Supervised ML

Institute of Technology of Cambodia

November 22, 2023

Decision Tree

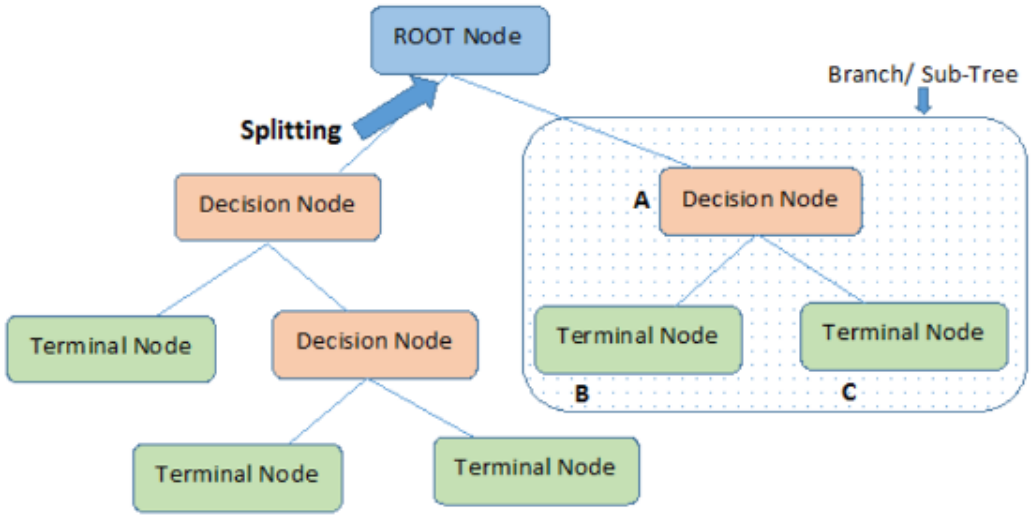
- Decision tree is a simple yet powerful way to do machine learning
- Meta-algorithm:
  - Recursively split the data in 2 groups of items, based on chosen attribute, so that elements in the same group have as close target values as possible
  - Predict that the value of a new item is the same as those of the group it belongs to

Decision Tree

Decision tree algorithms are commonly used for both classification and regression tasks. The type of task they are applied to depends on the nature of the problem and the type of output variable you are trying to predict.

- Regression: decision trees are used to predict a continuous numeric value.
- Classification: decision trees are used to predict the categorical class labels of instances.

Decision Tree



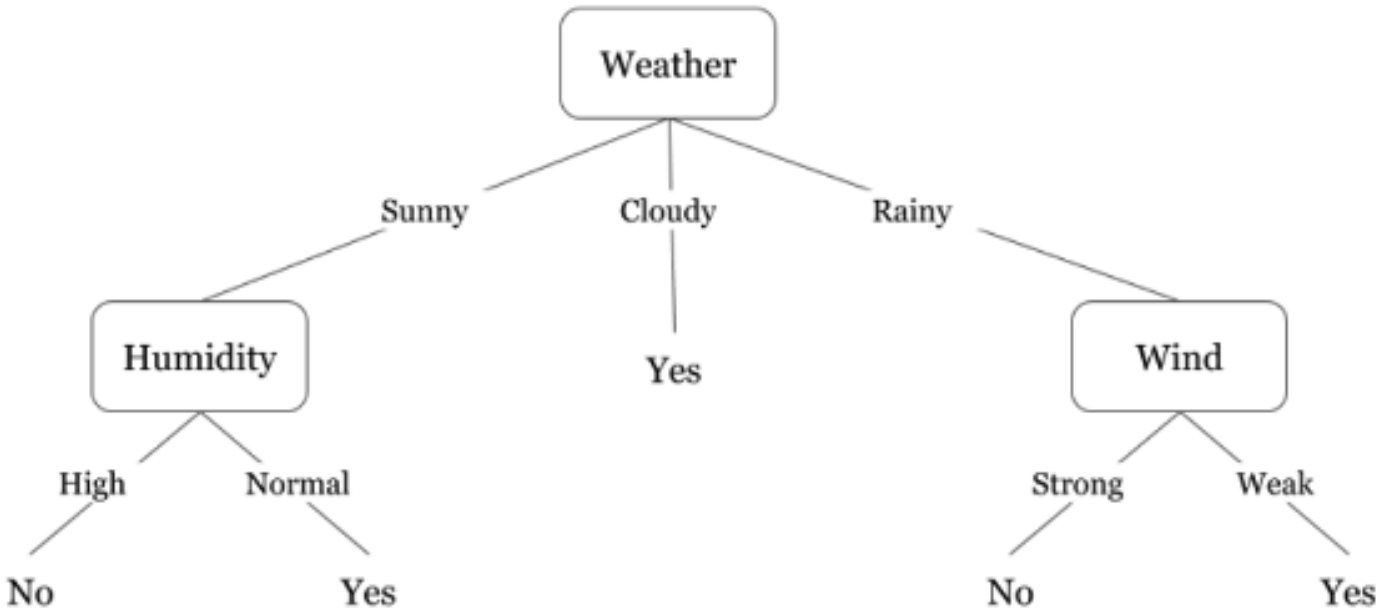
- Root node: initial node
- Decision node: nodes represent intermediate decisions or conditions within the tree

Decision Tree

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Decision Tree

- What is the weather?
- Is it sunny, cloudy, or rainy?



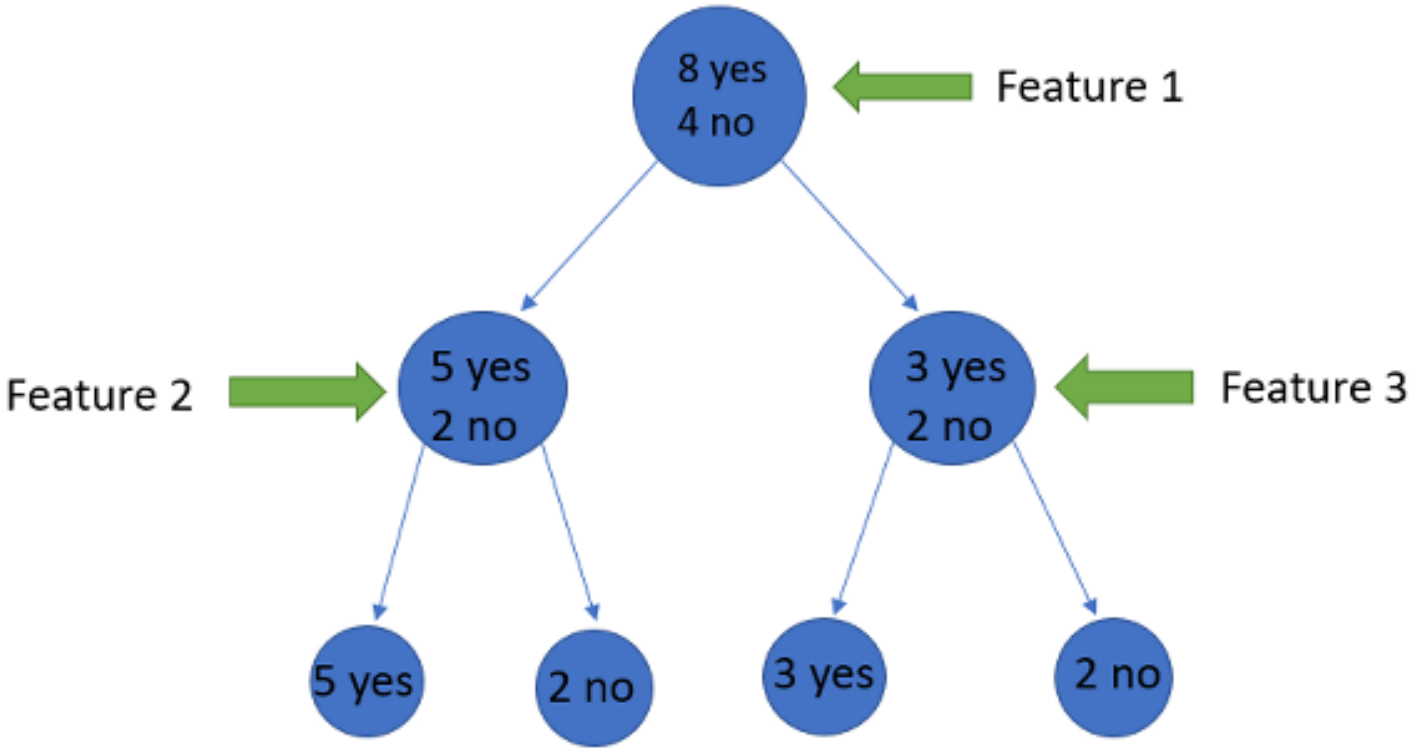
Decision Tree

- Why didn't it split more?
- Why did it stop there?

To answer this question, we need to know more concepts (entropy, information gain, and Gini index). But in simple terms, I can say here that the output for the training dataset is always yes for cloudy weather, since there is no disorderliness here we do not need to split the node further.

Decision Tree

Entropy basically measures the impurity of a node. Impurity is the degree of randomness; it tells how random our data.

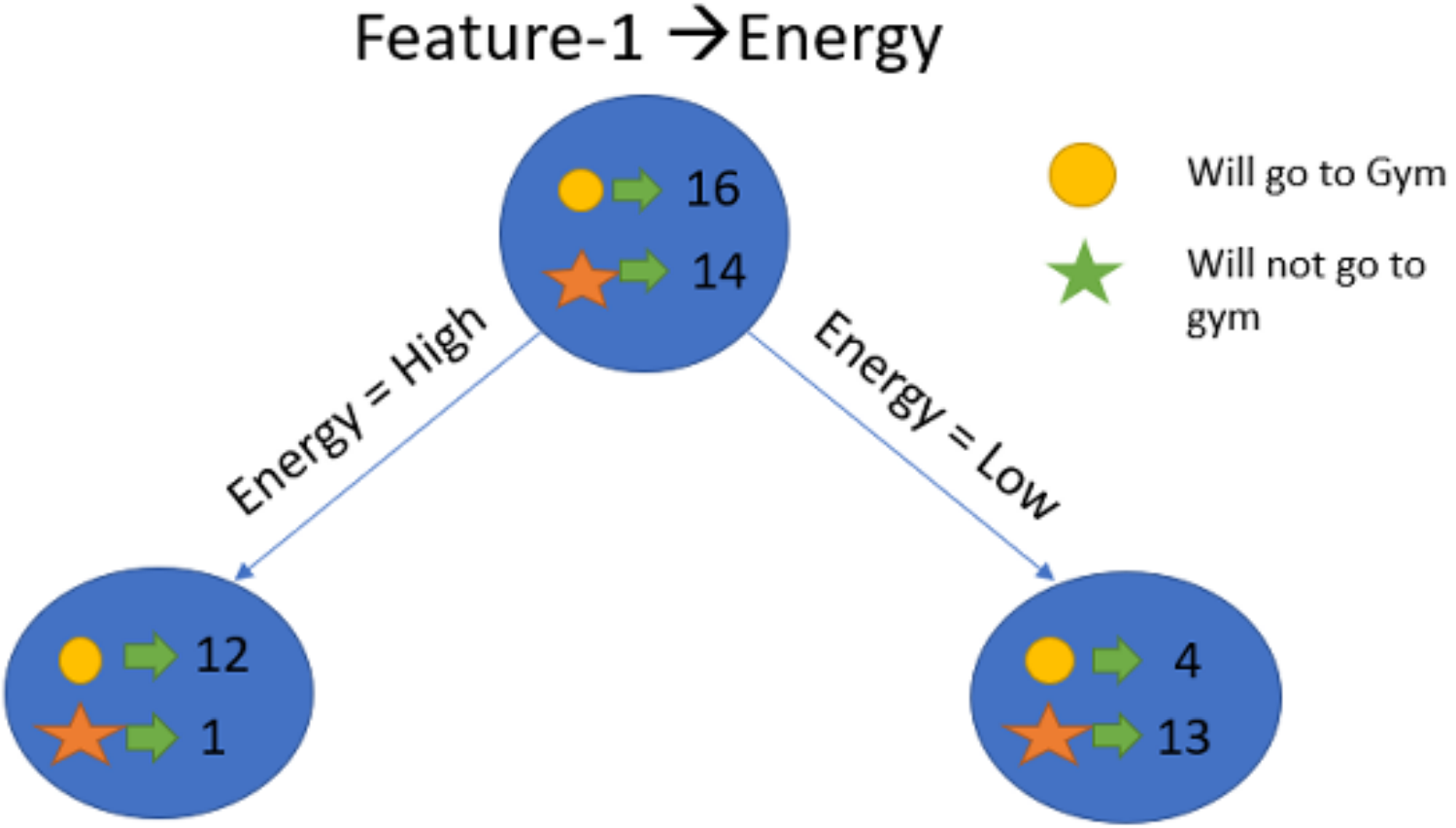


Decision Tree

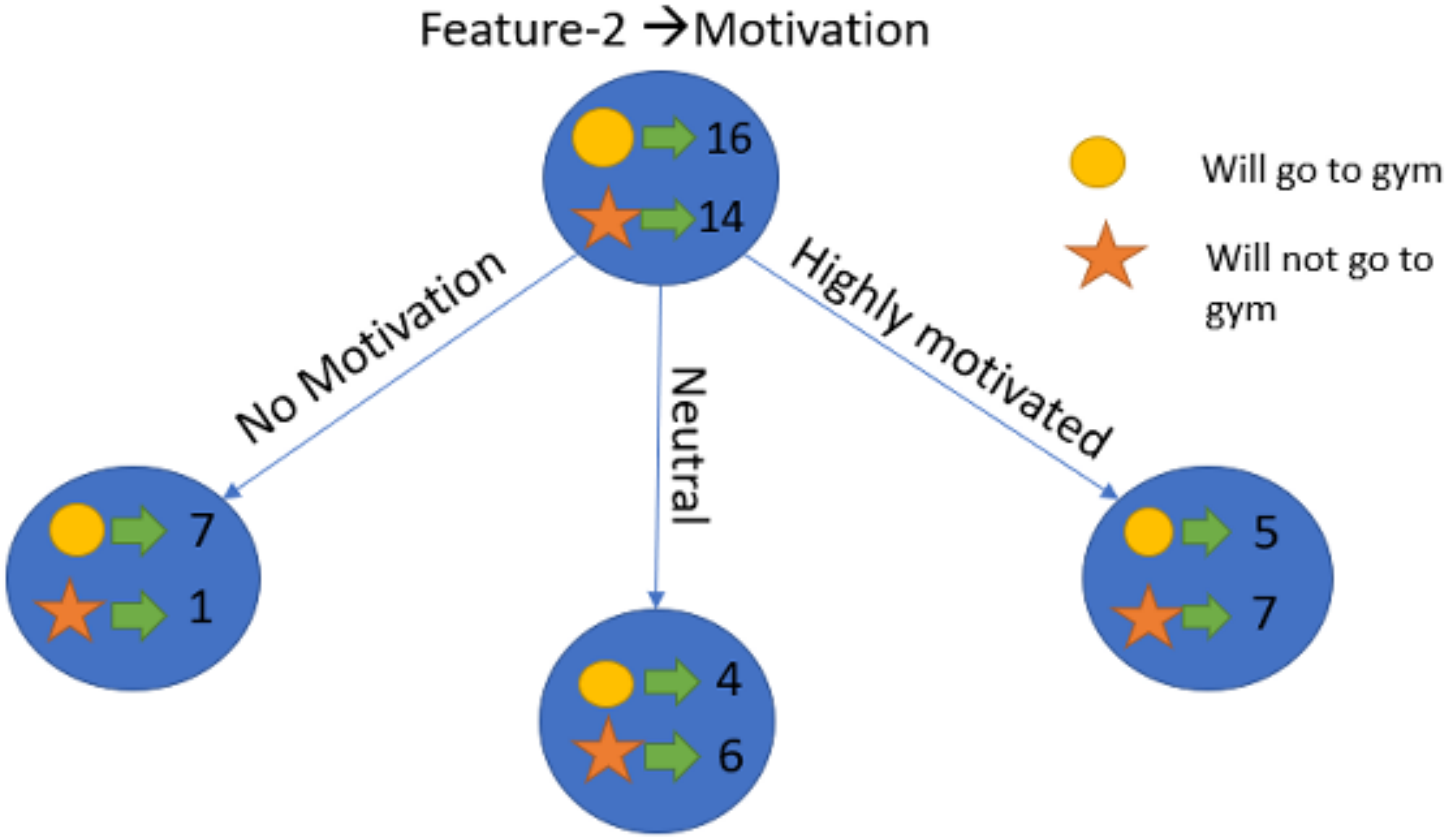
Information Gain measures the reduction of uncertainty given some feature and it is also a deciding factor for which attribute should be selected as a decision node or root node.
Suppose our entire population has a total of 30 instances. We want to predict whether the person will go to the gym or not based on two features.

1. "Energy" which takes two values "high" and "low"
2. "Motivation" which takes 3 values "No motivation", "Neutral" and "Highly motivated".

Decision Tree



Decision Tree



Decision Tree

When to stop splitting?

- min_samples_leaf: represent the minimum number of samples required to be in the leaf node. The more you increase the number, the more is the possibility of overfitting.
- max_features: it helps us decide what number of features to consider when looking for the best split.

Machine Learning: Solved

Or is it Overfitting and Underfitting?

Overfitting and Underfitting

- Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.
- Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data.

Avoiding Overfitting

- The most important rule of machine learning
  - And essential part of the scientific process
- Predicting what you already know is cheating
  - Even if you genuinely try not to cheat, you can cheat unintentionally
  - Experimental scientific experiments are done in **double blind**:
    - Neither the tested subject **nor the experimenter** know the placebo from real pill
- You mist hide a **test set**, that you will **never** use when learning, and that you will **only use once**, for evaluating

Avoid Overfitting

- When your data is ready, before any learning, split your data into:
  - A **training** set
  - A **test** set
- You can train as many method with as many parameters as you want on the training set
- Only when all your models are trained, you can evaluate it on the test set
  - You can never, ever reuse that (exactly same) test set

Train/Evaluation/Test

In some cases, you need to see the results on the test set to know how to improve your prediction

- Ex: how many levels in my decision tree? The right level is the one with the best results on the test set
- More generally: hyperparameter tuning
  - If my learning method has parameters, how to fix those parameters? (Coefficient of learning, number of layers in deep NN, etc)

Solution

- Use an evaluation set for intermediary steps (hidden like test set, but not for final evaluation)
- Keep a test set, that you will use only once at the end

Train/Test Split

What size should your test set have?

- No good answer: 66% Train, 33% Test is often a default choice

Rule of thumb:

- You need enough data for training. If your problem is simple (few features...) and you have many examples, then a random sample of 5% of it can be enough
  - e.g: predict weight of a tomato based on its species

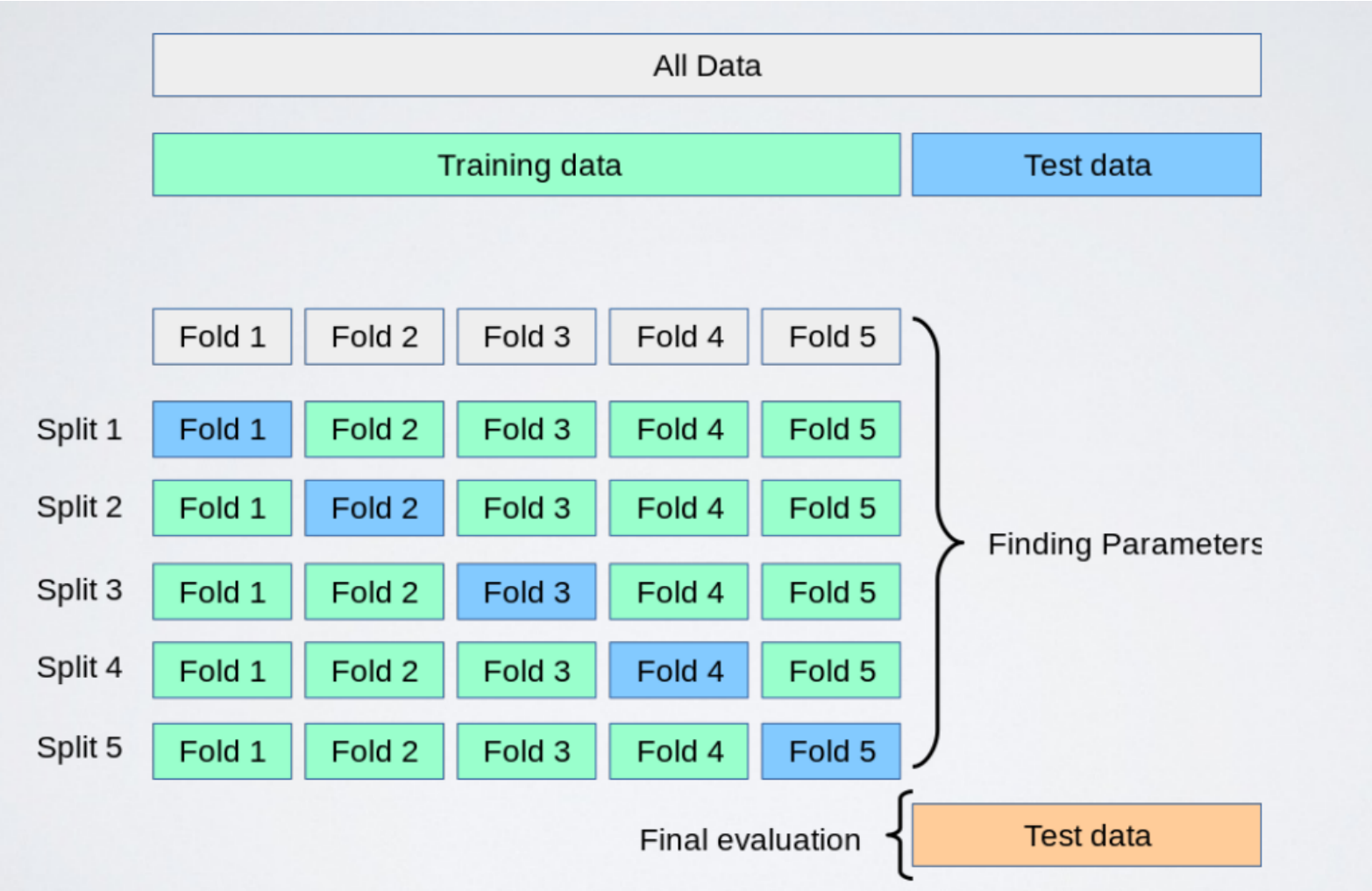
Problem is if data is scarce

- Cross validation

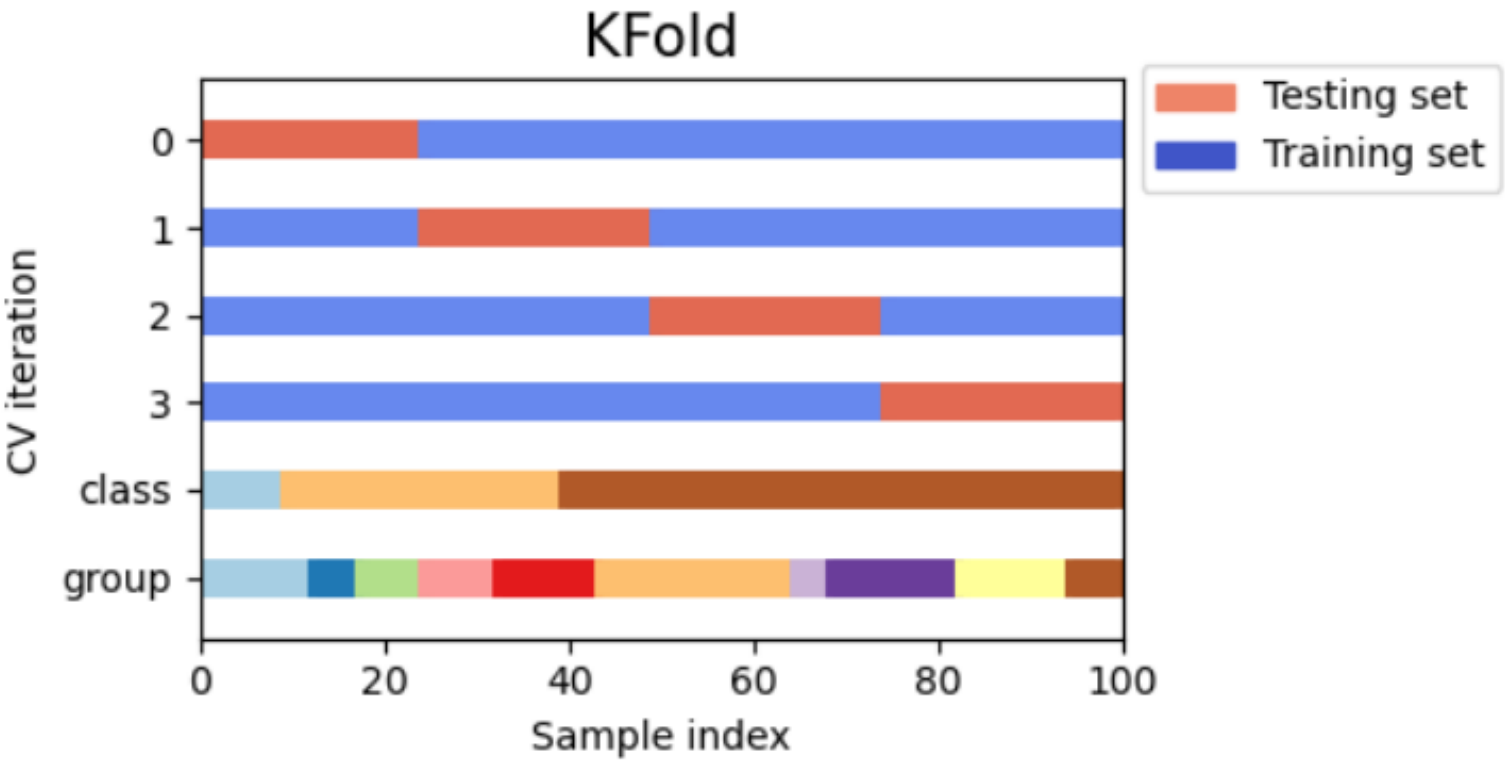
Cross Validation

- Gloden rule: One test set must be used only once
- But from a single dataset, you can create multiple test sets
  - If you just want to restart you training differently, make a new 66/33 split
- If you have too few data to put 33% aside for training:
  - Use only, e.g., 10% for the test set
  - But create 10 train/test sets, each with different test sets
    - Then, compute the average scores over the 10 sets

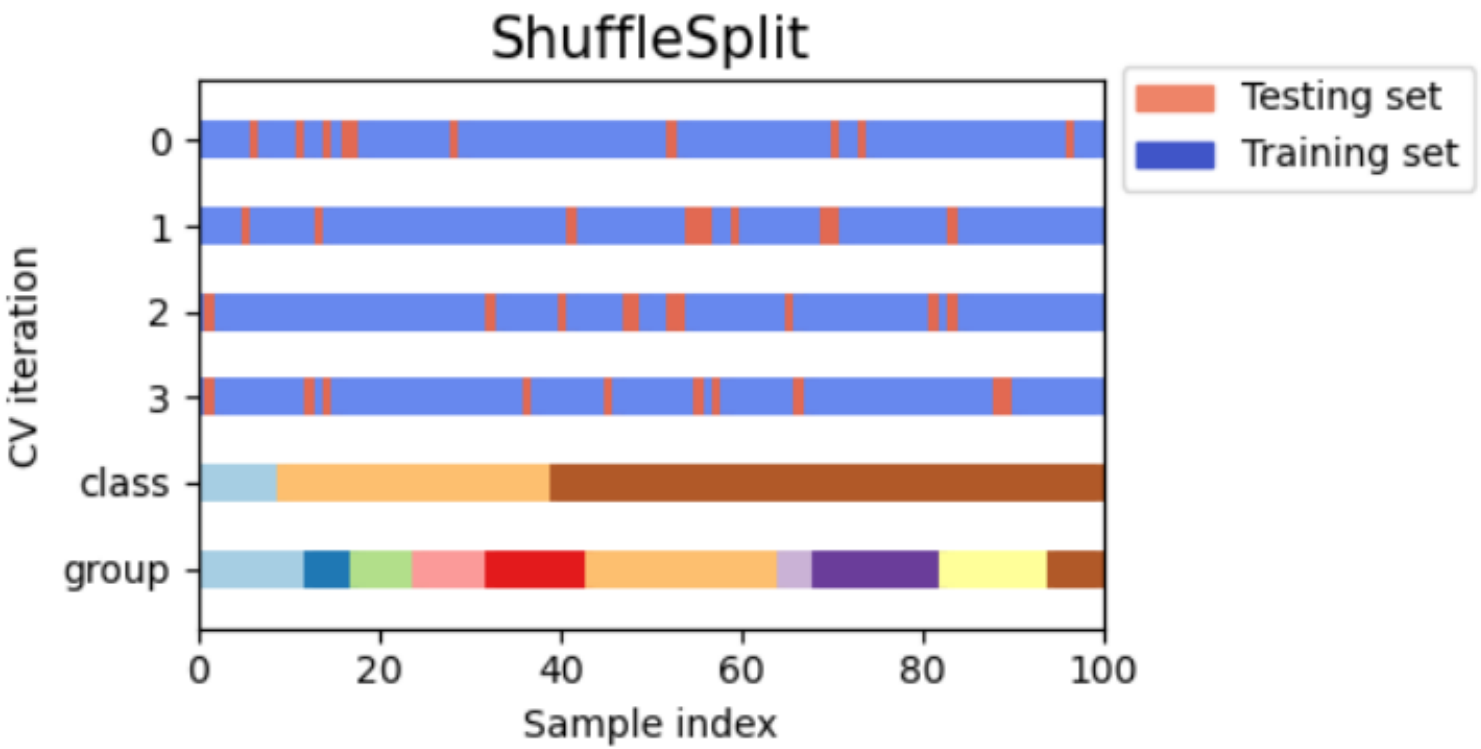
Cross Validation



Cross Validation



Cross Validation



Cross Validation

- Do you need Cross Validation and which one?
- In my opinion:
  - If your data is very scarce, you need it from the beginning
  - If you are at a point in which every fraction of a % of accuracy count
- A simple train/test is enough for most cases

Fighting Overfit and Underfit (Back to the Method)

Fighting Overfit

Implicit limit to overfit:

- Because a method has a limited power of expression, it cannot overfit "too much"
  - Trivial solution: each point has its own prediction. No **generalization**
- A linear regression method cannot overfit to the trivial solution, unlike decision tree
  - Unless there are enough variables...

Explicit limit to overfit:

- The method is not limited in its power of expression, but contains a safeguard against overfit

Not necessarily a clear boundary between the 2

Fighting Overfit

Avoiding overfit in decision trees: Pruning strategies

- One way to see: Artificially limit the expressivity of the model
- 1) Limit the number of levels (Simple but naive)
- 2) Limit the number of leaves
  - Split nodes in priority where it improves the most
- 3) Limit the size of leaves
  - Explicitly forbids the naive solution

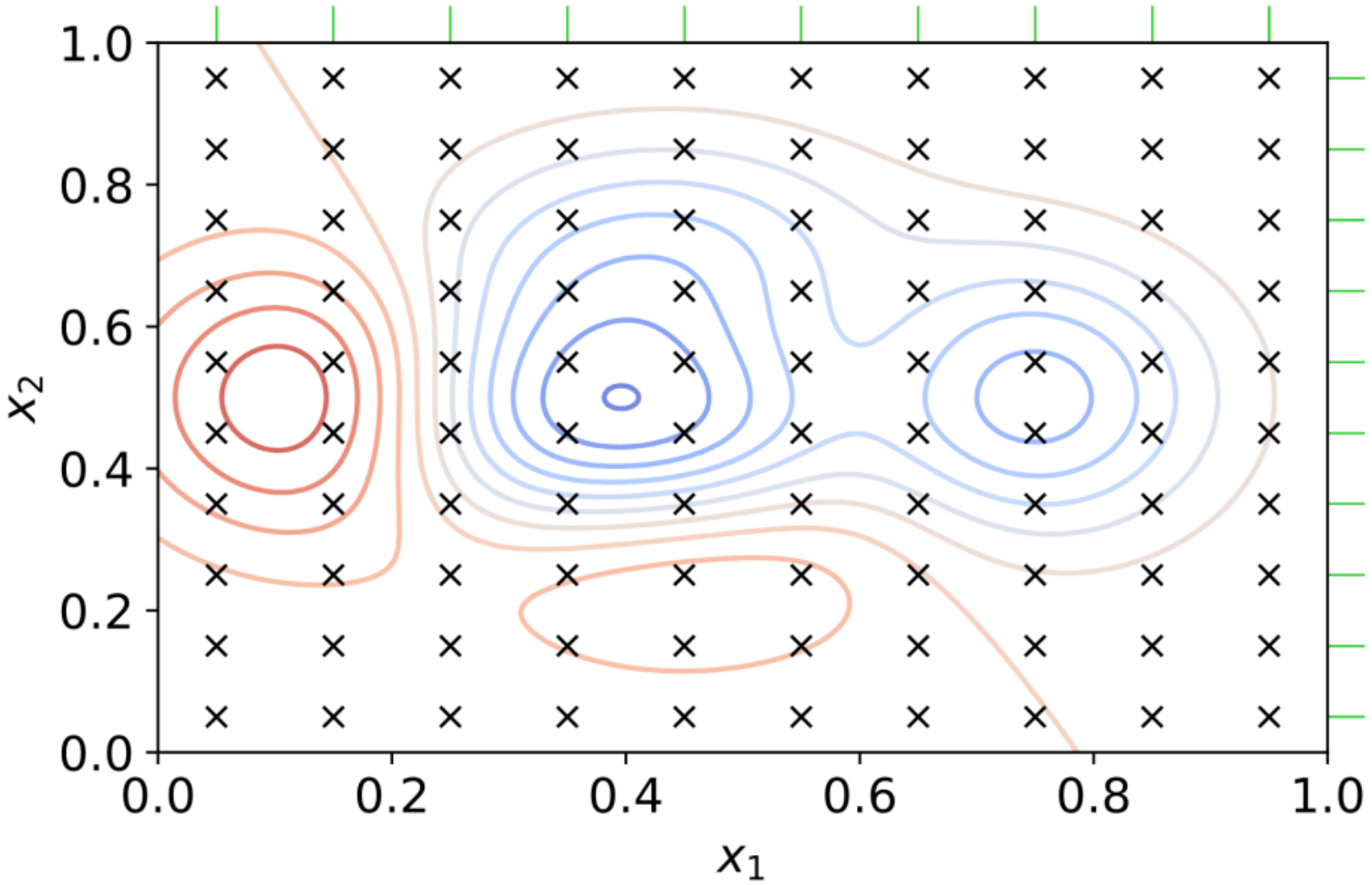
Hyperparameter tuning/optimization

- Typical approach: Grid search
- Fix a set of possible parameters. Test all possibilities on a validation test

Fighting Underfit

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
- Reducing the constraints on the model (e.g., reducing the regularization hyperparameter)
- Collect more data

Grid Search



More clever methods exist: Bayesian optimization, etc.

Note: Generalization

A very important notion in machine learning is Generalization

- Can we extract generic principles underlying our data?
- Can we generalize our observations to unseen cases?

Linear regression can predict an unseen value, while decision tree cannot

- What the weather be like in 5 years? Extrapolation from current condition...

Machine learning project

- Look at the big picture
- Get the data
- Discover and visualize the data to gain insights
- Prepare the data for machine learning algorithms
- Select a model and train it
- Fine-tune your model

How do we find the dataset

There are several websites to find dataset

- `https://www.kaggle.com/`
- `https://archive.ics.uci.edu/datasets`
- `https://datasetsearch.research.google.com/`
- `https://www.reddit.com/r/datasets/?rdt=48892`