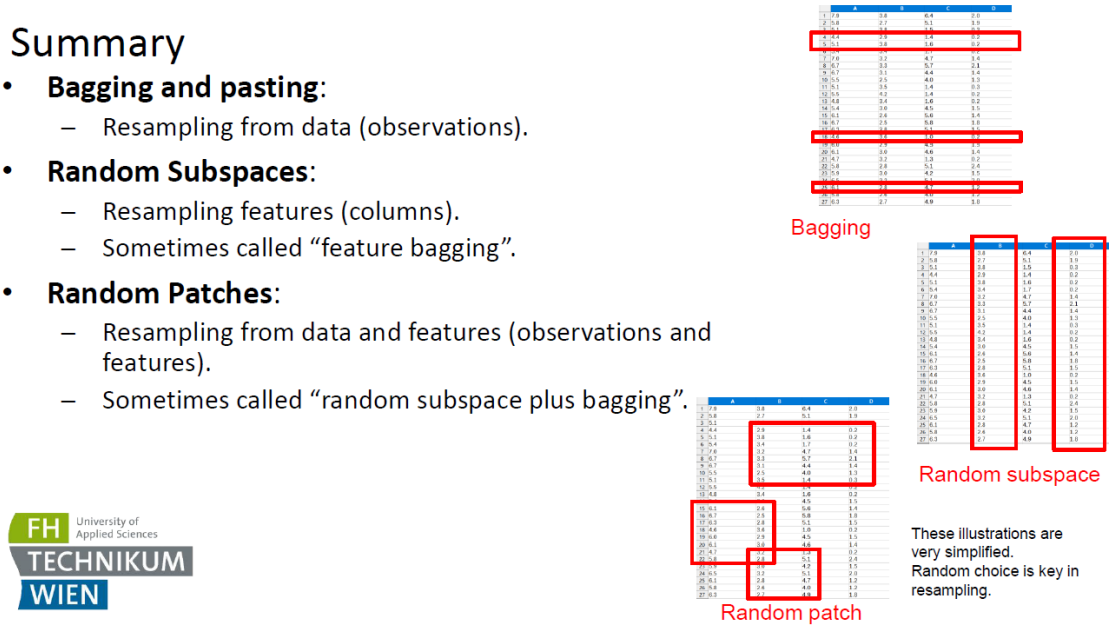
When we do **both** — sampling rows **and** features — we call this a **random patch**.

If we **only sample features**, and not rows, then it's called the **random subspace method**.

The diagram here illustrates that:

- From the full dataset, we draw several **random samples**,
- And from each sample, we select **only some features** (as shown by the red 'X's),
- Then, each model (in this case, decision trees) is trained on its own version of the data,
- Finally, predictions are aggregated.

This strategy is used in algorithms like **Random Forests**, where each tree is trained on both random instances **and** random subsets of features."



Slide 28 – Summary

”Let’s quickly summarize the main resampling strategies we’ve seen:

1 **Bagging and Pasting**

- Both involve resampling the **rows** of the dataset — that is, the observations.
- The difference? Bagging uses **sampling with replacement**. Pasting uses **sampling without replacement**.

2 **Random Subspaces**

- Instead of sampling data points, here we sample **features** — the columns.
- It’s sometimes referred to as **feature bagging**.

3 **Random Patches**

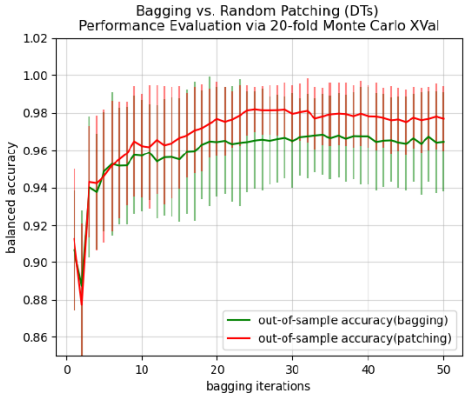
- This method samples **both**: the data points **and** the features.
- In other words, a learner might only see a slice of the data matrix — a ‘patch’.
- That’s why it’s also called **random subspace plus bagging**.

The diagrams on the right help to visualize these concepts:

- The top shows classic **bagging** (rows sampled).
- The middle shows **random subspaces** (columns sampled).
- And the bottom shows **random patches**, with both rows and columns sampled.

Keep in mind: the way we **randomly choose** rows and columns is **crucial** to ensure diversity among models and better ensemble performance.”

Bagging vs. Random Patches



- Patching increases the diversity of base learners.
- Increased diversity leads to higher gains from bagging.
- Since the diversity of base learners is higher, gains from bagging taper out later (roughly at ensemble size 25).

FH University of Applied Sciences TECHNIKUM WIEN

29

Slide 29 – Bagging vs. Random Patches

"Now let's compare **Bagging** with **Random Patching**, focusing on accuracy and ensemble size.

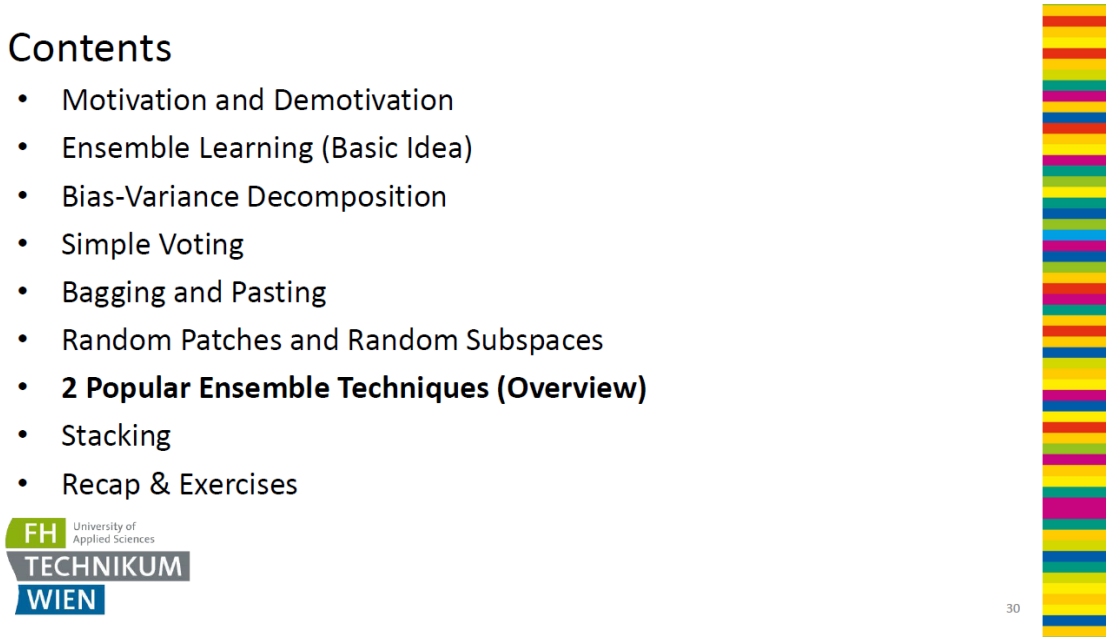
On the left, we see a performance chart based on 20-fold Monte Carlo cross-validation.

- The **green line** shows the out-of-sample accuracy of standard **bagging**.
- The **red line** shows the accuracy of **random patching**, which combines both data and feature sampling.

So what's happening here?

Patching increases diversity among base learners — because each learner sees not just different samples but also different **features**. This **higher diversity** leads to more **robust ensembles**, which in turn produce **better generalization** — as reflected in the consistently higher red line. However, because patching already adds a lot of diversity from the start, the **performance gains taper off later**, around **ensemble size 25**.

So if your goal is to maximize accuracy and you can afford the complexity, **random patching is a powerful strategy**."



Slide 30 – Two Popular Ensemble Techniques

"Next, we'll briefly look at **two of the most widely used ensemble techniques** in machine learning:

1. **Random Forests** – This is an ensemble of decision trees, typically built using **bagging** and **feature sampling** — which we've just seen as 'random patches.' – It's **robust, easy to use**, and often works well out of the box for both classification and regression problems.
2. **Boosting**, especially **Gradient Boosting Machines (GBM)** – Unlike bagging, boosting builds trees **sequentially**, where each new tree focuses on correcting the errors of the previous ones. – It's very **powerful**, but also **more sensitive to overfitting** and requires careful tuning.

These two methods represent different philosophies: – **Random Forest** reduces **variance** through parallel trees. – **Boosting** reduces **bias** by correcting mistakes over time.

In the next slides, we'll dive deeper into how each of these works."

Slide 31 – Random Forests

"Let's now focus on **Random Forests**, one of the most powerful and widely used ensemble methods.

- A **random forest** is essentially an ensemble of **unrestricted decision trees**, built using a **variant of random patches**.
- What makes it special is that, in addition to sampling data, it **resamples the features at every node**, not just once per tree. – That means, for each **split** in a tree, a random subset of features is chosen — which **increases diversity** and helps avoid overfitting.
- You can think of it as an **intensification of resampling**: – We move from simple bagging (just data), – to random patches (data and features), – to random forests (features resampled per split).

Finally, Random Forests are **so important** that we will dedicate an entire future lecture to understand them in detail."

Boosting

Base learners are trained **sequentially** (not in parallel as in bagging).

Aim: Reduction of both, bias and variance.

Two approaches:

1. Critical datapoints are reweighted to emphasize “complicated” obervations.
2. Single models are fitted to the residuals of the previous models.

Boosting will be the focus of a dedicated lecture in this course.

FH University of Applied Sciences TECHNIKUM WIEN

32

Slide 32 – Boosting

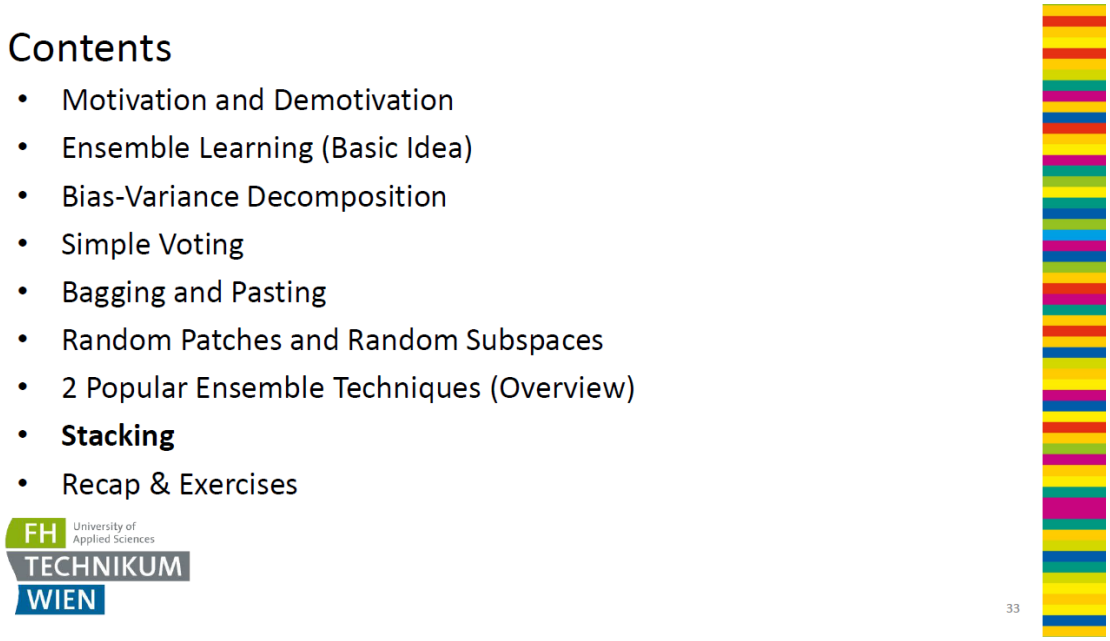
”Let’s wrap up this section by briefly introducing **Boosting**, another major ensemble learning technique.

- Unlike bagging, where models are trained **in parallel**, **boosting trains base learners sequentially**.
- Each model tries to **correct the mistakes** of the previous ones.
- The goal here is to **reduce both bias and variance**.

There are two main approaches:

1. First, boosting can **reweight difficult data points**, giving more attention to misclassified or hard-to-learn observations.
2. Second, it can **fit new models to the residuals** — the errors — of previous models, gradually improving performance.

And just like random forests, **boosting will have its own dedicated lecture** later in the course, where we’ll explore it in detail.”



Slide 33 – Stacking

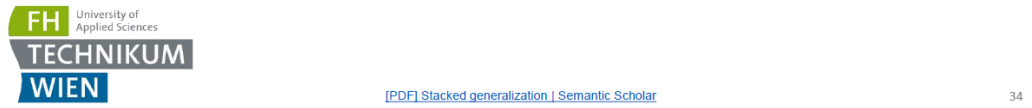
"Finally, let's introduce **Stacking**, another powerful ensemble technique.

- In stacking, we don't just vote or average outputs like in bagging or boosting.
- Instead, we use the predictions of several base models as **inputs to a new model**, called a **meta-learner**.

This meta-model learns **how to best combine** the base models' outputs to improve overall performance.

- For example, you might train a decision tree, a logistic regression, and a neural network as your base learners.
- Then, the meta-learner — say, a linear model — takes their predictions and learns how to weight them optimally.

Stacking often **outperforms individual models and simpler ensembles**, but it can be trickier to train and validate."

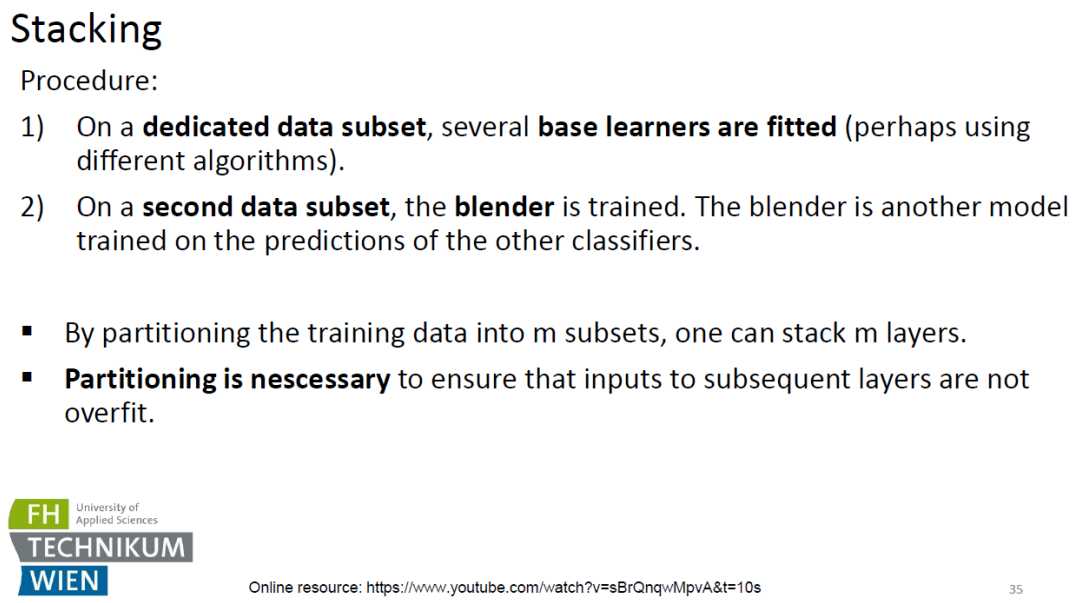


Slide 34 – Stacking (continued)

"To deepen the explanation of stacking:

- Stacking is an **extension of voting classifiers or regressors**. But instead of simply taking the majority vote or the average of predictions, stacking introduces a **meta-model**, sometimes called a **blender**.
- This blender is **another classifier or regressor** trained to combine the outputs of the base models in an optimal way.
- The idea is **vaguely similar to deep learning**: each layer builds on top of the previous one. Here, base models make predictions, and the blender uses those predictions as input.
- Like in voting, **different types of models can be used** — for example, decision trees, SVMs, and neural networks — all contributing to the final output.

Stacking can capture patterns that individual models may miss, making it a powerful ensemble technique."



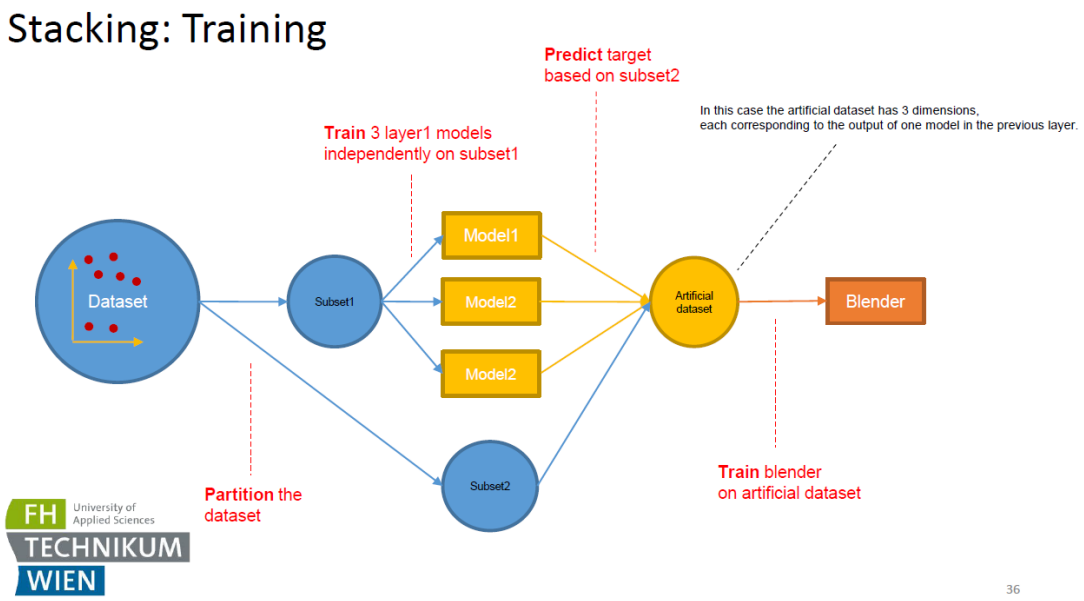
Slide 35 – Stacking (Procedure)

"Now let's look at **how stacking works in practice**:

1. First, we divide the data. On a **dedicated data subset**, we train multiple **base learners** — for instance, a decision tree, an SVM, and a neural network. These models are trained independently and possibly use different algorithms.
2. Then, on a **second data subset**, we train the **blender** — a model that takes the predictions of the base learners as its input and learns how to best combine them.

By **partitioning the training data into multiple subsets**, we can build **multiple layers**, stacking as many times as needed.

And importantly: **partitioning is necessary** to prevent overfitting. If we train the blender on the same data used to train the base models, it will simply learn to replicate their mistakes."



Slide 36 – Stacking: Concept and Training Overview

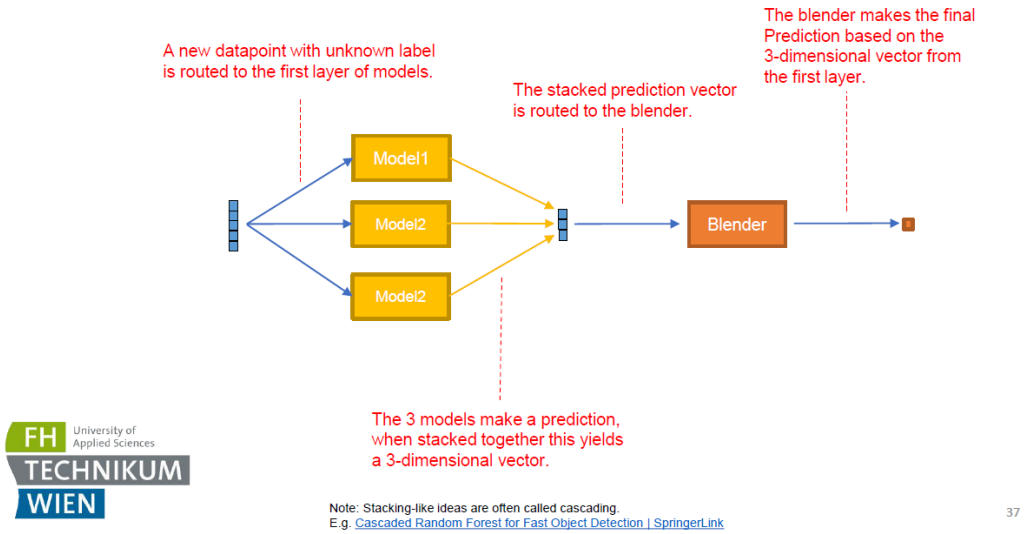
"This slide explains how stacking works from both a conceptual and procedural perspective.

First, the original dataset is split into two subsets. We train several **base models** (e.g., a decision tree, SVM, and neural network) on the first subset. These models then make predictions on the second subset.

Those predictions are **not final** — instead, they form a new dataset, where each feature represents a model's prediction. This is called the **artificial dataset**.

A second-level model, the **blender**, is trained on this artificial dataset to learn how to best combine the base models' outputs.

Unlike simple voting, stacking allows the blender to learn complex relationships between model predictions, often leading to improved accuracy."



Slide 37 – Stacking: Prediction

”Now let’s look at how stacking makes predictions once it’s trained.

When a **new datapoint** with an unknown label comes in, it is first passed to the **base models** — in this case, three models in the first layer.

Each of these models produces a prediction — typically a class probability or a continuous score.

These predictions are **stacked together** to form a **vector**, which becomes the input to the next stage.

This vector is then passed to the **blender**, which is the model trained on such prediction vectors during the training phase.

The **blender now makes the final prediction** — based not on raw features, but on the collective predictions from the base learners.

This architecture allows stacking to leverage the strengths of multiple models and refine their output through a meta-learner.”

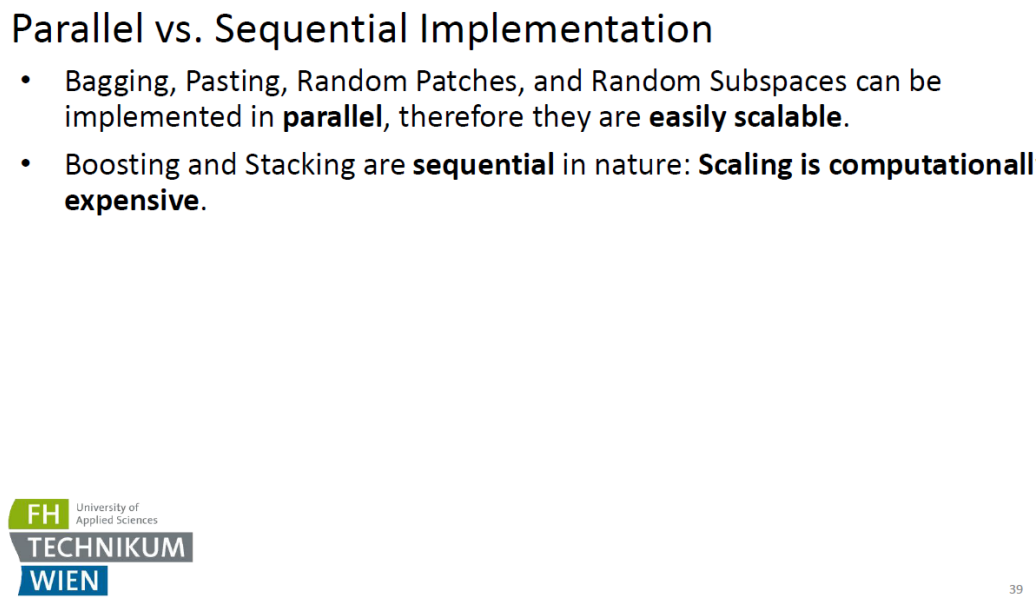
Slide 38 – Stacking: Regression vs. Classification

"In stacking, the structure of the first layer depends on whether we are solving a **regression** or a **classification** problem.

- In **regression**, the base models in the first layer output **real-valued predictions** — typically continuous numbers.
- In **classification**, the models output **categorical values**, often in the form of class probabilities.

To make stacking more effective for classification, especially when using **scikit-learn**, it is best practice to use the method `predict_proba()` in the first layer. This provides richer information to the blender by outputting probability distributions over classes, rather than just class labels.

This distinction is important because the blender learns from what the first layer provides — so we need to give it the right type of input."

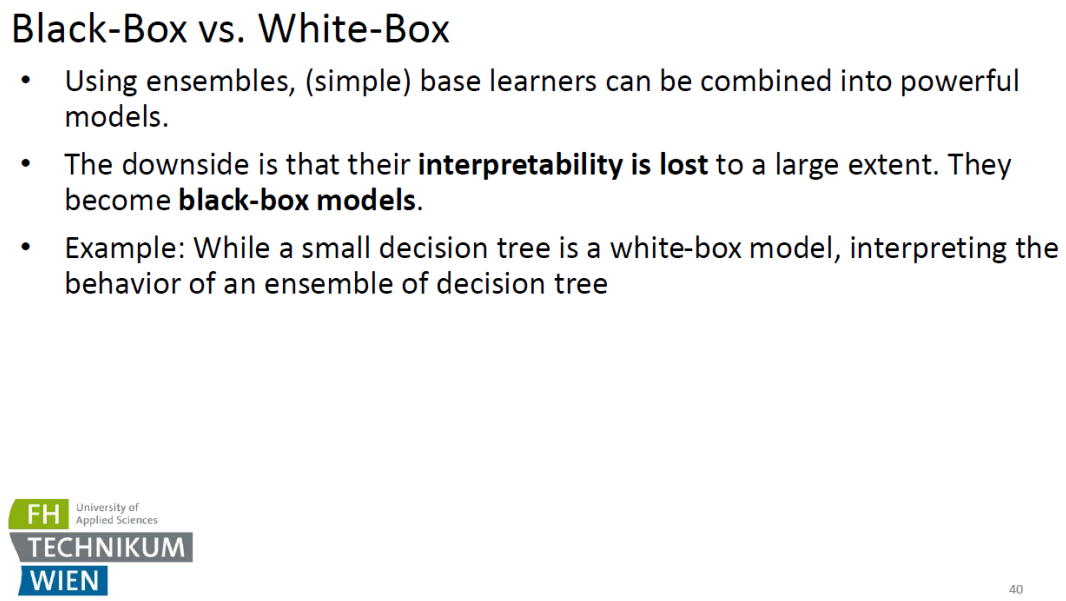


Slide 39 – Parallel vs. Sequential Implementation

"In this slide, we compare two fundamental implementation styles in ensemble learning: **parallel** versus **sequential**.

- Methods like **Bagging**, **Pasting**, **Random Patches**, and **Random Subspaces** can be executed **in parallel**. That means all base learners can be trained simultaneously, making these methods highly **scalable** and well-suited for distributed or multi-core systems.
- On the other hand, techniques like **Boosting** and **Stacking** follow a **sequential** structure. Here, each new model depends on the output of the previous one. This makes them **more computationally expensive** and harder to parallelize.

So, the tradeoff is between scalability and the power of dependency-aware learning."

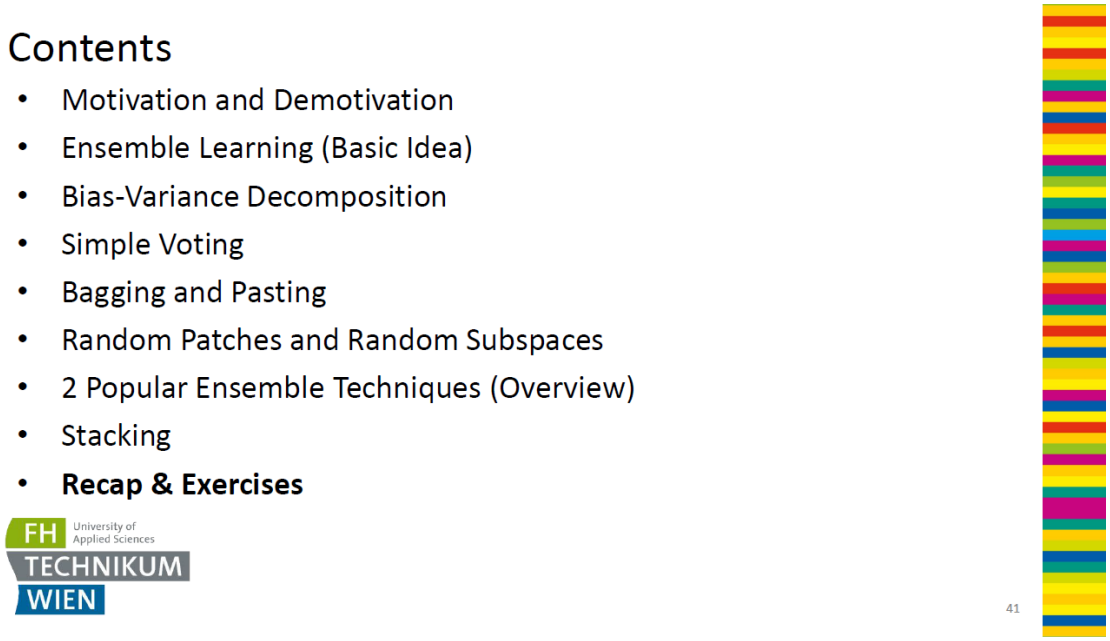


Slide 40 – Black-Box vs. White-Box

"This slide touches on a key tradeoff in ensemble learning: **performance vs. interpretability**.

- When we use ensembles of simple base learners—like decision trees—we often get highly **accurate and powerful models**.
- However, this power comes at a cost: **interpretability**. The individual models might be easy to understand, but the combined behavior is not. This turns the ensemble into a **black-box model**.
- For example, a single, small decision tree is a **white-box**—we can easily interpret its logic. But if we combine hundreds of them, as in a random forest or gradient boosting, **understanding the full ensemble becomes very difficult**.

So while ensembles are great for prediction, we must be aware that they sacrifice explainability in the process."



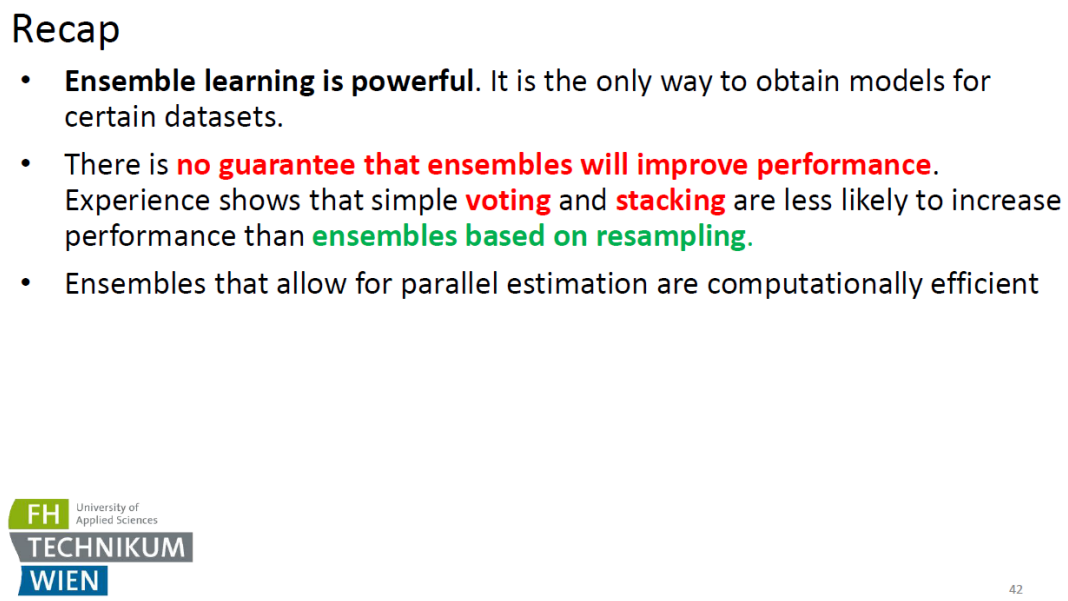
Slide 41 – Recap & Exercises

"Let's conclude this lecture with a brief recap of what we covered:

- We started with the **motivation** for ensemble learning, showing how combining models can overcome individual weaknesses.
- Then we introduced the **basic idea** of ensemble methods—combining multiple classifiers or regressors to improve performance.
- Through the **bias-variance decomposition**, we understood how different models behave and how ensembles can reduce error.
- We explored **Simple Voting** (hard and soft), and how it helps in bias reduction.
- We saw how **Bagging and Pasting** improve generalization by resampling the data.
- Then we went further with **Random Patches and Subspaces**, which add diversity by sampling features too.
- We reviewed **Random Forests** and **Boosting** as two popular ensemble strategies.
- **Stacking** brought another level of sophistication—using one model to learn from the outputs of others.
- Finally, we discussed computational tradeoffs and the **interpretability challenge** of ensemble models.

Now it's time to **apply what you've learned** with some exercises.

Thanks for your attention, and good luck!"



Slide 42 – Final Recap

"To wrap it up:

- **Ensemble learning is powerful** — for many complex datasets, it's the only viable way to achieve high-performing models.
- But a **word of caution**: there's **no guarantee** that an ensemble will outperform a strong single model. Especially simple methods like **voting** or **stacking** don't always bring significant gains.
- In practice, **resampling-based ensembles**—like bagging or random forests—often outperform simple combinations.
- And finally, **parallelizable ensembles** (e.g. bagging, random patches) offer a **big advantage in scalability**, unlike sequential methods like boosting or stacking, which can be **computationally expensive**.

References

- Géron A. (2017): Hands-On Machine Learning with Scikit-Learn & Tensorflow. – O'Reilly.
- James G., Witten D., Hastie T., Tibshirani R. (2017): An introduction to Statistical Learning. – Springer.
- Kuhn M., Johnson K. (2016): Applied Predictive Modeling. – Springer.
- Berk R. (2016): Statistical Learning from a Regression Perspective. – Springer.
- Bagging predictors | SpringerLink
- The random subspace method for constructing decision forests | IEEE Journals & Magazine | IEEE Xplore

FH University of Applied Sciences TECHNIKUM WIEN

44