Crash course on Machine Learning

Artificial Intelligence and Data Mining Methods in Ecology

University of Tuscia – Viterbo, July 21–25, 2025

Alex Falcon
Beatrice Portelli
University of Udine

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 "Education and Research" - Component 2: "From research to business" - Investment 3.1: "Fund for the realisation of an integrated system of research and innovation infrastructures"

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani

Where We Left Off

From good practices to good inputs

✅ We discussed how to set up ML workflows properly:

- Train/test splits & reproducibility
- Grid search for hyperparameter tuning
- Red flags in real-world datasets

✅ You saw how hand-crafted features like color histograms, SIFT, and HOG can help extract meaning from raw data

🧠 The overall message: clean, thoughtful data → better results

Where We're Going Today

ML models and the data they eat

- 📈 Introduction to classical ML models
  → classification, regression, and clustering
- 🧃 Deep dive into data preprocessing:
  - Encoding categorical, cyclical, and textual data
  - Normalization and scaling techniques

📌 Machine learning models are picky eaters… they need well-prepared input

By the end of today, we should be able to…

✅ Describe basic ML models for regression, classification, clustering

✅ Know when to use each type of model

✅ Understand why and how to preprocess different types of data

✅ Choose and apply the right scaler or encoder

✅ Recognize when poor data prep is sabotaging your model

Day 1

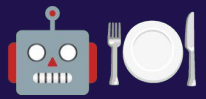
Why AI for Environment?

Day 2

Best Practices

Day 3

ML & Preprocessing



What is Machine Learning?

Training process of a **software**, called **model**, so that it learns to make **predictions** or generate contents starting from some input data.

*"A computer program is said to **learn** from **experience** E with respect to some class of **tasks** T and **performance measure** P, if its performance at tasks in T, as measured by P, **improves** with experience E."*

— Tom Mitchell, 1997, Machine Learning

Let's say we want to build a *system to forecast tomorrow's weather.*

How to do that?

What is Machine Learning?

"Traditional" approach: (based on our totally relevant background)

(1) create a physical representation of the Earth, its surface, the atmosphere, … maybe something else?

(2) use lots of fluid dynamics equations? Or some kind of complex equations

(3) ???

(??) we (maybe?) have a model that gives an answer

What is Machine Learning?

“Innovative” approach:

(1) collect a dataset with relevant information

(2) design a ML model

(3) train it until it starts to work well

(4) you got your forecasting model!


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

xkcd 1838

Types of ML

Four main types:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Generative AI

Types of ML

Four main types:

- Supervised Learning
  - The model is trained iteratively after seeing many examples with the **correct** answer.
  - Similar to a student: it “studies” old exercises/exams and learns how to answer correctly.
- Unsupervised Learning
- Reinforcement Learning
- Generative AI

Types of ML

Four main types:

- Supervised Learning
- Unsupervised Learning
  - The model is **not** given correct answers, instead it searches **meaningful patterns** in the data.
  - E.g. it is given images of cats, dogs, and mice, and tries to separate them.
- Reinforcement Learning
- Generative AI

Types of ML

Four main types:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
  - Models are required to achieve a goal (e.g. drive a car from A to B without crashes).
  - Models get **rewards**/penalties based on actions they perform within an environment.
  - They generate a **policy** defining the strategy to maximize the rewards.
- Generative AI

Types of ML

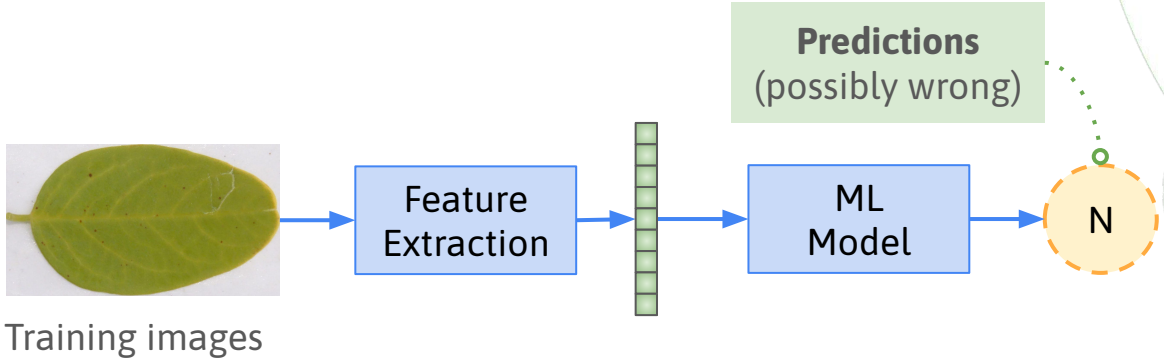
Four main types:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning
- Generative AI
  - The model **generates "new" content** based on user input.
  - E.g. DALL-E creates images from textual input
  - E.g. ChatGPT can help in writing code

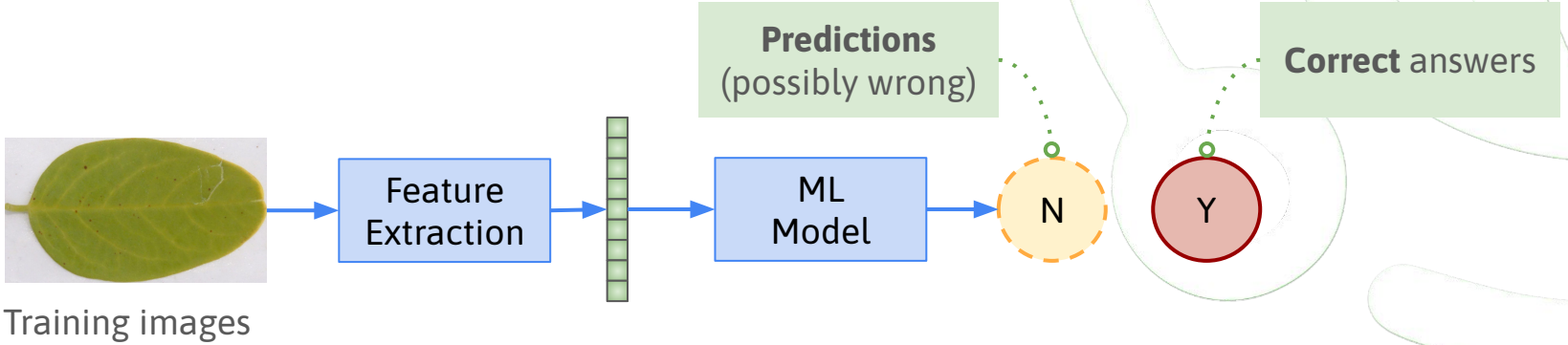
https://openai.com/index/dall-e-3/ → generate an image showing some students taking a phd course on machine learning applied to ecology

Supervised ML: Training and Inference

Supervised ML - Training phase



Supervised ML - Training phase



Supervised ML - Training phase

Training images

Feature Extraction

ML Model

Predictions (possibly wrong)

N

Y

Correct answers

A quantification of the "**error**"/**cost**

Cost/loss function

Low cost

Supervised ML - Training phase

Training images

Feature Extraction

ML Model

Predictions (possibly wrong)

N

Y

Correct answers

A quantification of the "**error**"/**cost**

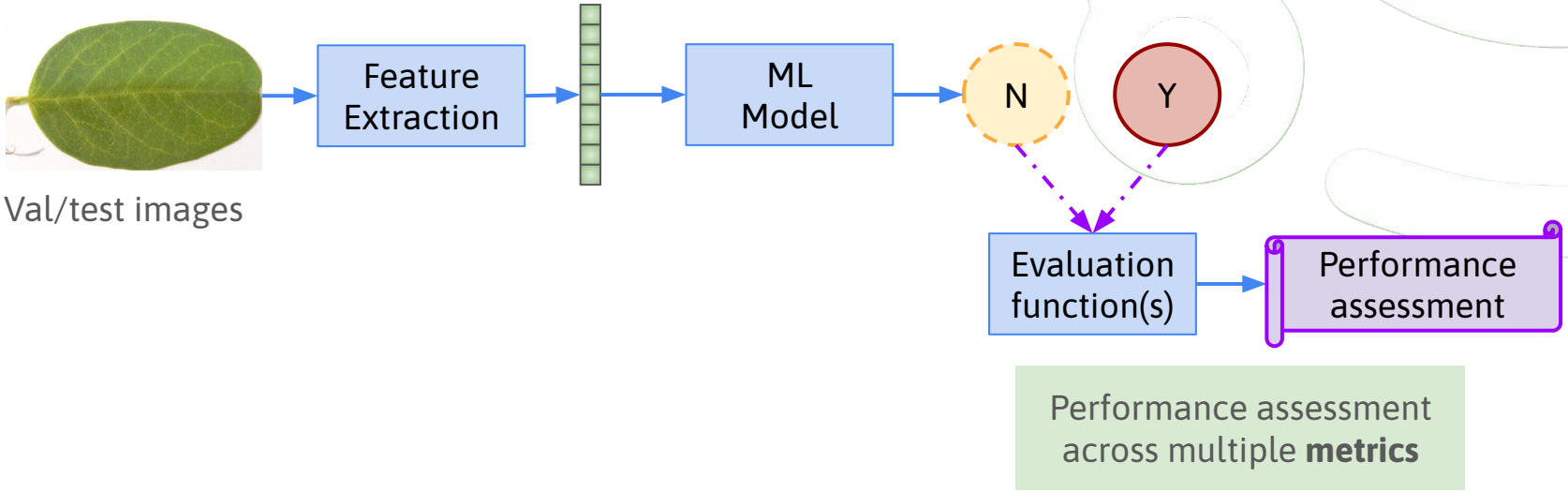
Cost/loss function

Low cost

The cost is used to update the **weights** of the model to help it improve

Supervised ML - Inference phase

Inference = making predictions and assessing performance (i.e. validation/testing)



Performance assessment across multiple **metrics**

Supervised ML - Inference phase

Inference = making predictions and assessing performance (i.e. validation/testing)

Val/test images

Feature Extraction

ML Model

N

Y

Evaluation function(s)

Performance assessment

No update loop (otherwise it would learn on val/test images)

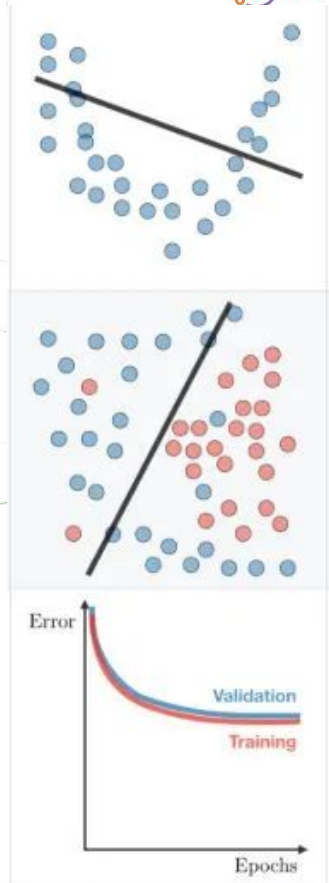
Performance assessment across multiple **metrics**

Supervised ML - Check your loss!

- Always compare your model's loss (and/or performance) on the training and validation data

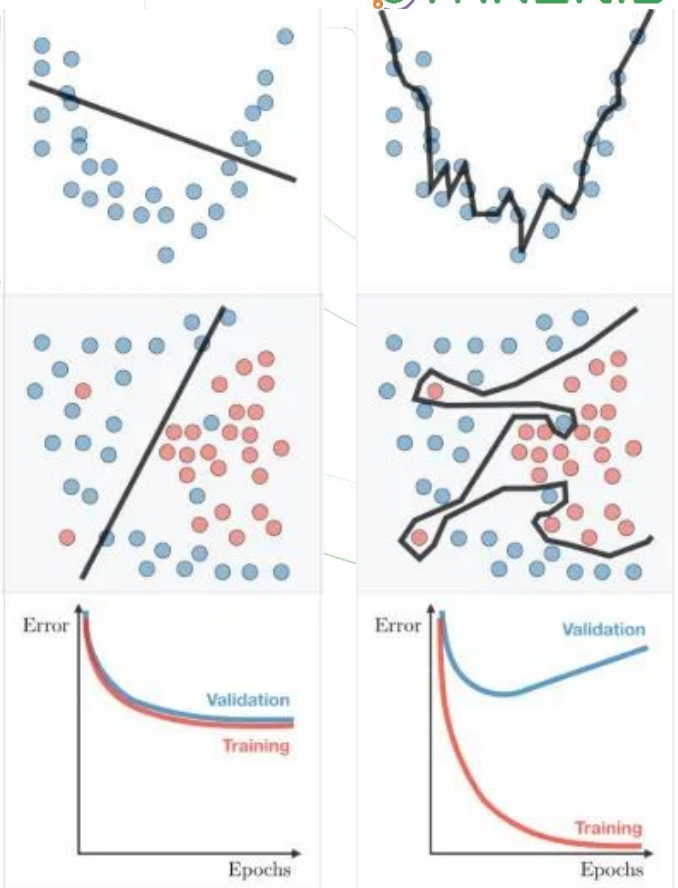
Supervised ML - Check your loss!

- Always compare your model's loss (and/or performance) on the training and validation data
- Things you want to look out for:
  - **Underfitting**: you are not learning enough! Your model is **too simple** and you have low performance on both training and validation data



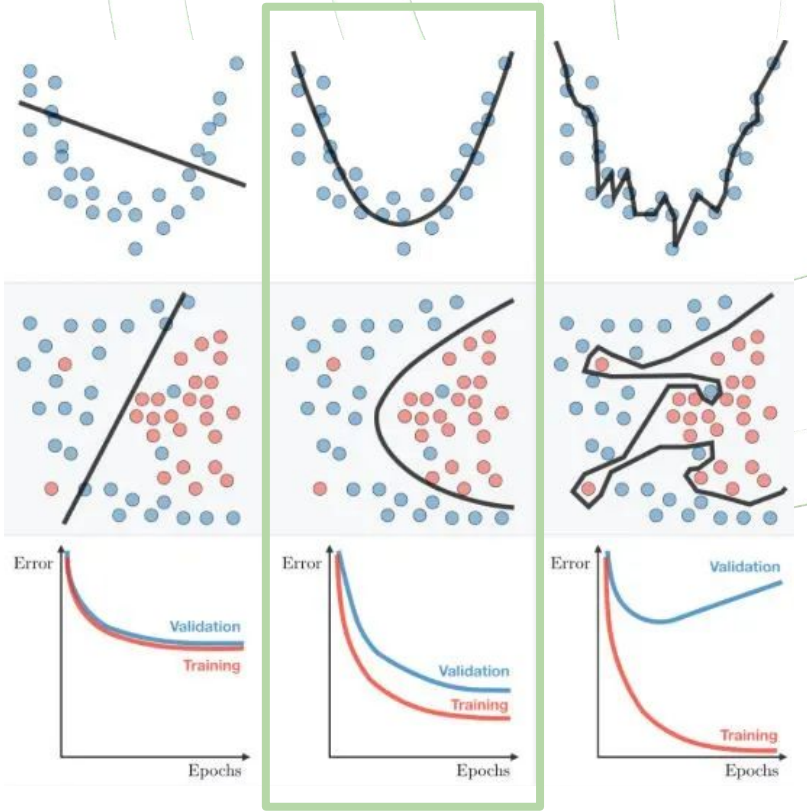
Supervised ML - Check your loss!

- Always compare your model's loss (and/or performance) on the training and validation data
- Things you want to look out for:
  - **Underfitting**: you are not learning enough! Your model is **too simple** and you have low performance on both training and validation data
  - **Overfitting**: you are focusing on details! You model is **too complex** and is "memorizing" the training set, including details which do not generalize on new data.



Supervised ML - Check your loss!

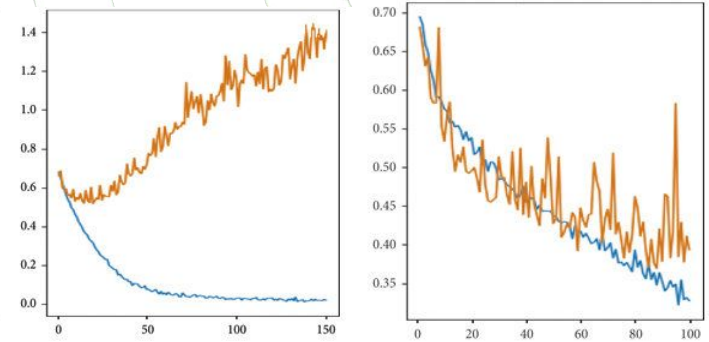
- Always compare your model's loss (and/or performance) on the training and validation data
- Things you want to look out for:
  - Underfitting
  - Overfitting
- Aim for a similar loss (and performance) on training data and validation data



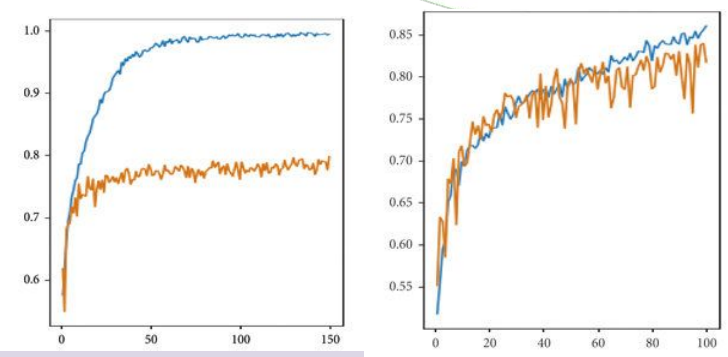
Supervised ML - Check your loss!

- Always compare your model's loss (and/or performance) on the training and validation data
- Things you want to look out for:
  - Underfitting
  - Overfitting
- Aim for a similar loss (and performance) on training data and validation data

Noticing overfitting looking at the loss



Noticing overfitting looking at the accuracy



Supervised ML tasks

Supervised ML - general tasks

Most supervised problems can be framed into one of these “general tasks”:

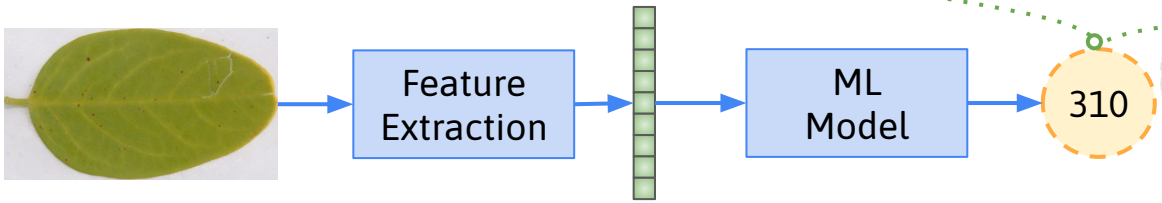
- Regression
- Classification
  - Binary classification
  - Multiclass classification
  - Multilabel classification

Regression

Supervised ML - Regression

Predictions are made in the **continuous domain**

(e.g.) it answers the question: "*what is the area (cm^2) of the leaf?*"



Intuitively: we want to quantify a quantity related to the input sample

Supervised ML - Regression

Predictions are made in the **continuous domain**

(e.g.) it answers the question: "*what is the area (cm^2) of the leaf?*"

Feature Extraction

ML Model

310

330

Cost/loss function

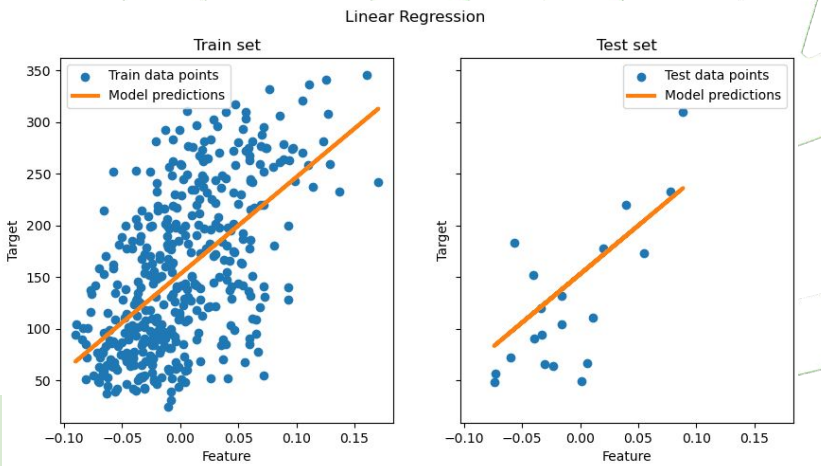
MAE/MSE loss, L1/L2 loss, Elastic (L1+L2)

Intuitively: we want to quantify a quantity related to the input sample

Regression - Ordinary Least Squares

Linear model with coefficients $w = [w_1, ..., w_p]$ (p = num of input features).

Minimizes the residual sum of squares between observed/correct targets and predictions, by linear approximation.



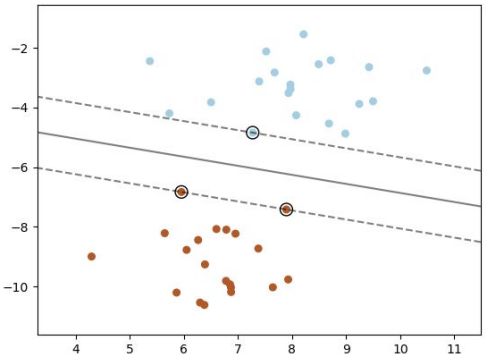
Intuitively: if 2D data points, with dimensions $[x_1, x_2]$, have values distributed around (e.g.) $2^*x_1 + 2.5^*x_2$, then OLS will try to find $w_1=2$, $w_2=2.5$

https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols_ridge.html

Regression - Support Vector Regression

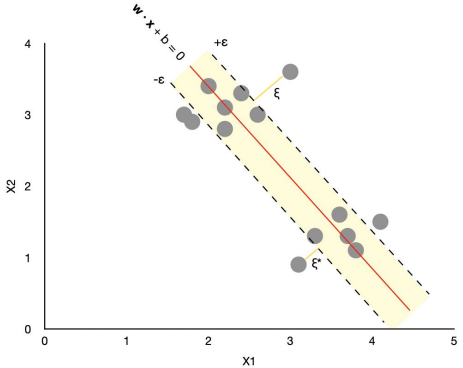
Derived from Support Vector Machines (SVM).

SV**M** searches for an hyperplane separating the training points into their respective classes, distancing them as much as possible.



SV**R** searches for an hyperplane predicting the expected values (ε-tube), while penalizing those falling outside such plane.

ε represents the tolerance to errors.



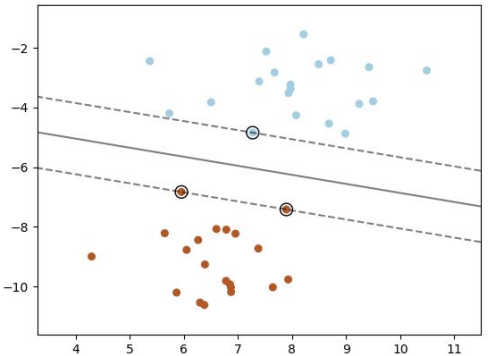
https://scikit-learn.org/stable/modules/svm.html
https://towardsdatascience.com/support-vector-regression-svr-one-of-the-most-flexible-yet-robust-prediction-algorithms-4d25fbdaca60/

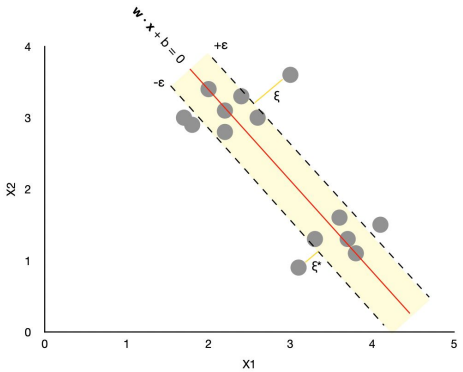
Regression - Support Vector Regression

SV**R** searches for an hyperplane predicting the expected values (ε-tube), while penalizing those falling outside such plane.

ε represents the tolerance to errors.

Intuitively: on 2D data points, it finds a “line” which captures all points **within** a certain range of error (ε)





https://scikit-learn.org/stable/modules/svm.html

https://towardsdatascience.com/support-vector-regression-svr-one-of-the-most-flexible-yet-robust-prediction-algorithms-4d25fbdaca60/

Regression - Task examples

Examples:

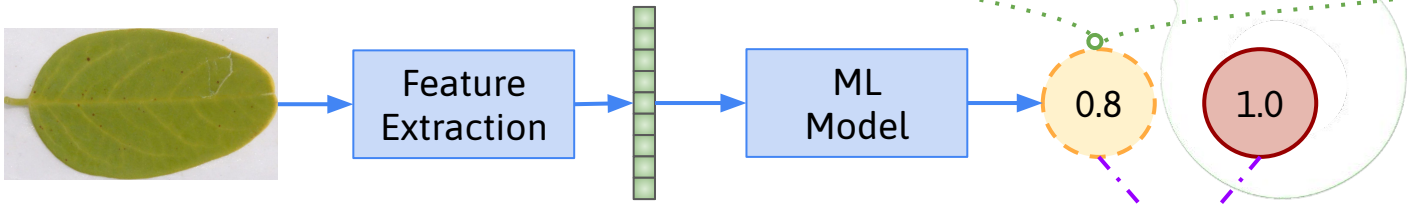
- Crop yield prediction (how much will this crop yield)
- Rainfall (how much will it rain)
- Plant coverage estimation (how much of the ground is covered by plants)
- Carbon storage/sequestration (how much carbon is stored in this plant)

Classification

Supervised ML - (Binary) Classification

Predictions are **confidence** values for a predefined class

(e.g.) it answers the question: "*is the leaf damaged?*" → YES or NO (binary)



Intuitively: we want to predict whether a specific class/property **is valid or not** for the sample

Cross Entropy

Supervised ML - Multiclass Classification

Predictions are **confidence** values for predefined classes (one class is correct)

(e.g.) it answers the question: "*what species is this plant?*"

Feature Extraction

ML Model

0.05	Fraxinus ornus
0.10	Quercus pubescens
...	...
0.75	Robinia pseudoacacia

Cost/loss function

Categorical Cross Entropy

Intuitively: given some fixed classes, we want to predict **which one** is valid for the sample

Supervised ML - Multilabel Classification

Predictions are **independent confidence** values for predefined classes (many can be valid)

(e.g.) it answers the question: "*which characteristics are relevant for this plant?*"

Feature Extraction

ML Model

0.05
0.20
0.55
...
0.70

White patina
Yellowing
Brown spots
...
Regular border

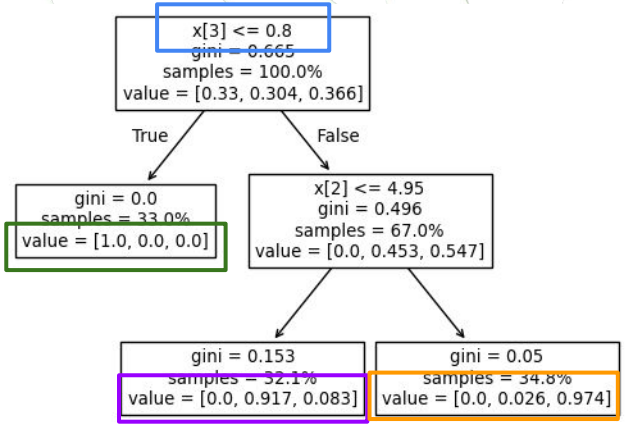
Cost/loss function

Binary Cross Entropy

Intuitively: given some fixed classes, we want to predict **all classes** valid for the sample

Classification - Decision Tree

DTs try to classify the input features by learning **simple decision rules** inferred from the data.

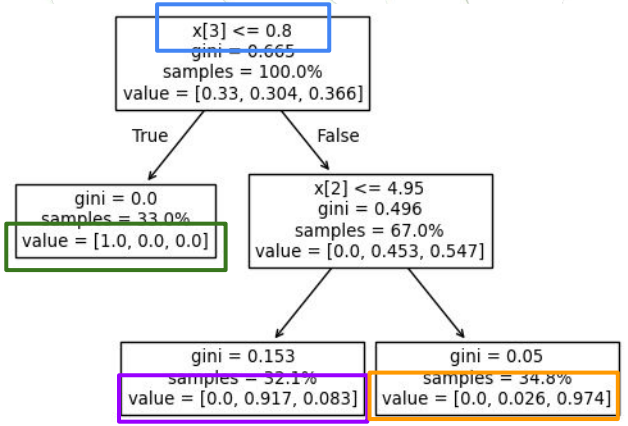


https://scikit-learn.org/stable/modules/tree.html

Classification - Decision Tree

DTs try to classify the input features by learning **simple decision rules** inferred from the data.

Intuitively: it learns a sequence of “ifs” to progressively separate the samples until a class becomes dominant.

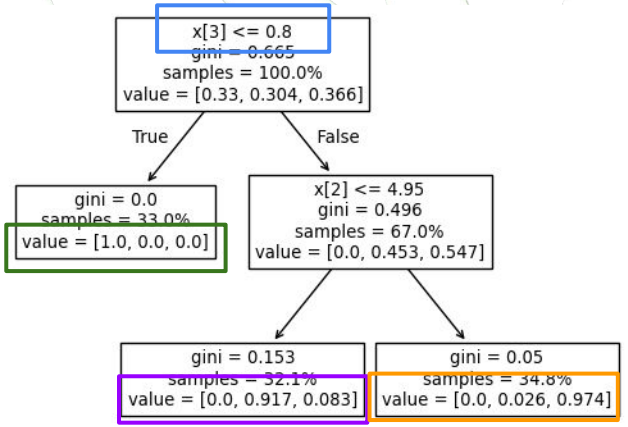


https://scikit-learn.org/stable/modules/tree.html

Classification - Decision Tree

DTs try to classify the input features by learning **simple decision rules** inferred from the data.

Intuitively: it learns a sequence of “ifs” to progressively separate the samples until a class becomes dominant.



E.g. input data (x) have four features, there are three possible classes [A,B,C]. This DT says that:

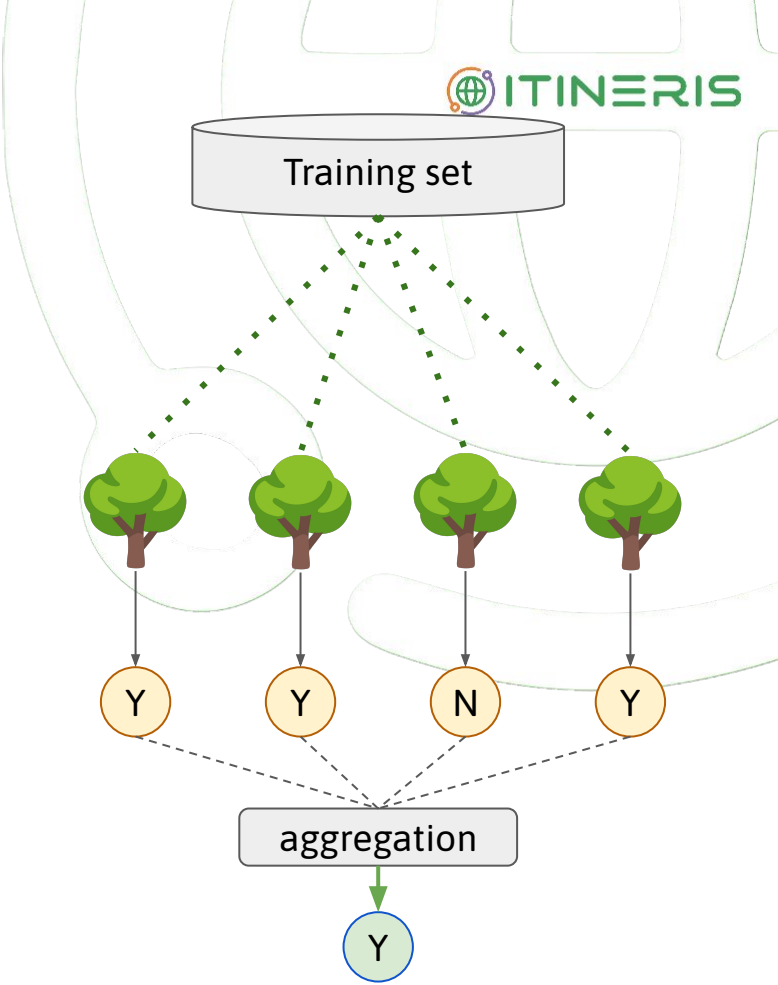
- A: **all** samples have x[3] <= 0.8
- B: 91.7% have x[3] > 0.8 and x[2] <= 4.95
- C: 97.4% have x[3] > 0.8 and x[2] > 4.95

https://scikit-learn.org/stable/modules/tree.html

Classification - Random Forest

Ensemble of DT.

Intuitively, instead of asking *one* model whether the kiwi is ripe or not, we ask it to *tens or hundreds* of models and then pool the answers.

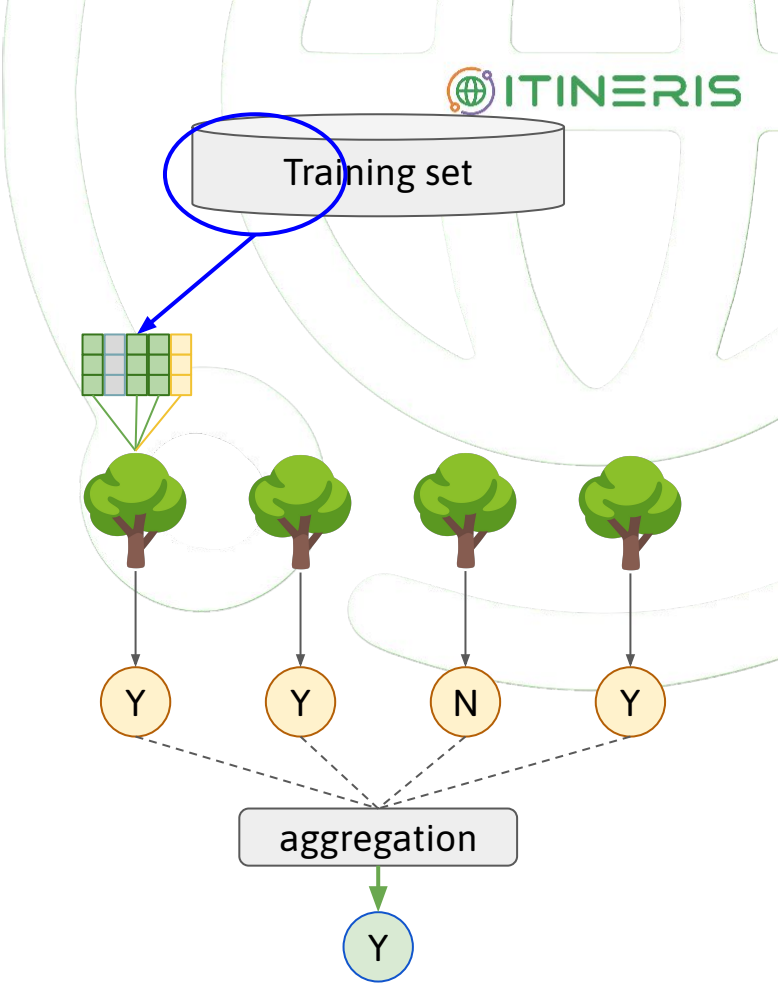


Classification - Random Forest

Ensemble of DT.

Intuitively, instead of asking *one* model whether the kiwi is ripe or not, we ask it to *tens or hundreds* of models and then pool the answers.

Two source of randomness (data, features) to reduce variance (sensitiveness to data) at the cost of increased bias (simplicity of model), improving robustness.

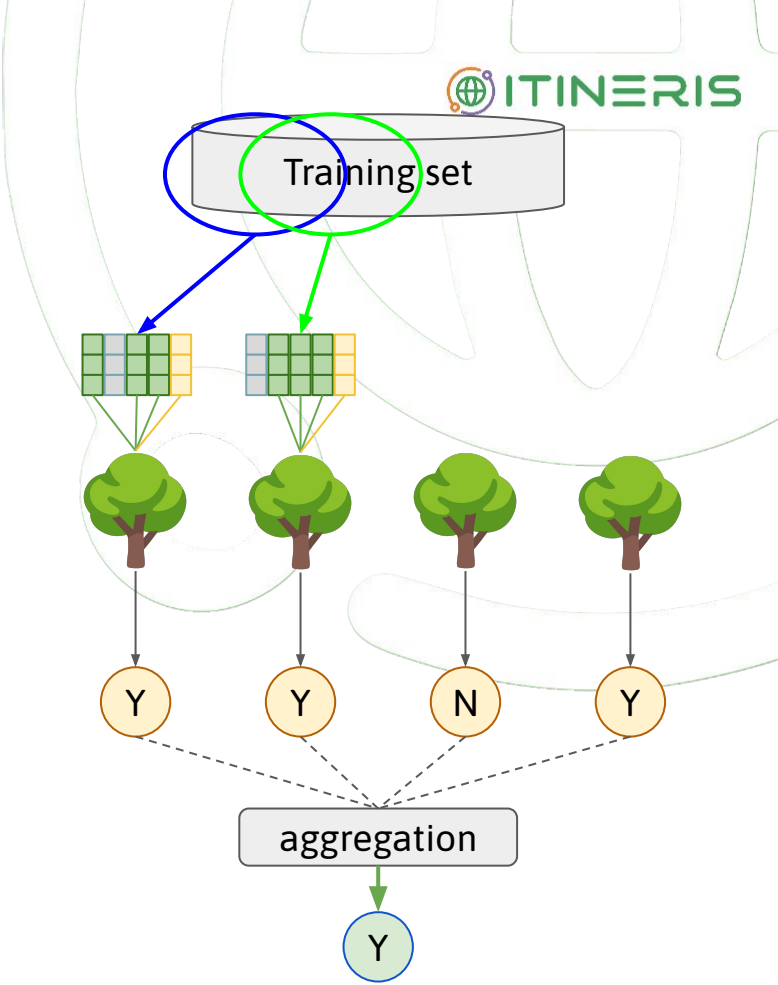


Classification - Random Forest

Ensemble of DT.

Intuitively, instead of asking *one* model whether the kiwi is ripe or not, we ask it to *tens or hundreds* of models and then pool the answers.

Two source of randomness (data, features) to reduce variance (sensitiveness to data) at the cost of increased bias (simplicity of model), improving robustness.

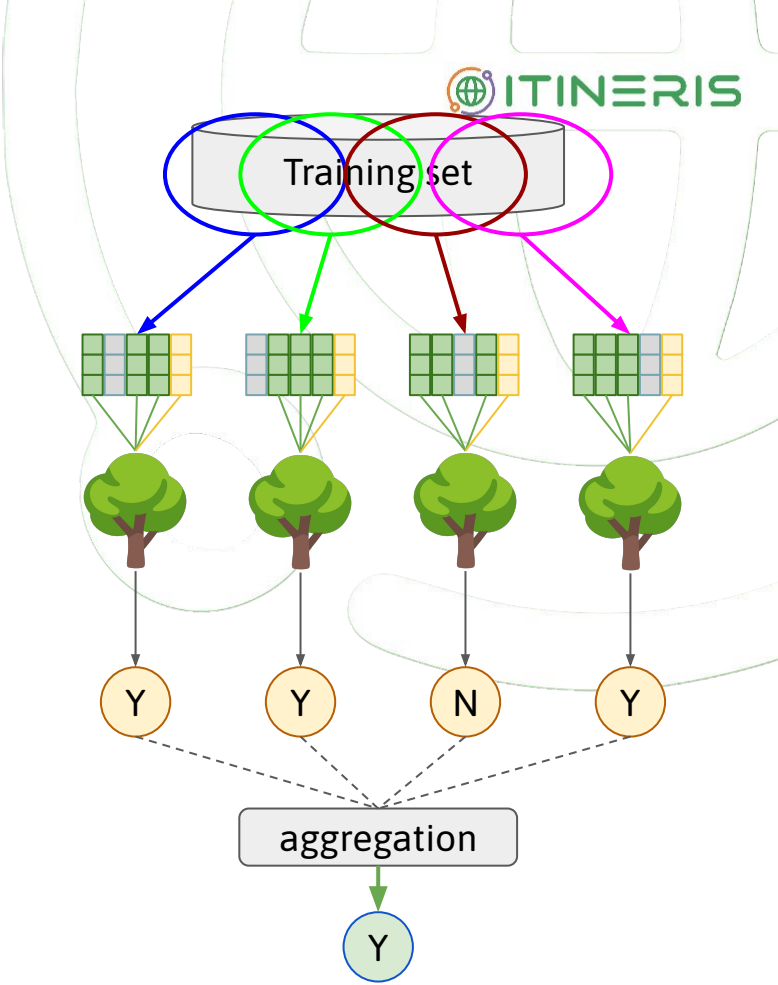


Classification - Random Forest

Ensemble of DT.

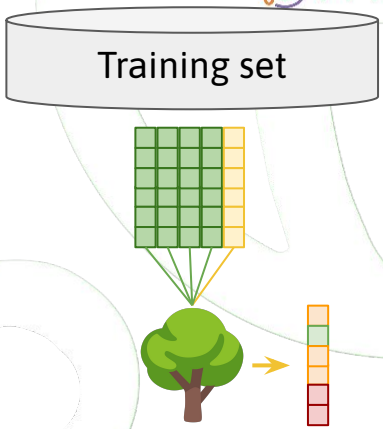
Intuitively, instead of asking *one* model whether the kiwi is ripe or not, we ask it to *tens or hundreds* of models and then pool the answers.

Two source of randomness (data, features) to reduce variance (sensitiveness to data) at the cost of increased bias (simplicity of model), improving robustness.



Classification - Gradient Boosting

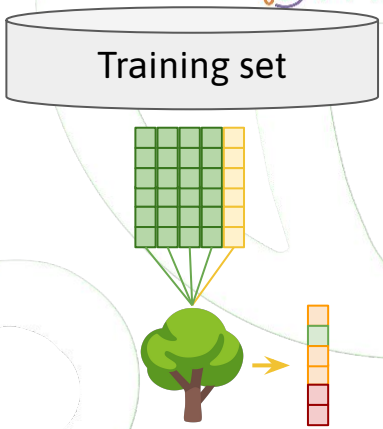
Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.



Classification - Gradient Boosting

Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.

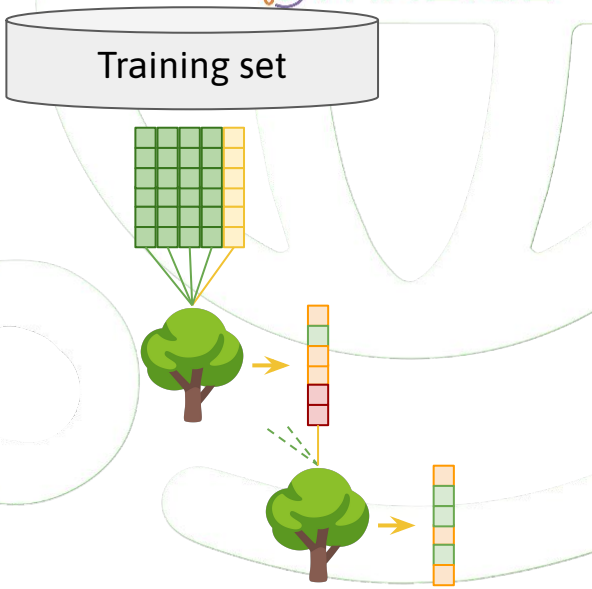
Intuitively: 1st DT learns an initial solution, yet still makes some error on each sample; 2nd DT is trained to adjust these *residual* errors. This process goes on iteratively, improving little by little.



Classification - Gradient Boosting

Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.

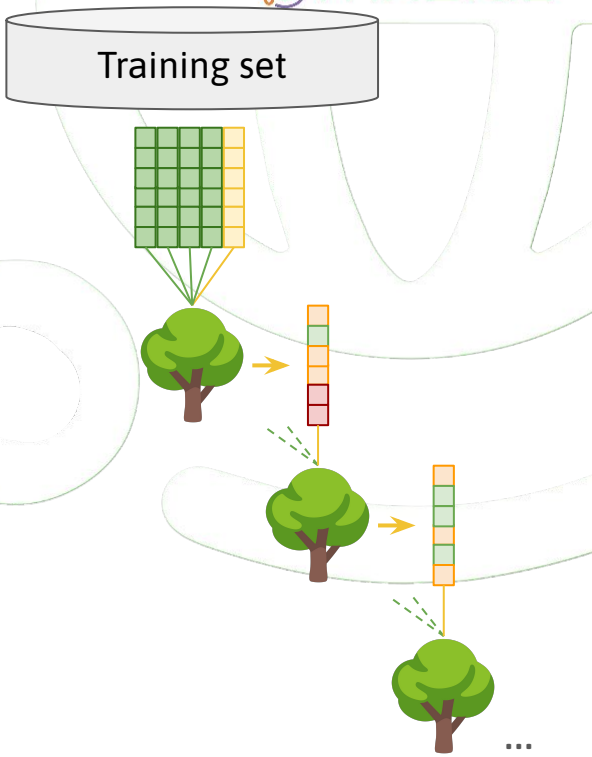
Intuitively: 1st DT learns an initial solution, yet still makes some error on each sample; 2nd DT is trained to adjust these *residual* errors. This process goes on iteratively, improving little by little.



Classification - Gradient Boosting

Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.

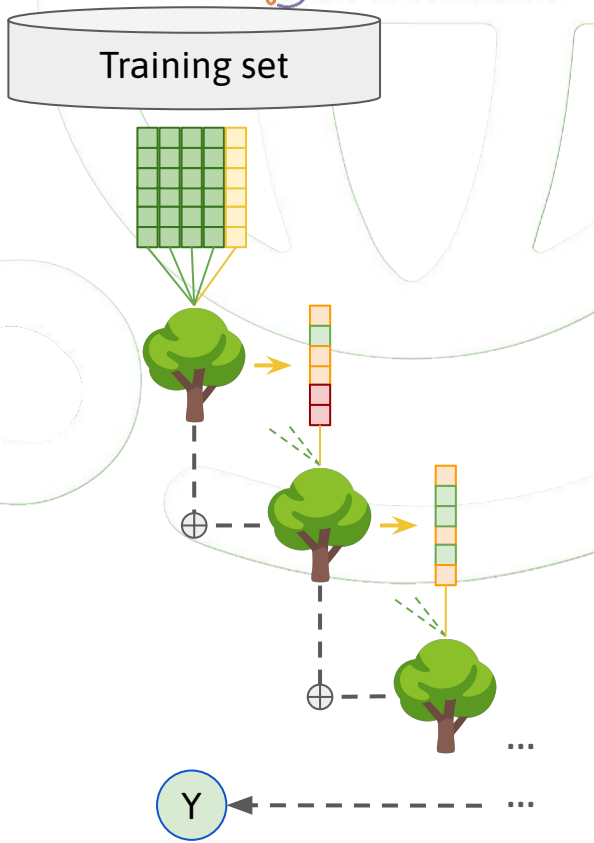
Intuitively: 1st DT learns an initial solution, yet still makes some error on each sample; 2nd DT is trained to adjust these *residual* errors. This process goes on iteratively, improving little by little.



Classification - Gradient Boosting

Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.

Intuitively: 1st DT learns an initial solution, yet still makes some error on each sample; 2nd DT is trained to adjust these *residual* errors. This process goes on iteratively, improving little by little.

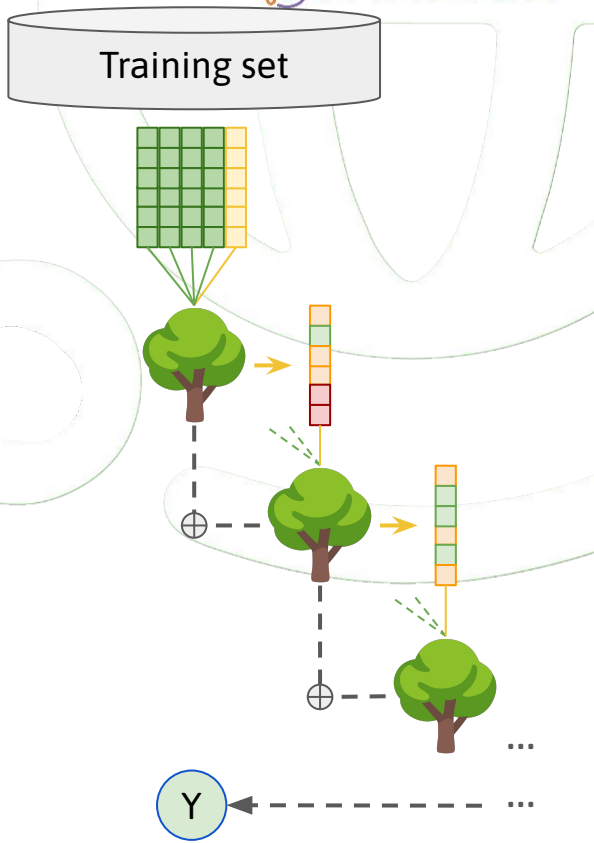


Classification - Gradient Boosting

Ensemble (on DT). Key difference with RF: **sequence of DT** and *not DTs in parallel*.

Intuitively: 1st DT learns an initial solution, yet still makes some error on each sample; 2nd DT is trained to adjust these *residual* errors. This process goes on iteratively, improving little by little.

As for RFs, “weak learners” (here, DT) are built on a subsample of training to introduce randomness and reduce overfitting.



Classification - Task examples

Examples:

- Fish species classification
- Crop disease classification
- Food type recognition
- Pixel-level terrain use classification

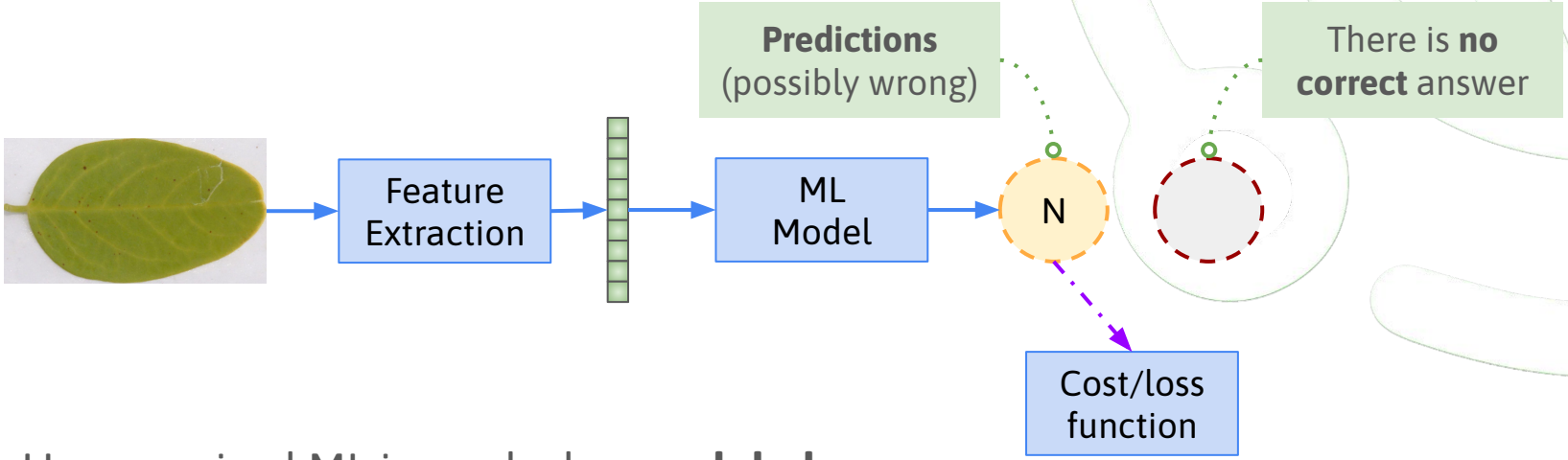
Supervised ML - A recap

- **Regression**
  - We want to *predict a numerical value* of some kind
  - E.g. crop yield prediction, carbon sequestered in trees, calories estimation from food image, ...

- Classification
  - **Binary** classification → whether the sample is *a member of the class or not*
    - E.g. healthy/unhealthy leaves, is the kiwi ripe or not, ...
  - **Multiclass** classification → to *which one of some fixed classes* does my sample belong to
    - E.g. recognizing animal species, leaf species, which *disease* is affecting this plant, ...
  - **Multilabel** classification → identify *all classes* to which my samples belongs to
    - E.g. which of these characteristics are valid for this animal, which *disease**s** are* affecting this plant, ...

Unsupervised ML:
data → ??? → insights

Unsupervised ML - The lack of supervision



Unsupervised ML is used when **no labels** are available.

Unsupervised ML - The lack of supervision

If no labels are available, for what do we use it?

Useful for:

- Exploring the data, and
- Finding patterns/groups/hidden structures

A positive note: as labels are often created by humans, they are sometimes **biased**.

- Imprecise identification of plant species, phenological stage, ...;
- “Subjective” evaluation (e.g. differs with different annotators, even experts);
- Different labeling style over the years (e.g. taxonomy reclassification)
- ...

Unsupervised ML - General problems

Three main “classes” of problems for unsupervised ML:

- Clustering
- Manifold learning
- Dimensionality reduction
- Outlier/novelty detection

Clustering

Unsupervised ML - Clustering

Main idea: automatic discovery of **data groupings**.

If some data points end up in the same cluster, then they are “similar”, i.e. they share some common features. So, we should ask ourselves:

- Why are they “similar”?
- Is there some commonality I hadn’t noticed before?
- Do they belong to same “class”? (e.g. same species)
- Does this relation tell me something about potential “classes” in my field?

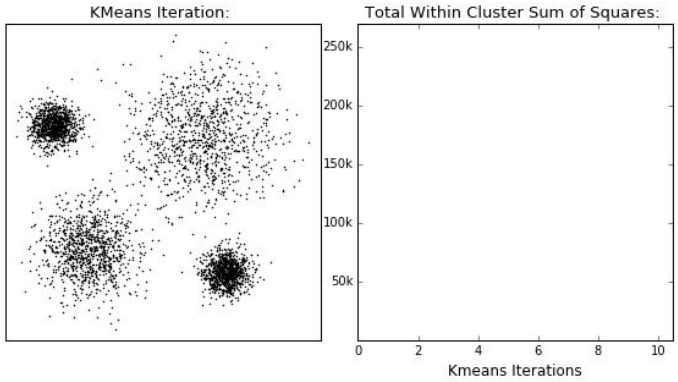
Especially interesting when dealing with **multivariate** samples, as relations between points are likely more difficult to grasp for a human.

Clustering - K-Means

Given *k*, it tries to find **k clusters** within the data by iteratively updating their **centroids** (i.e. mean of points in the cluster).

Clustering - K-Means

Given *k*, it tries to find **k clusters** within the data by iteratively updating their **centroids** (i.e. mean of points in the cluster).

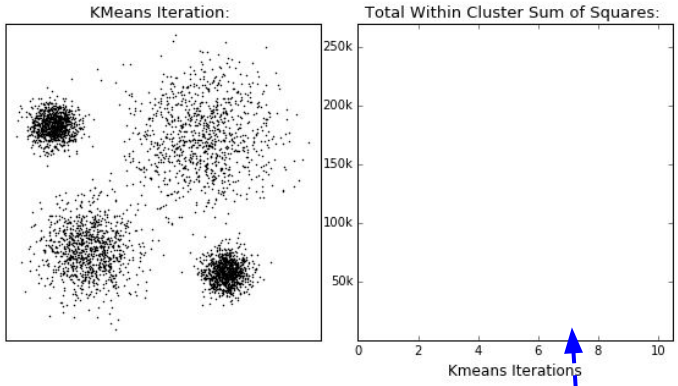


1) *Randomly pick* k initial “centroids”.
2) *Assign* the points to the *nearest* centroid.
3) *Re-compute* the centroids.
4) *Repeat* from step 2 (until convergence).

Clustering - K-Means

Given *k*, it tries to find **k clusters** within the data by iteratively updating their **centroids** (i.e. mean of points in the cluster).

1) *Randomly pick* k initial “centroids”.
2) *Assign* the points to the *nearest* centroid.
3) *Re-compute* the centroids.
4) *Repeat* from step 2 (until convergence).



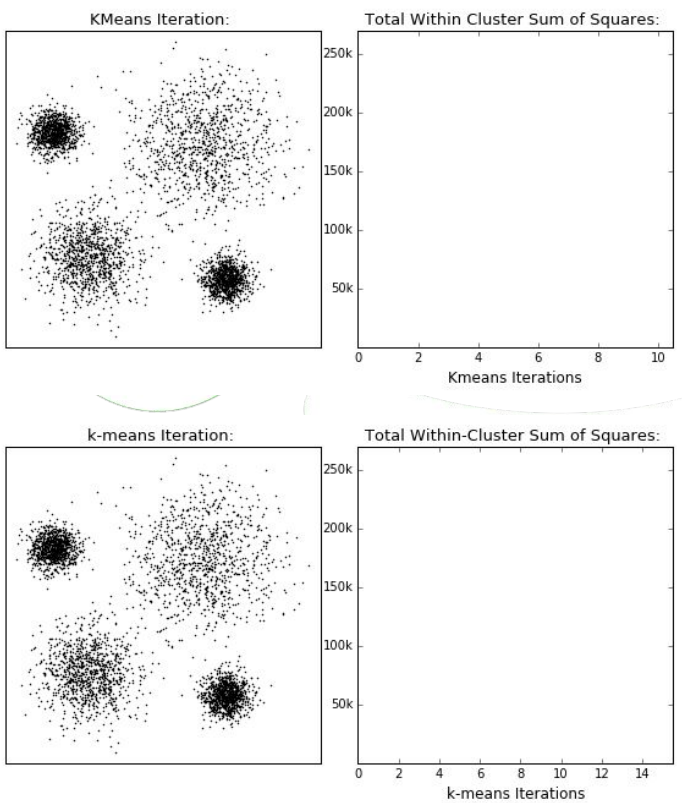
Convergence → this value here stops decreasing

Clustering - K-Means

Given *k*, it tries to find **k clusters** within the data by iteratively updating their **centroids** (i.e. mean of points in the cluster).

1) *Randomly pick* k initial "centroids".
2) *Assign* the points to the *nearest* centroid.
3) *Re-compute* the centroids.
4) *Repeat* from step 2 (until convergence).

K-Means is fairly sensitive to outliers and **initial conditions.** → run multiple times with random initial centroids, or use k-means++ ("smarter" selection of initial points).



Unsupervised ML - Dimensionality reduction

2D/3D data can be **plotted/visualized**, what about **4D+** data?

Unsupervised ML - Dimensionality reduction

2D/3D data can be **plotted/visualized**, what about **4D+** data?

Random projections? → likely losing interesting structures

Unsupervised ML - Dimensionality reduction

2D/3D data can be **plotted/visualized**, what about **4D+** data?

Random projections? → likely losing interesting structures

Common technique: Principal Component Analysis (**PCA**).

It decomposes high-dimensional signals into **linear combinations** of features that "explain" the max amount of variance.

Unsupervised ML - Dimensionality reduction

2D/3D data can be **plotted/visualized**, what about **4D+** data?

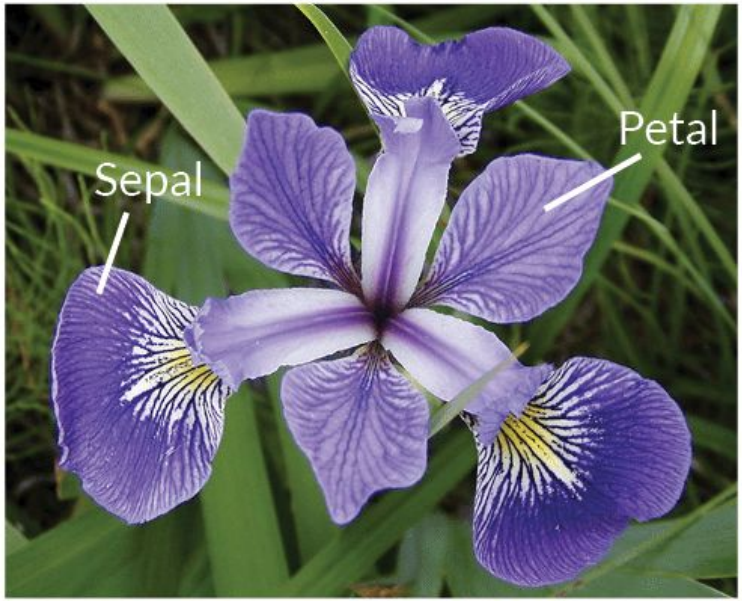
Random projections? → likely losing interesting structures

Common technique: Principal Component Analysis (**PCA**).

It decomposes high-dimensional signals into **linear combinations** of features that "explain" the max amount of variance.

Intuitively: it searches for "new axes" that explain the variance in the data. These axes are searched iteratively and are progressively less "important".

Dimensionality reduction - Iris dataset

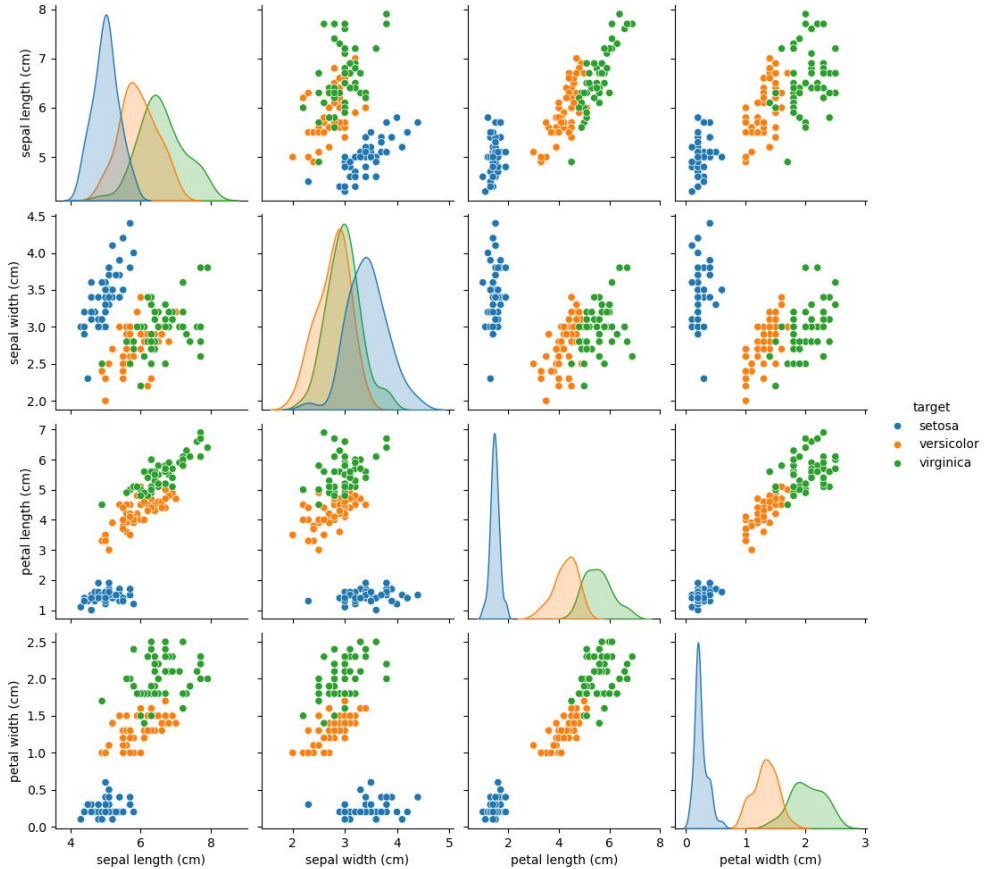


Iris Versicolor

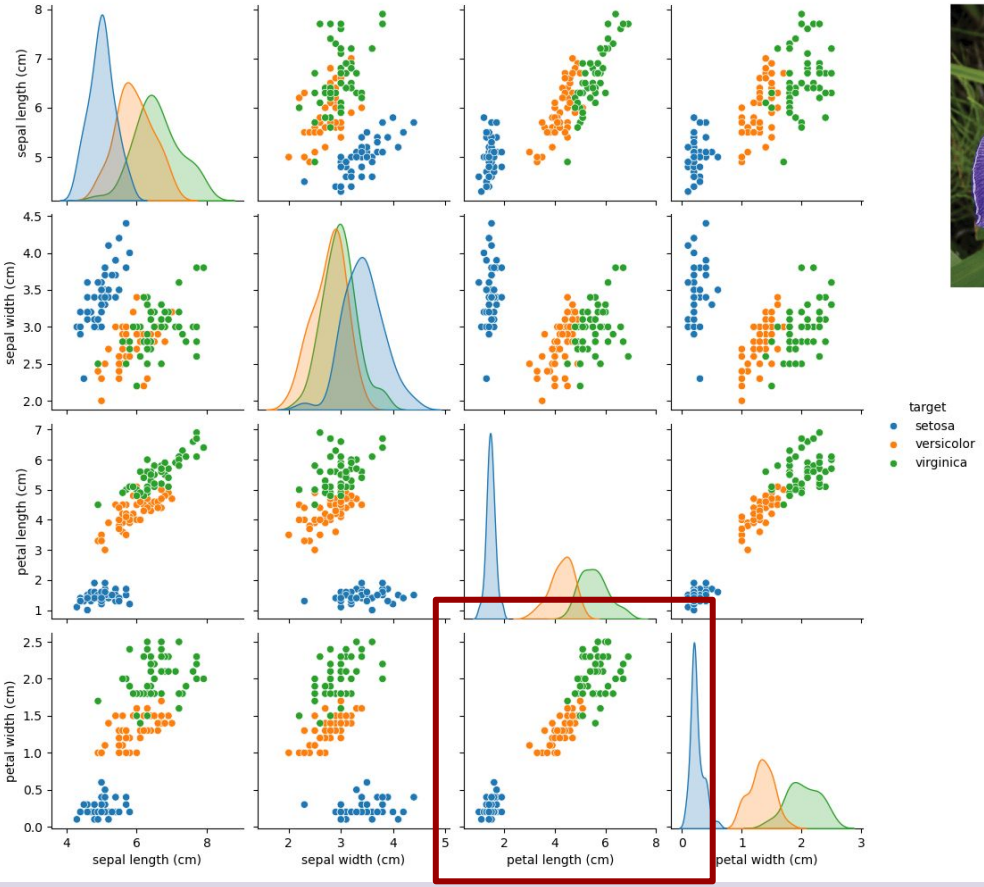
Iris Setosa

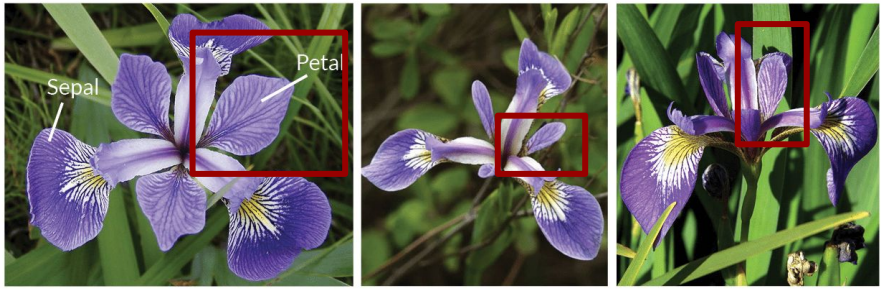
Iris Virginica

Dimensionality reduction - PCA example



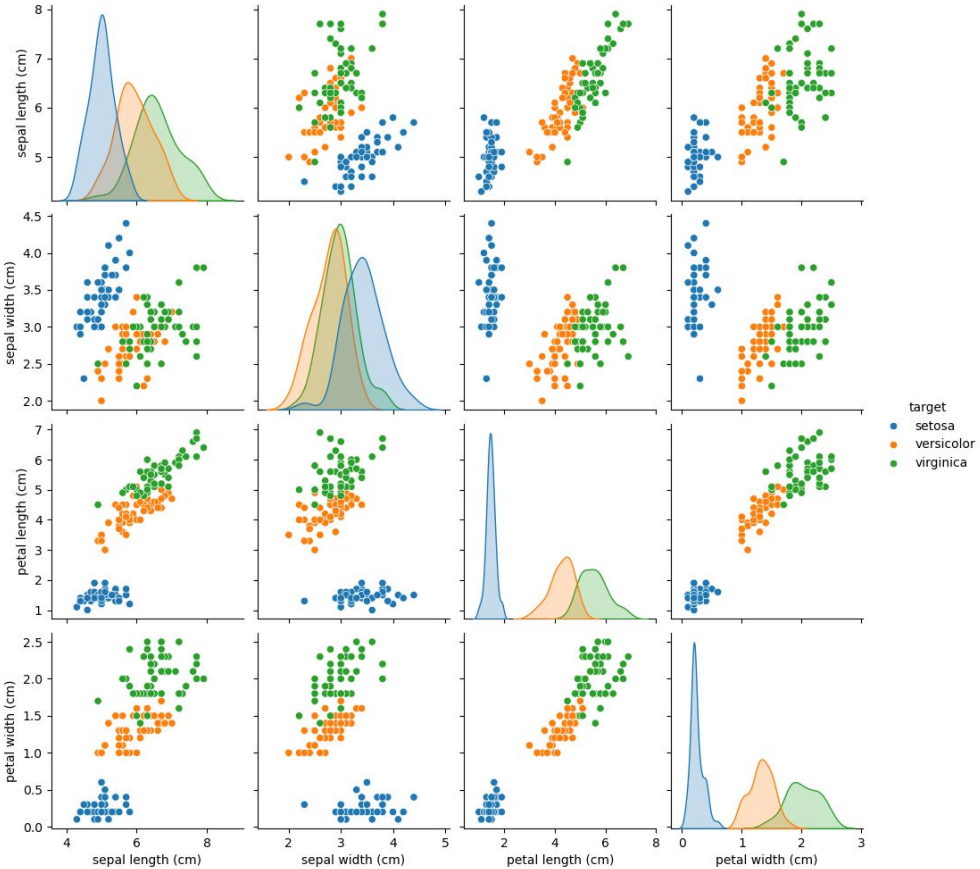
Dimensionality reduction - PCA example

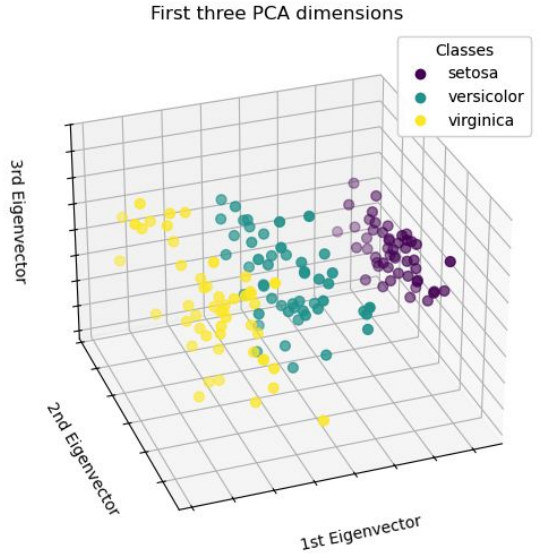




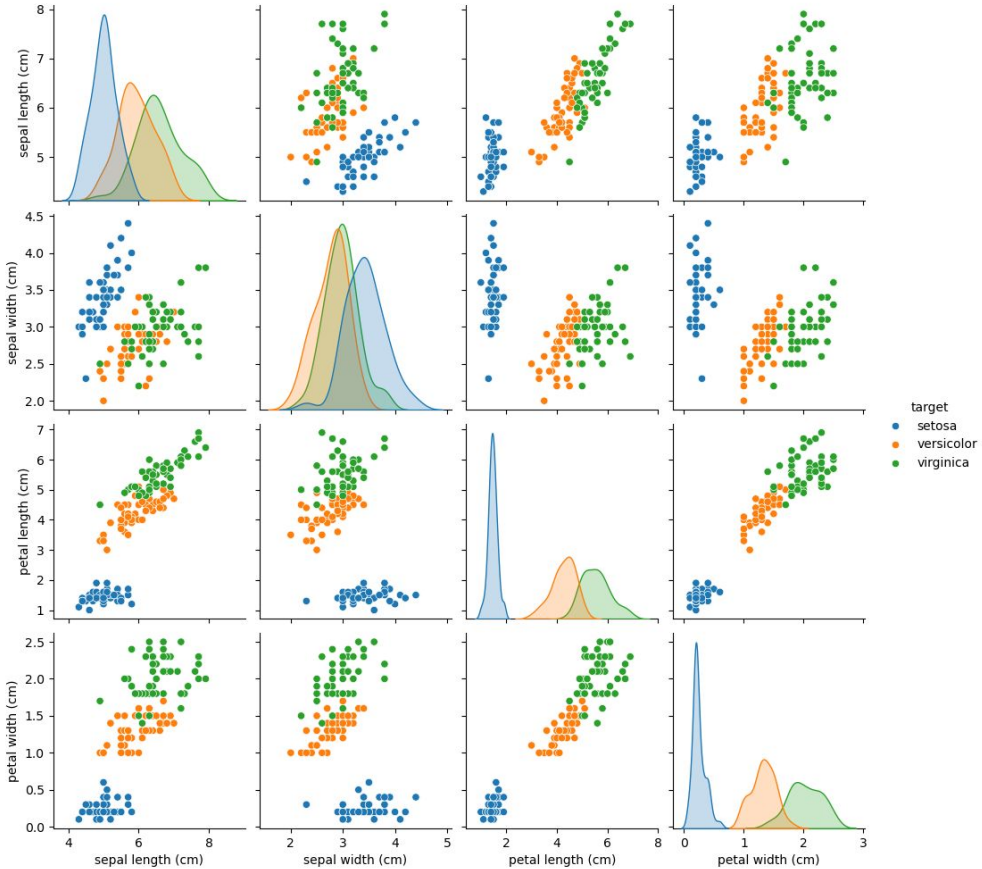
Iris Versicolor **Iris Setosa** **Iris Virginica**

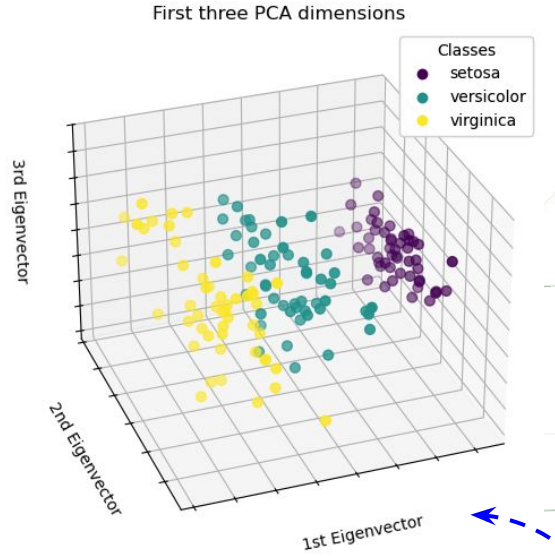
Dimensionality reduction - PCA example





Dimensionality reduction - PCA example





Components found by PCA:

0.36 * SL - 0.08 * SW + 0.86 * PL + 0.36 * PW

0.66 * SL + 0.73 * SW - 0.17 * PL - 0.07 * PW

-0.58 * SL + 0.60 * SW + 0.08 * PL + 0.55 * PW

Explaining 92.5%, 5.3%, 1.7% of the total variance.

Unsupervised ML - Manifold learning

A drawback of PCA is that it misses important **nonlinear** structure in the data.

Manifold learning attempts at obtaining a “nonlinear-sensitive” PCA.

Unsupervised ML - Manifold learning

A drawback of PCA is that it misses important **nonlinear** structure in the data.

Manifold learning attempts at obtaining a “nonlinear-sensitive” PCA.

Most common technique: t-distributed Stochastic Neighbor Embedding **(t-SNE)**

Intuitively: t-SNE focuses on “**who is close to whom**” in the original, high-dimensional data. It then learns a map into a low-dimensional embedding space (2D/3D) preserving those **neighborhood relations**.

Unsupervised ML - Manifold learning

A drawback of PCA is that it misses important **nonlinear** structure in the data.

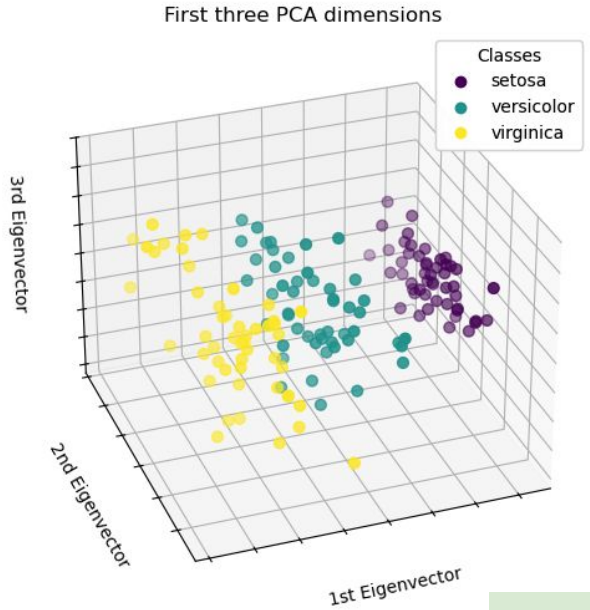
Manifold learning attempts at obtaining a “nonlinear-sensitive” PCA.

Most common technique: t-distributed Stochastic Neighbor Embedding **(t-SNE)**

Intuitively: t-SNE focuses on “**who is close to whom**” in the original, high-dimensional data. It then learns a map into a low-dimensional embedding space (2D/3D) preserving those **neighborhood relations**.

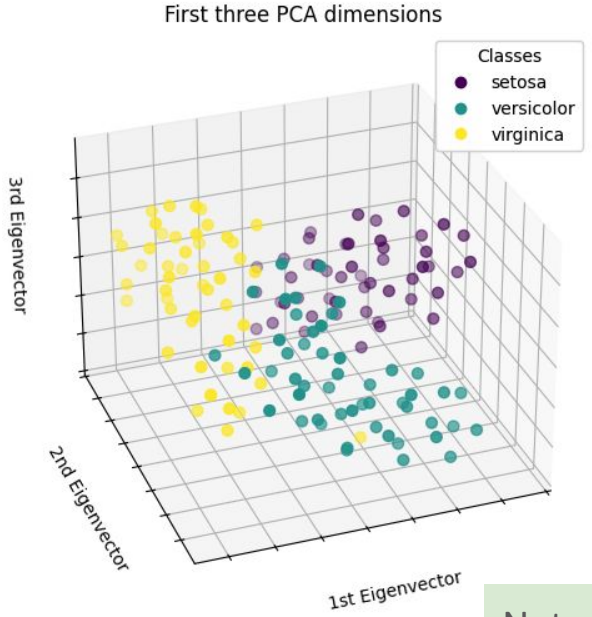
Sad note: it’s slow compared to PCA, especially on large datasets (e.g. millions of points → maybe hours, whereas it’s seconds/minutes with PCA).

Manifold learning - t-SNE example (iris)



PCA

Clear distinction

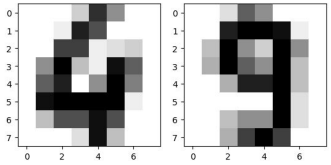


t-SNE

Not so clear distinction

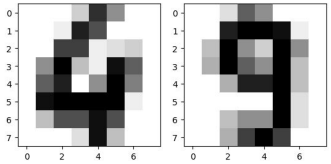
Manifold learning - t-SNE example (digits)

“easy”

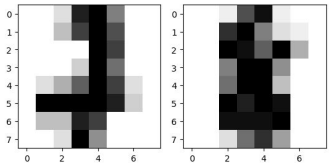


Manifold learning - t-SNE example (digits)

“easy”

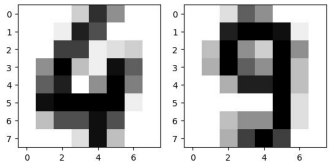


“hard”

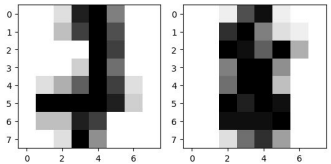


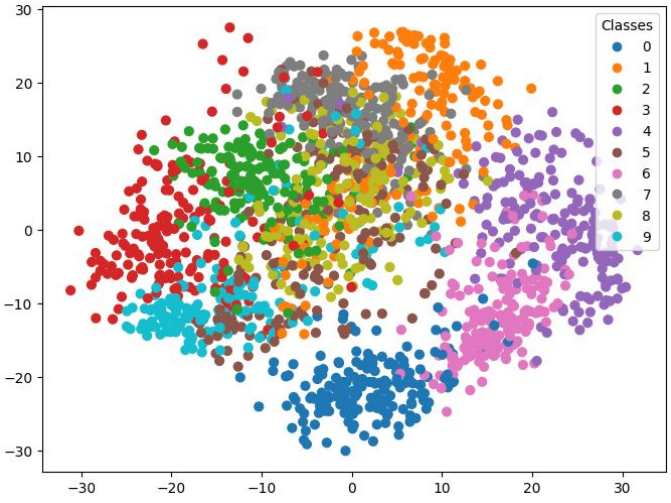
Manifold learning - t-SNE example (digits)

“easy”

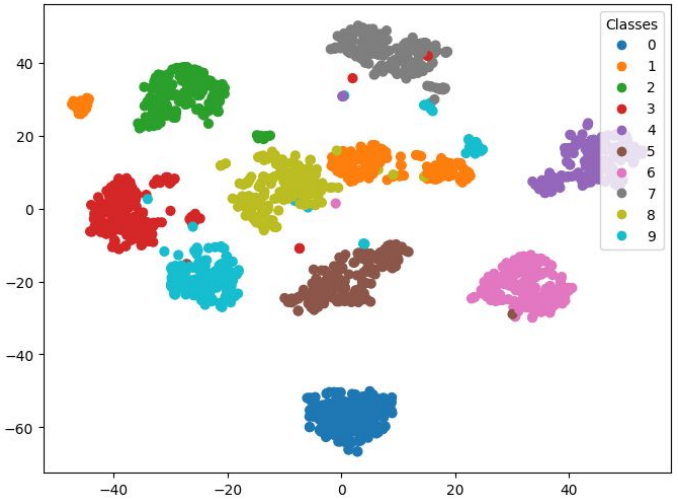


“hard”





PCA



t-SNE

Unsupervised ML - Outlier/novelty detection

Intuitively: the model is trained to recognize “normal” points, and thus “abnormal” ones are considered anomalous.

Unsupervised ML - Outlier/novelty detection

Intuitively: the model is trained to recognize “normal” points, and thus “abnormal” ones are considered anomalous.

Related problems: anomalies, outliers, novelties.

- Outlier det.: identify points *within* training data that are far outside the distribution of the other samples (i.e. training polluted with outliers)
- Novelty det.: determine if points *outside* the *non-polluted* training data are outliers (i.e. training data = only “normal” data)
- Anomaly det.: like novelty det., but “anomaly” is more popular than “novelty”

Novelty detection (e.g.) - One Class SVM

Start from *n* observations from the same distribution.

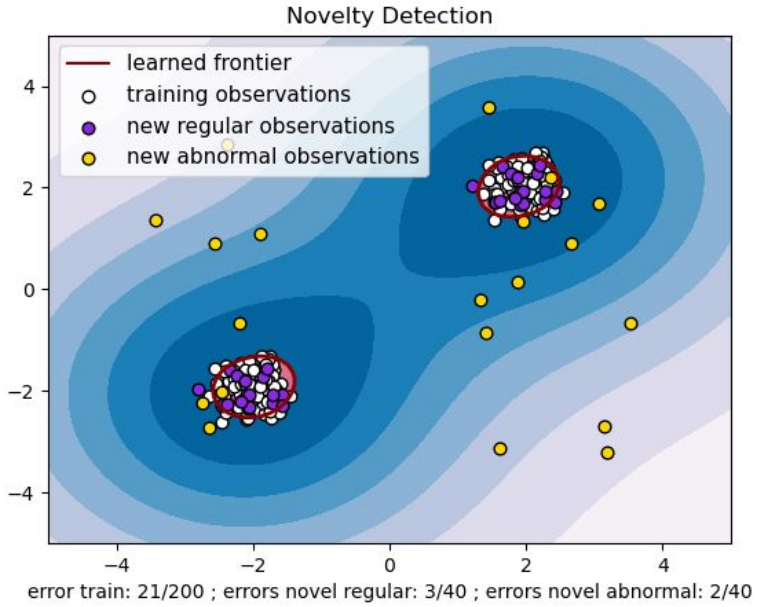
A new observation o arrives. Is it *so different* from the others that we doubt it's from the same distribution?

Novelty detection (e.g.) - One Class SVM

OneClassSVM learns a *frontier* delimiting the contour of the initial observations. Two cases:

- o falls *inside* the frontier → same distribution
- o falls *outside** the frontier → anomaly/novelty

* outside with “confidence” (i.e. there is a tolerance parameter)



Dealing with Data

Data - different types

Different kinds of data – but the model only sees numbers!

Machine learning models only understand numbers 🤖

But our real-world data are heterogeneous:

- Numerical (temperature, pH, precipitation, ...)
- Categorical (soil type, plant species, disease name, ...)
- Cyclical (month, time of day, slope orientation, wind direction, ...)
- Long texts (disease description, treatment plans, ...)

Data - encoding different types

- Numerical
- Categorical
- Cyclical
- Long texts

Year

Kind of crop
(corn, dwht, wwht, soyb, sgbt)

Crop rotation
(1,2,3,4)

Tilling method
(CA, CV, NT, TR)

Name of the site

Yearly crop yield

Sow month and harvest month

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Data - encoding

- Numerical
  - Already fine, we can just use them as they are (for the most part)

- Categorical
- Cyclical
- Long texts

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Data - encoding

- Numerical
- Categorical
  - **One-Hot Encoding**
    Turns categories into binary columns
    ⚠️ Can create **many** columns
    if we have too many categories

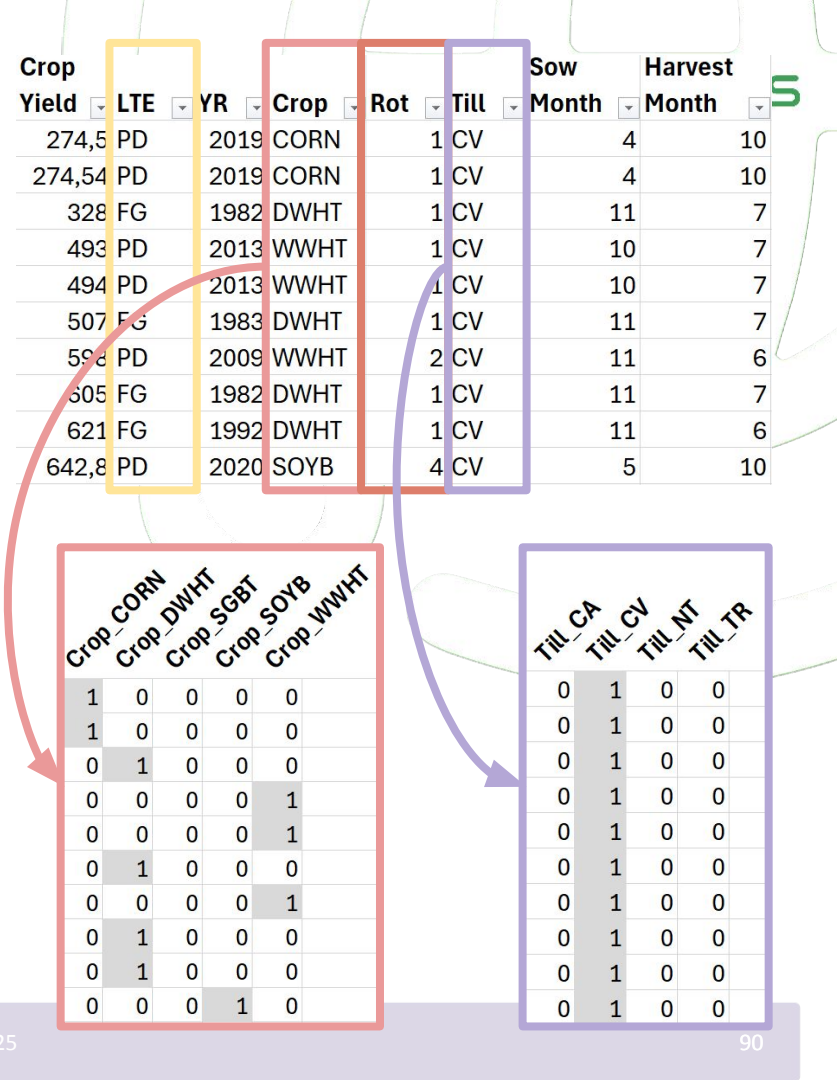
  - **Label Encoding**
    Assigns numbers to categories
    ⚠️ Danger: model might assume ordering!

- Cyclical
- Long texts

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Data - encoding

- Numerical
- Categorical
  - **One-Hot Encoding**
    Turns categories into binary columns
    ⚠️ Can create **many** columns
    if we have too many categories
  - **Label Encoding**
    Assigns numbers to categories
    ⚠️ Danger: model might assume ordering!
- Cyclical
- Long texts



Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
59[illegible]	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Crop_CORN	Crop_DWHT	Crop_SGBT	Crop_SOYB	Crop_WWHT
1	0	0	0	0
1	0	0	0	0
0	1	0	0	0
0	0	0	0	1
0	0	0	0	1
0	1	0	0	0
0	0	0	0	1
0	1	0	0	0
0	1	0	0	0
0	0	0	1	0

Till_CA	Till_CV	Till_NT	Till_TR
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0
0	1	0	0

Data - encoding

- Numerical
- Categorical
  - **One-Hot Encoding**
    Turns categories into binary columns
    ⚠️ Can create **many** columns
    if we have too many categories
  - **Label Encoding**
    Assigns numbers to categories
    ⚠️ Danger: model might assume ordering!
- Cyclical
- Long texts

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Rot
1
1
1
1
1
1
2
1
1
4

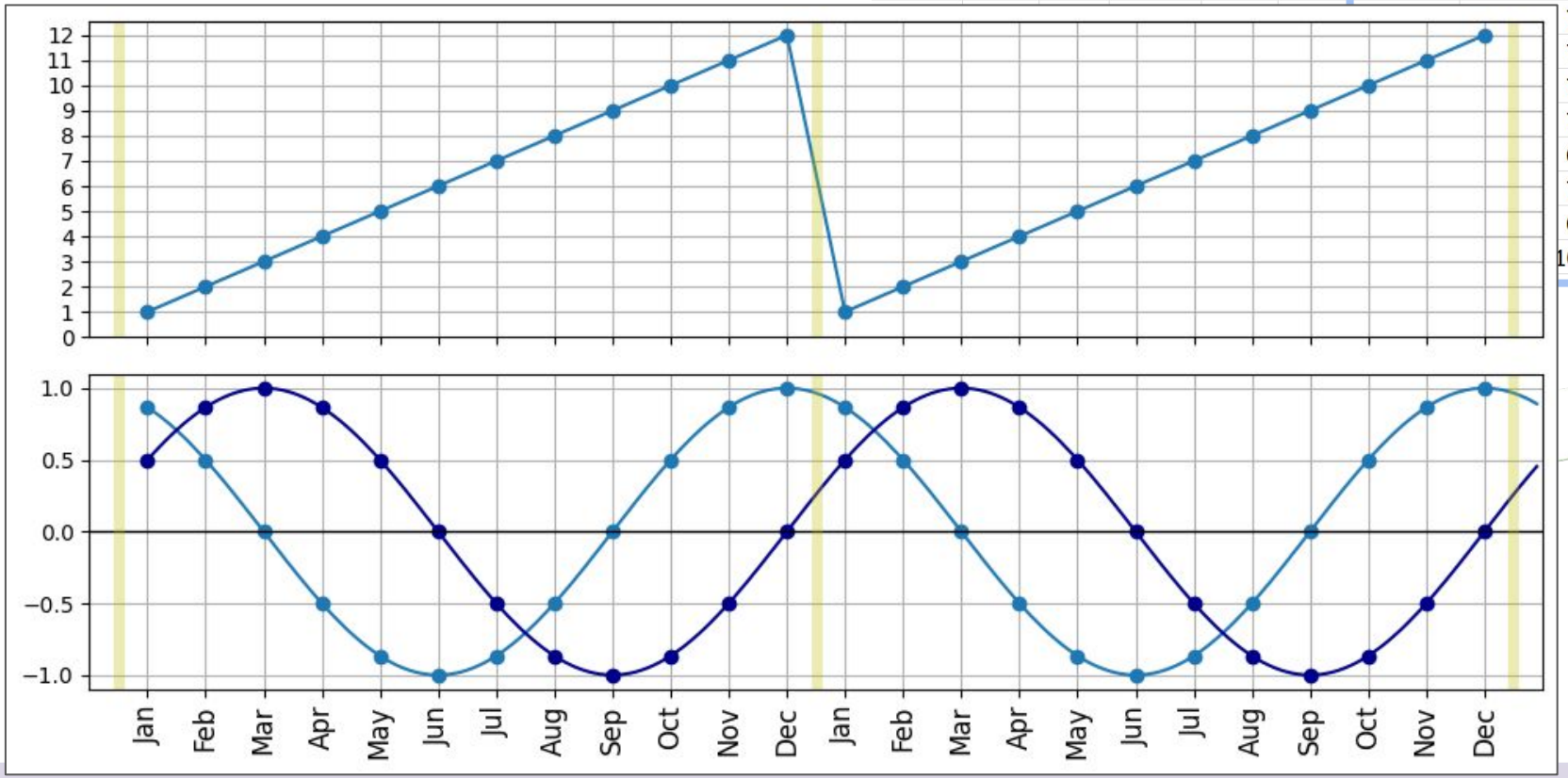
Data - encoding

- Numerical
- Categorical
- Cyclical
  - Map all data (e.g. months) to angles
  - Calculate sine and cosine
  - Now the start and the end of the range are “close together”
- Long texts

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

Data - encoding

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10



Data - encoding

- Numerical
- Categorical
- Cyclical
  - Map all data (e.g. months) to angles
  - Calculate sine and cosine
  - Now the start and the end of the range are “close together”
- Long texts

Crop Yield	LTE	YR	Crop	Rot	Till	Sow Month	Harvest Month
274,5	PD	2019	CORN	1	CV	4	10
274,54	PD	2019	CORN	1	CV	4	10
328	FG	1982	DWHT	1	CV	11	7
493	PD	2013	WWHT	1	CV	10	7
494	PD	2013	WWHT	1	CV	10	7
507	FG	1983	DWHT	1	CV	11	7
598	PD	2009	WWHT	2	CV	11	6
605	FG	1982	DWHT	1	CV	11	7
621	FG	1992	DWHT	1	CV	11	6
642,8	PD	2020	SOYB	4	CV	5	10

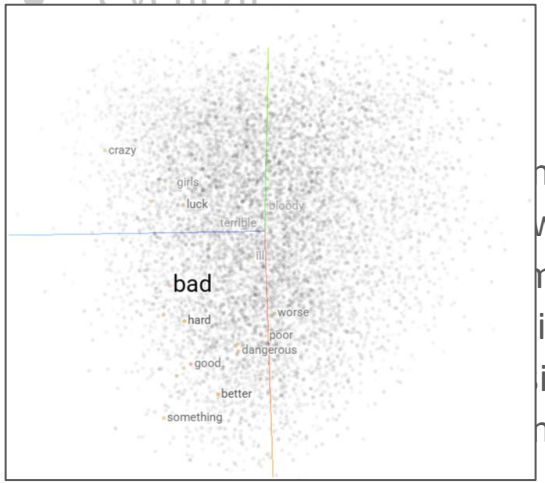
SowMonth_cos	SowMonth_sin	HarvestMonth_cos	HarvestMonth_sin
-0,50	0,87	0,50	-0,87
-0,50	0,87	0,50	-0,87
0,87	-0,50	-0,87	-0,50
0,50	-0,87	-0,87	-0,50
0,50	-0,87	-0,87	-0,50
0,87	-0,50	-0,87	-0,50
0,87	-0,50	-1,00	0,00
0,87	-0,50	-0,87	-0,50
0,87	-0,50	-1,00	0,00
-0,87	0,50	0,50	-0,87

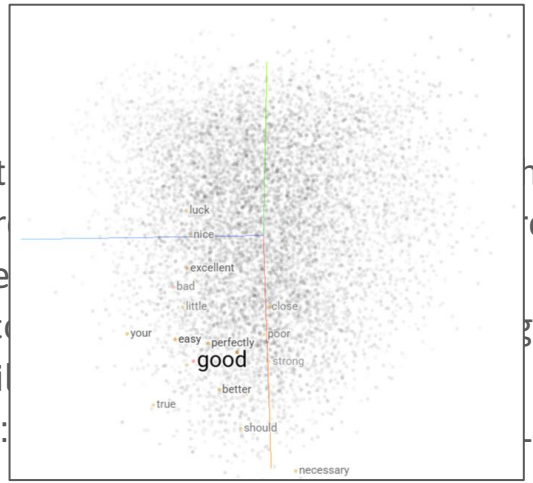
Data - encoding

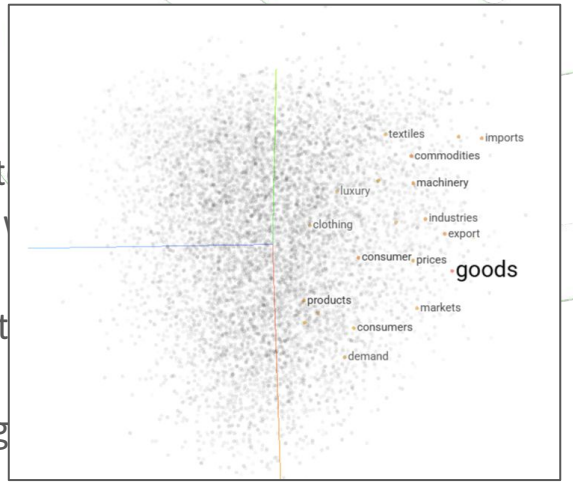
- Numerical
- Categorical
- Cyclical
- Long texts
  - Basic approaches:
    Bag of Words: counting the repetitions of each word in the text
    TF-IDF: frequency of words in a single document compared with other documents
  - Modern approach: Embeddings
    Each word/sentence is converted in a dense vector using a trained model
    Similar meanings → similar vectors
    Pre-trained embeddings: Word2Vec, GloVe, BERT, other Large Language Models

Data - encoding

- Numerical
- Categorical
- Cyclical







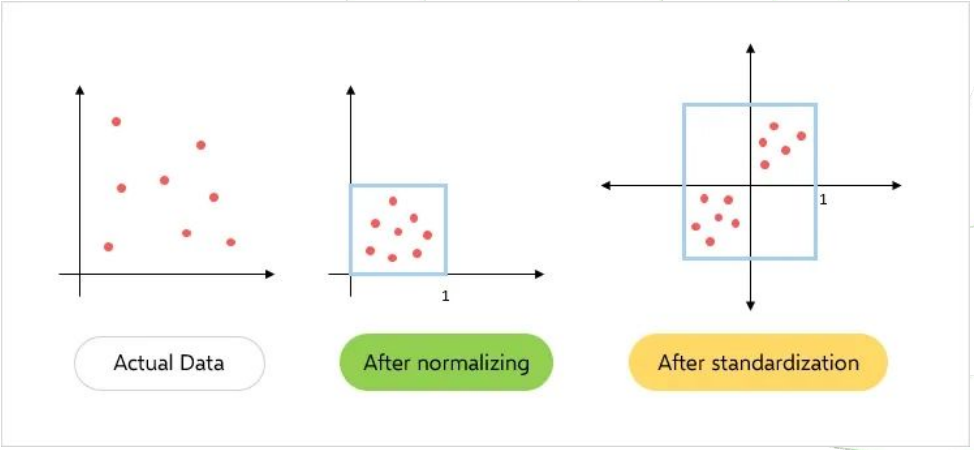
Data - different scales

Different scales of data – but the models see… just numbers!

- **Some** models (like linear regression and neural networks) assume features are on **similar scales**.
- If one feature has very large values, it can dominate training
- Results: unstable training, slower learning, worse predictions

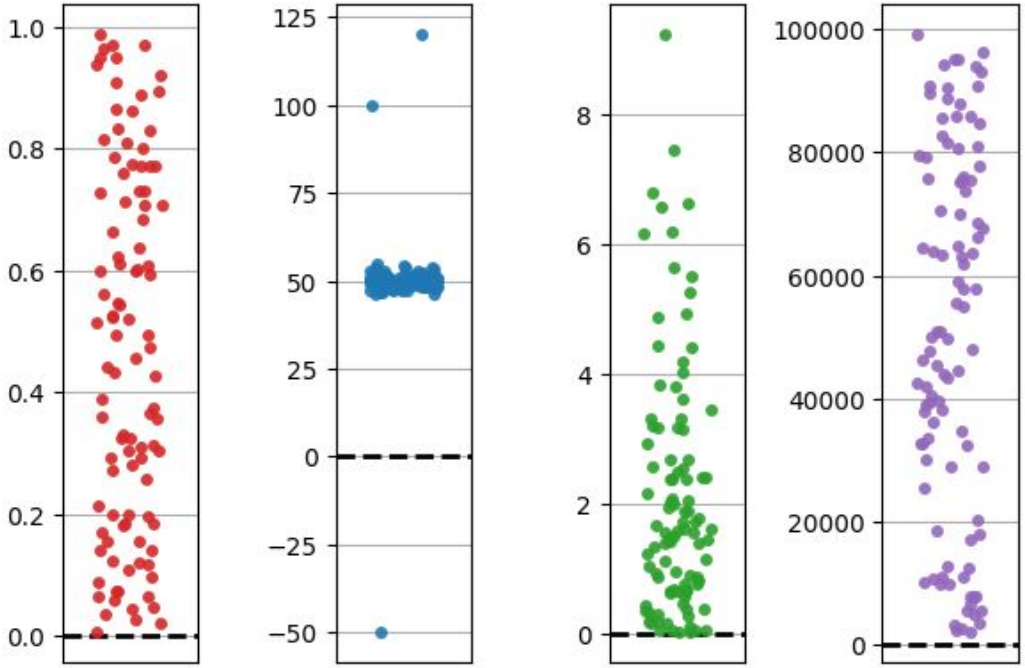
Data - scaling

- Min-Max Scaling
  - Rescales values to [0, 1]
  - Very sensitive to outliers
- Z-score Standardization (Standard Scaling)
  - Mean = 0, Std Dev = 1
  - More robust to outliers
- Others
  - E.g., Robust scaler, Power transformation, Quantile transformation, ...



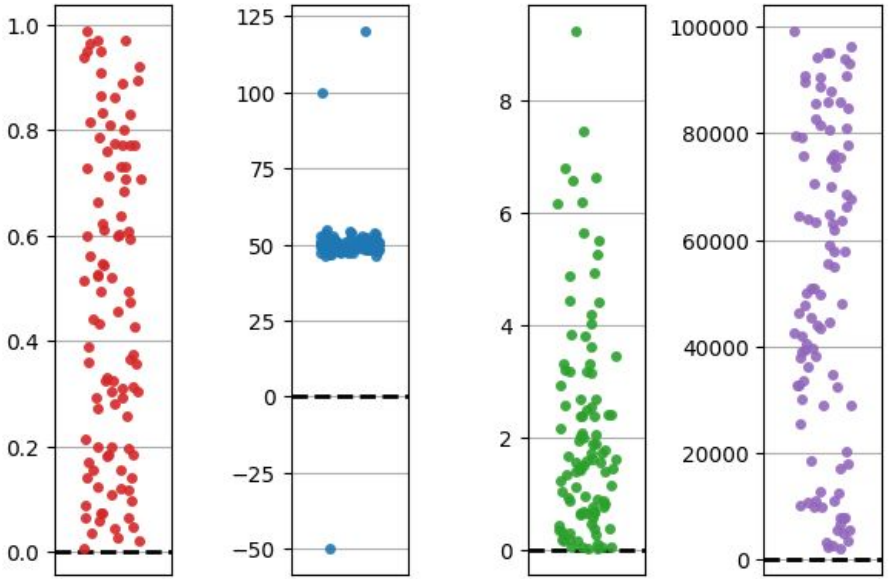
Data - scaling

Original features

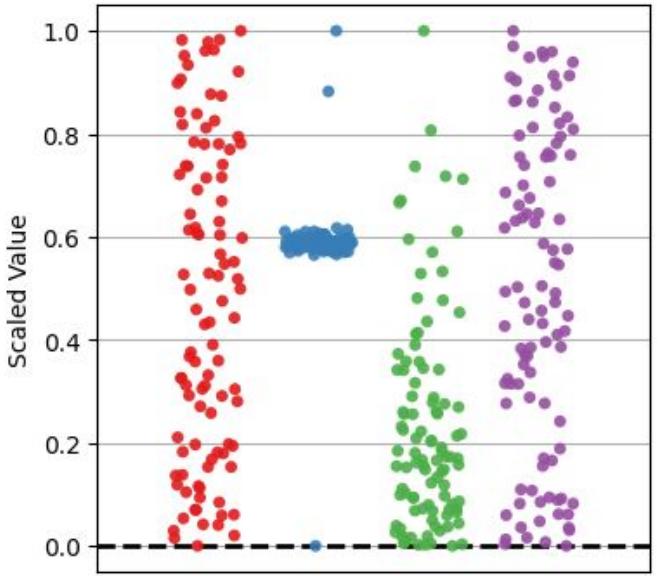


Data - scaling

Original features

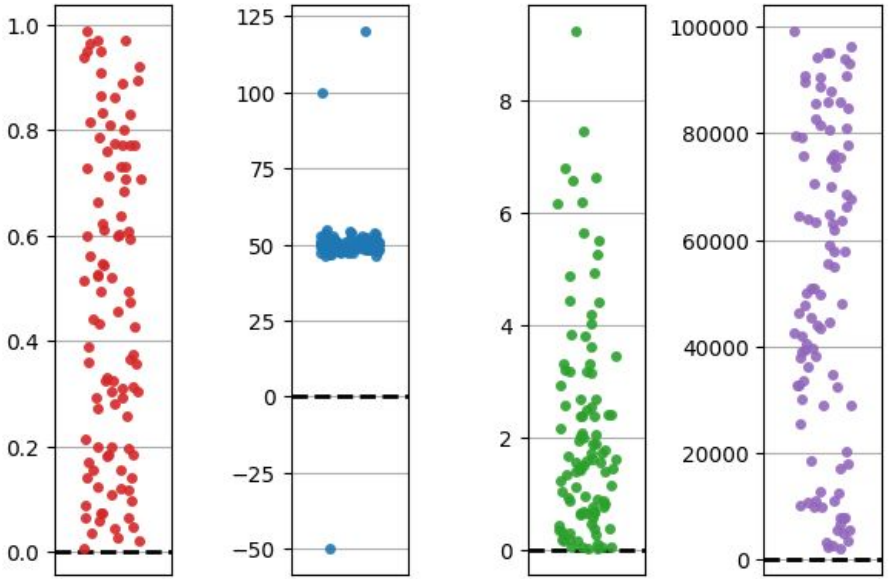


Min-max normalization

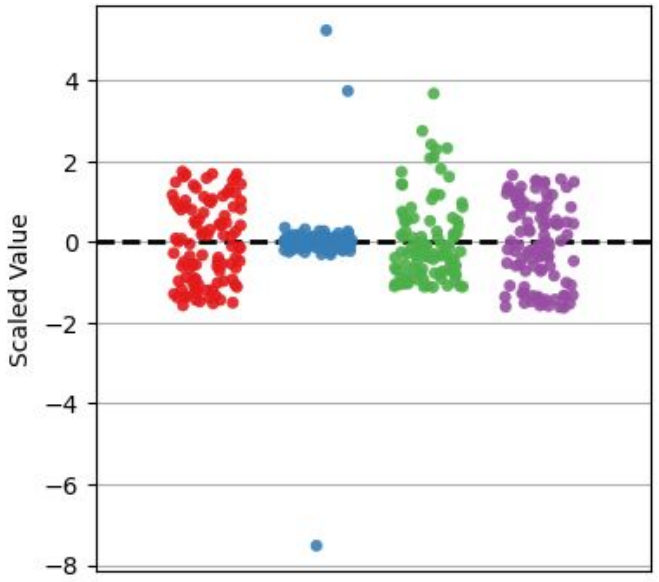


Data - scaling

Original features

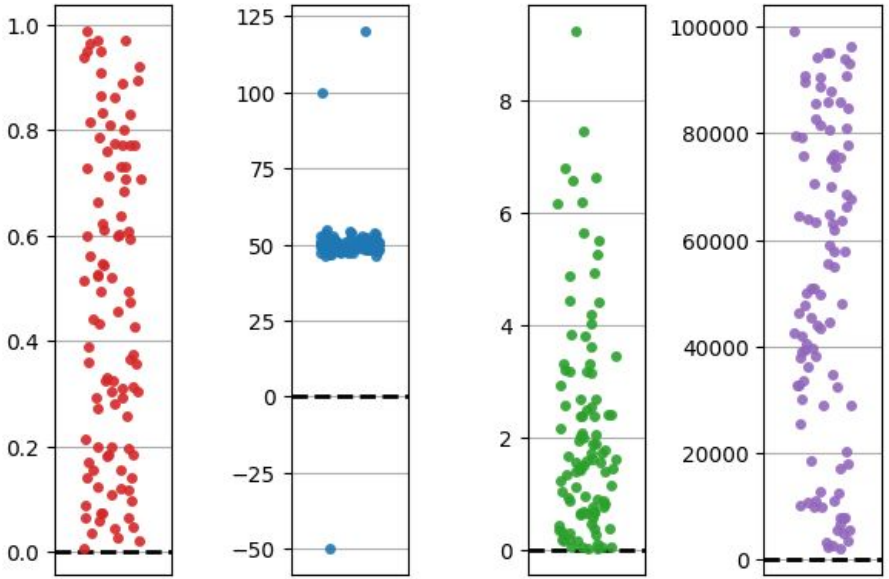


Standard scaling

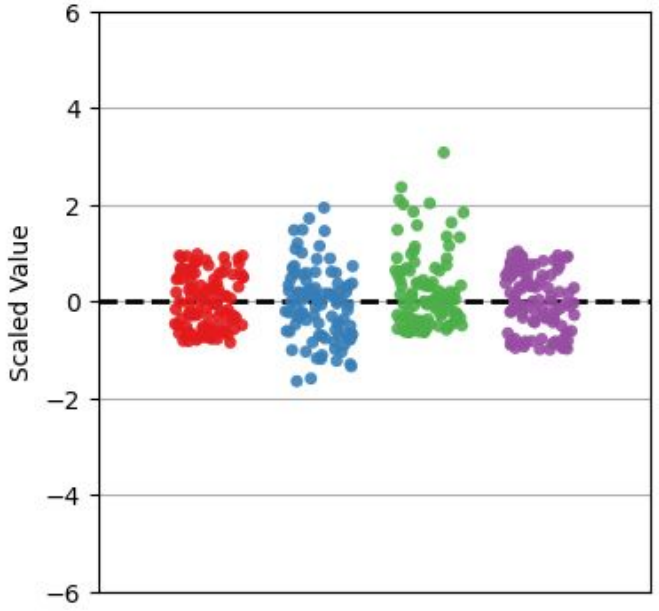


Data - scaling

Original features



Robust scaling



Data - using the right technique

Data Type	Recommended Technique	Notes
Numerical (no outliers)	Min-Max Scaling	Simple and fast
Numerical (with outliers)	Standard Scaling or Robust Scaling	More stable
Categorical (few categories)	One-Hot Encoding	Easy to interpret
Categorical (many categories)	Embeddings / Target encoding	Risk of overfitting
Cyclical (e.g., time)	Cyclical Encoding (sin/cos)	Keeps continuity

Data Scarcity

Data Scarcity - Why?

Not enough data? Welcome to the club!

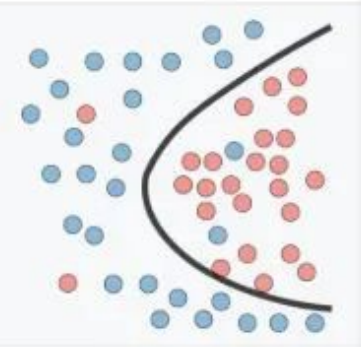
- ML models are data-hungry: the more examples, the better they learn
- But in real-world science (like environmental data):
  - Data collection is expensive
  - Some data are rare events (e.g., floods, pests)
  - Labels may require manual expert work
- So we often face “data scarcity”

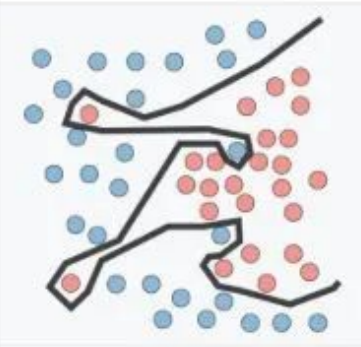
Data Scarcity - Why is it a problem?

Small data, big risk: overfitting!

- With too few examples, the model may:
  - Memorize the training data
  - Fail to generalize to new/unseen data

Enough data for the model





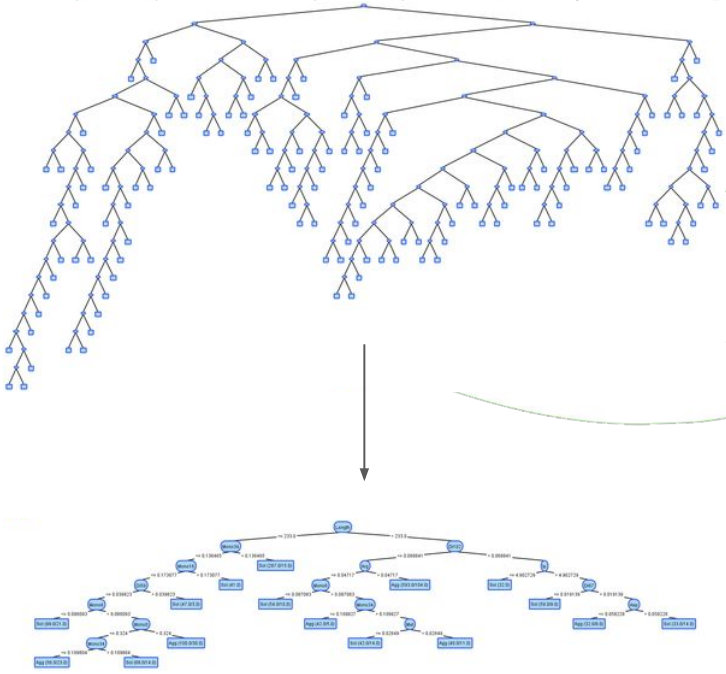
Not enough data for the model

Data Scarcity - What can we do?

- Try to **collect (and label) more data**
  → Costly but best option in the long run

Data Scarcity - What can we do?

- Try to **collect (and label) more data**
- Use **simpler models** or put limitations on the models you are using
  (e.g., linear regression, decision trees with depth limit)
  → Less capacity to memorize noise



Data Scarcity - What can we do?

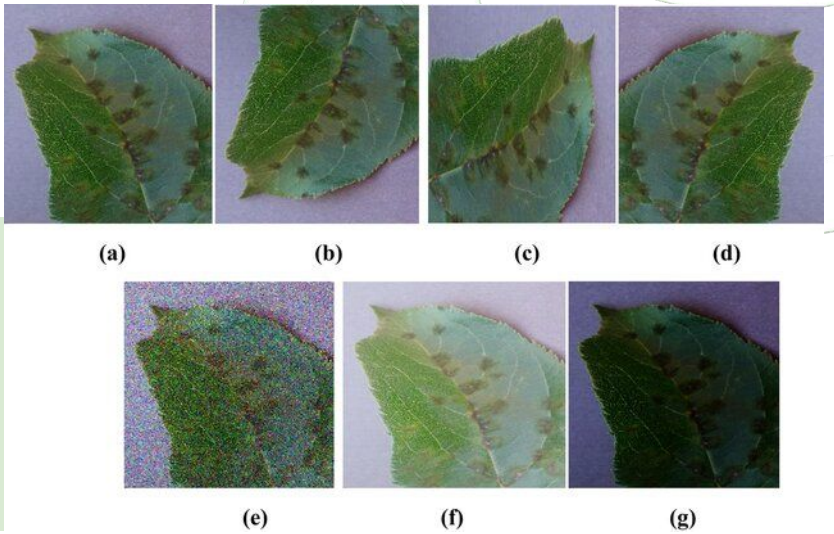
- Try to **collect (and label) more data**
- Use **simpler models** or put limitations on the models you are using

- Apply **data augmentation**
  Create new training examples by tweaking existing ones

- Generate **synthetic data**
  Create new training examples by “simulating” or generating new ones

Data Scarcity - Data Augmentation

- Create new training examples by **tweaking existing ones**
  - Images: crop, rotate, flip, change brightness, ...
  - Text: replace synonyms, paraphrase, ...
  - Time series: scale, window slicing, ...
- Helps models generalize better



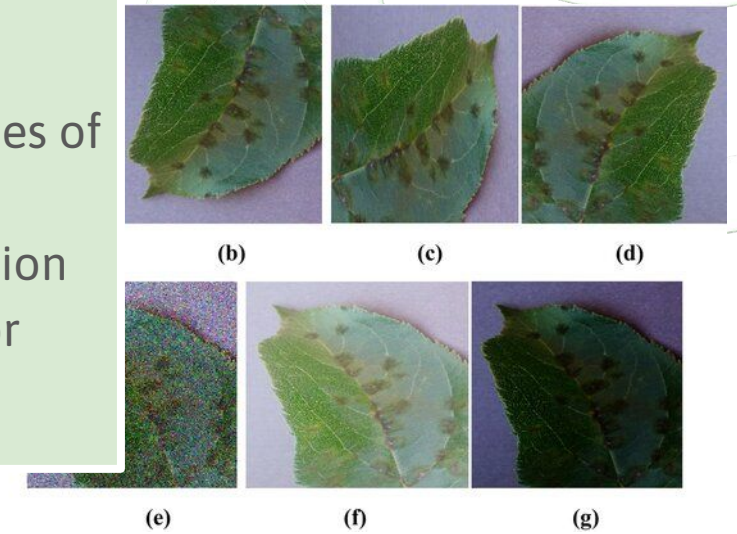
🚫 Only to use on **training** data!
Measuring the model's performance on augmented images may be "cheating" (or make the results non comparable)

Data Scarcity - Data Augmentation

⚠️ Try to use only **meaningful** modifications depending on the task

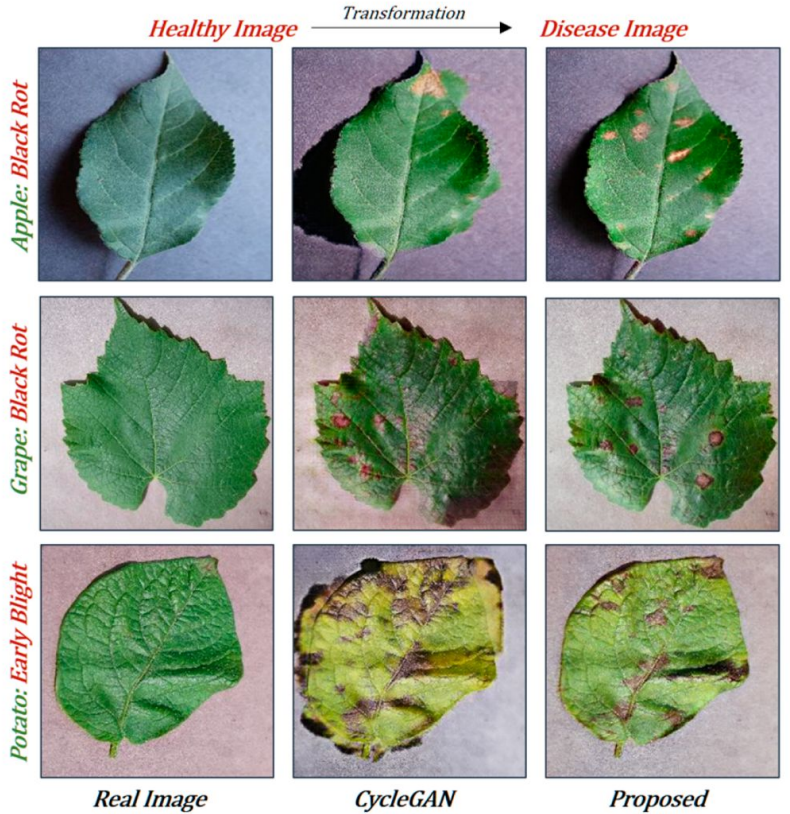
Example: leaf disease classification

✅ modify brightness to simulate different times of the day
✅ flip/rotate image for different leaf orientation
🚫modify hue (from green to yellow-red): color might be a symptom of a disease



Data Scarcity - Synthetic data

- When real data is hard to get:
  - Use domain simulations (e.g., climate models)
  - Use generative models (e.g., GANs, VAEs)
- Can help “pre-train” a model before fine-tuning on real data



🚫 Synthetic data is usually **realistic** and only used as **training** data!
Measuring the model’s performance on synthetic images is “cheating”.

THANKS!

IR0000032 – ITINERIS, Italian Integrated Environmental Research Infrastructures System
(D.D. n. 130/2022 - CUP B53C22002150006) Funded by EU - Next Generation EU PNRR-Mission 4 “Education and Research” - Component 2: “From research to business” - Investment 3.1: “Fund for the realisation of an integrated system of research and innovation infrastructures”

Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani