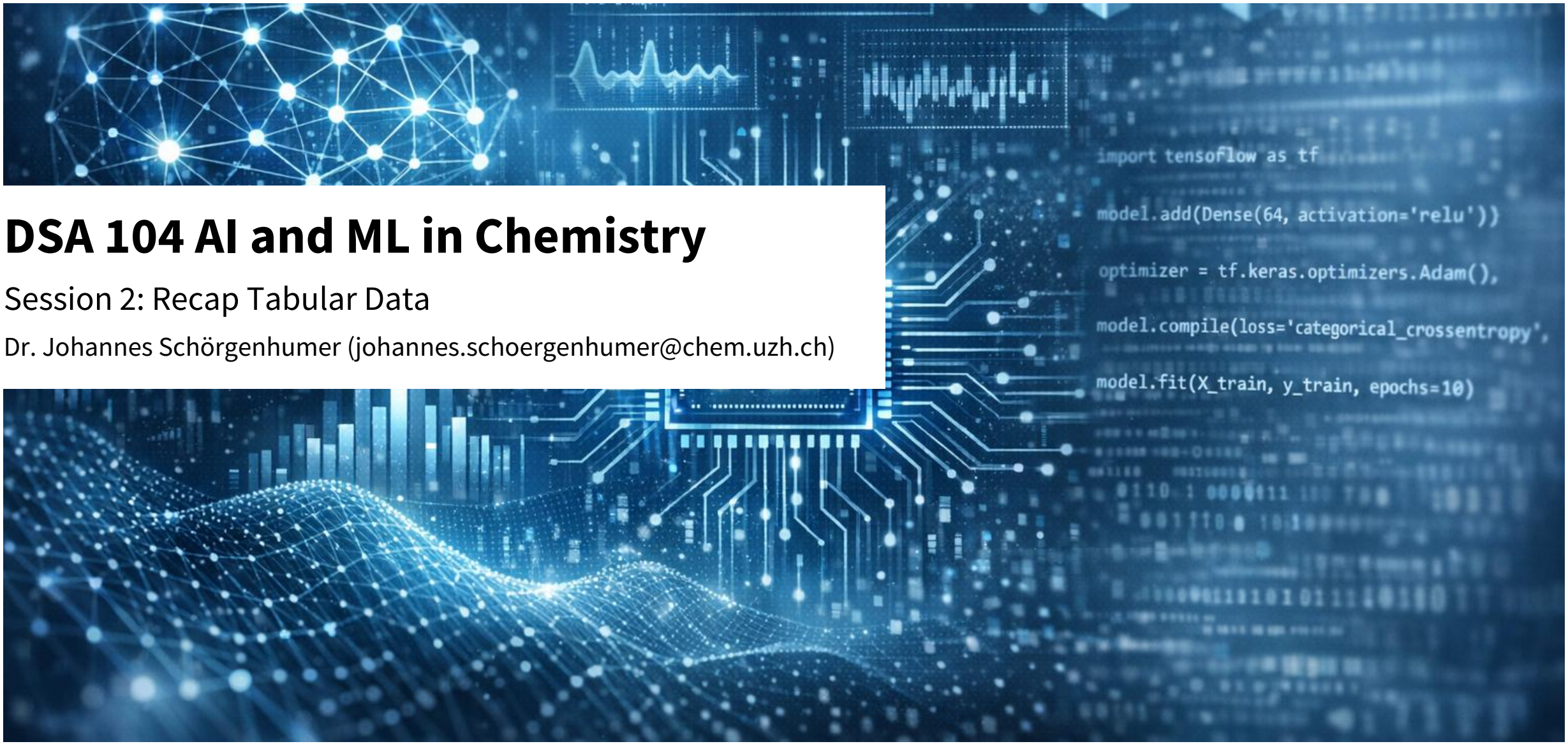
Universität Zürich[UZH]

Department of Chemistry



DSA 104 AI and ML in Chemistry

Session 2: Recap Tabular Data

Dr. Johannes Schörgenhumer (johannes.schoergenhumer@chem.uzh.ch)

Lesson outline

See that the environment is working: Walk through last times exercise

Prerequisites: Python

Tabular data

Pandas

Visualisation

Dimensionality reduction

Exercise: EDA

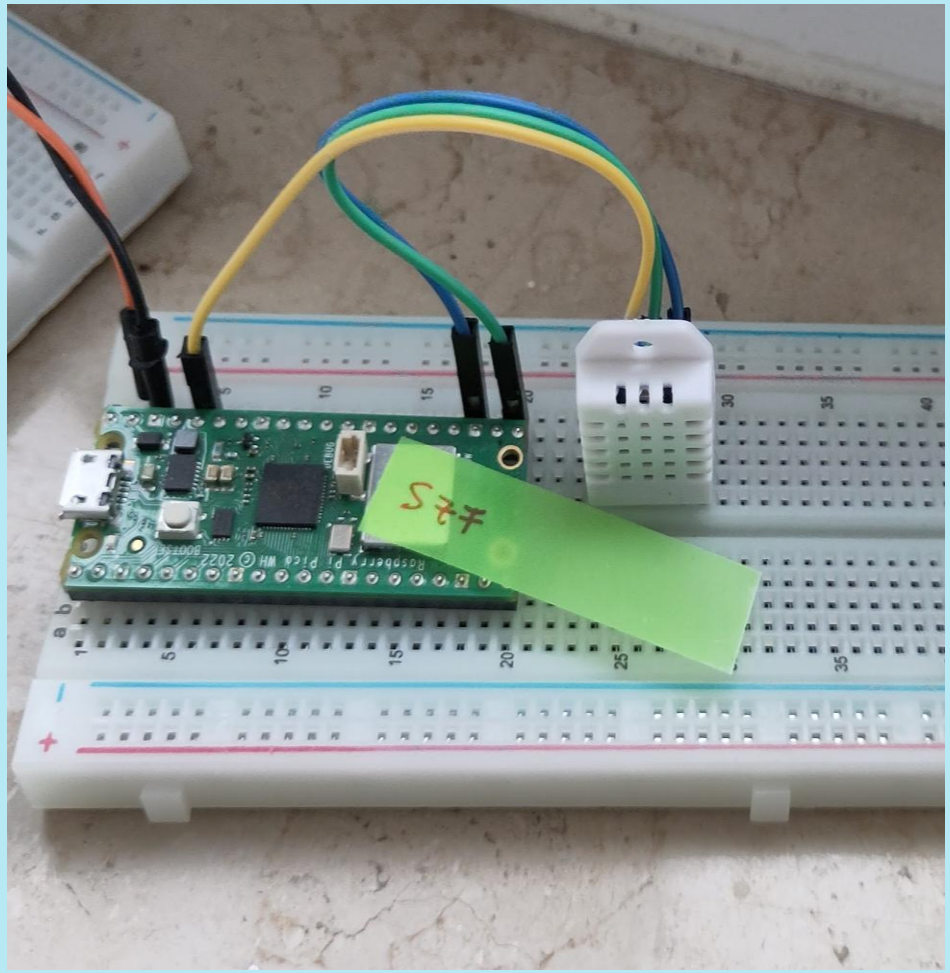
Exercise: Predicting sensor data

Scenario: Heating in a room in our apartment not sufficient, high temperature gradient towards outer wall and floor.

Several sensors (DHT22) have been placed around the apartment and on the balcony to record temperature and relative humidity. On a Raspberry Pi Pico, the data is timestamped and written into a .csv file.

Tasks:

1) **Functionality check: Run the provided notebook “dht_prediction”**
2) Predict missing temperature and humidity values by regression models
3) Find out which model works best for the prediction and play around with some parameters



Learn how you can do this (and more fun stuff):
CHE 725 Hacking for Chemists (2 ECTS elective, HS)

Prerequisites: Python

Why python?

- Massive ecosystem (Pandas, PyTorch, Seaborn, …)
- Easy to learn – quick to test
- A “classic” data science choice
- Huge community and support
- Interoperability

Resources to acquire the basics:

- Dedicated Modules, e.g.
  - Informatics I (https://studentservices.uzh.ch/uzh/anonym/vvz/?sap-language=DE&sap-ui-language=DE#/details/2025/003/SM/51110636)
- Online courses, e.g.
  - https://www.learnpython.org/
  - https://www.w3schools.com/python/default.asp
- ZI trainings: https://zi-training.zi.uzh.ch/angebot/details/9686?culture=en

Prerequisites: Python

Recommendations for Python in the age of generative AI:

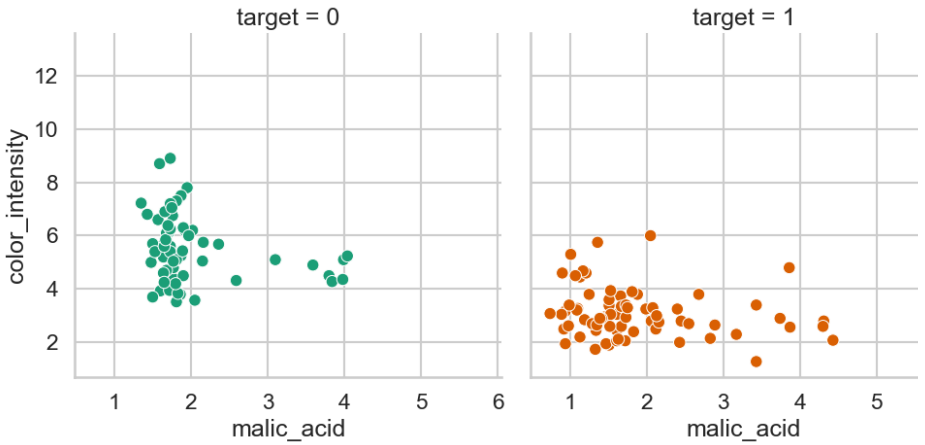
- Learn the basics!
- Explore and apply – learning by doing!
- Use all tools available – “no heroes in coding”, **but:**
- Vibecoding has its downsides (errors, **dependency!!**)
- Explore other sources four trouble shooting:
  - docstrings (e.g. in Jupyter add *?*)
  - Package documentation
  - stackoverflow
  - Good old google ☺

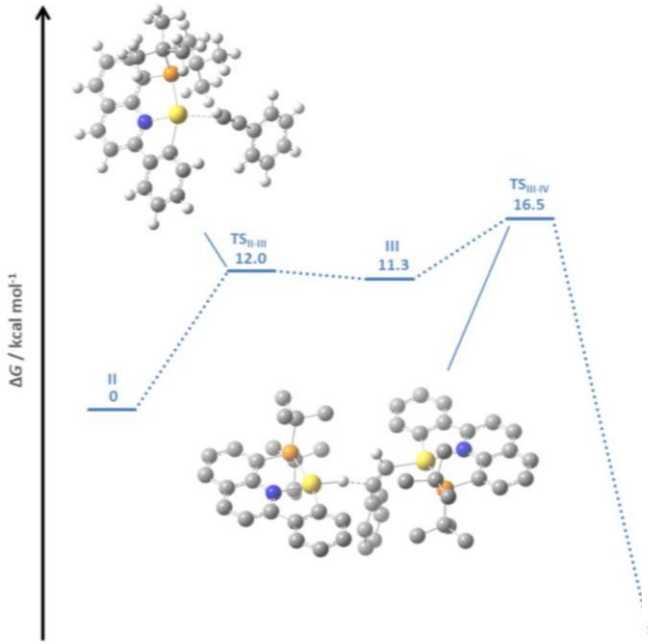
What level of skills do you need for this course?

- Understand core syntax and logic!
  - Import packages
  - Data types
  - Define functions
  - Conditional logic, loops

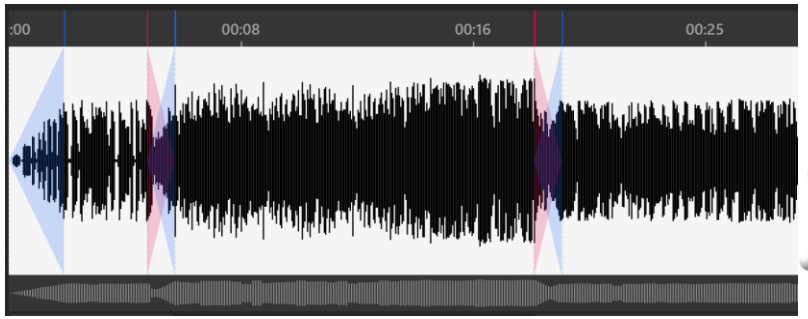
Motivation: The data science loop

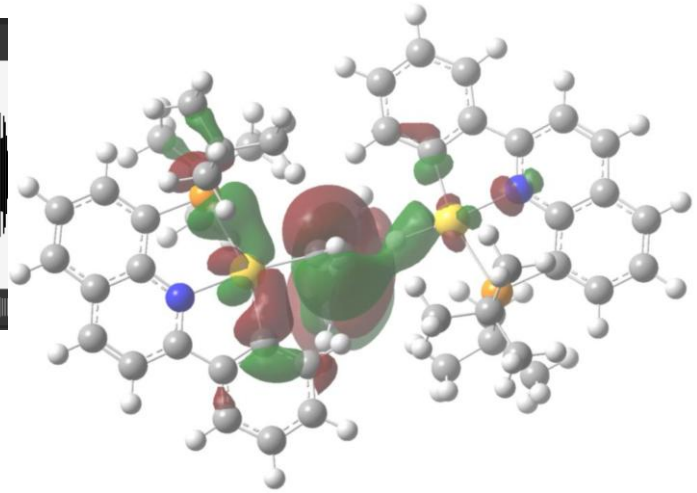
Data Types

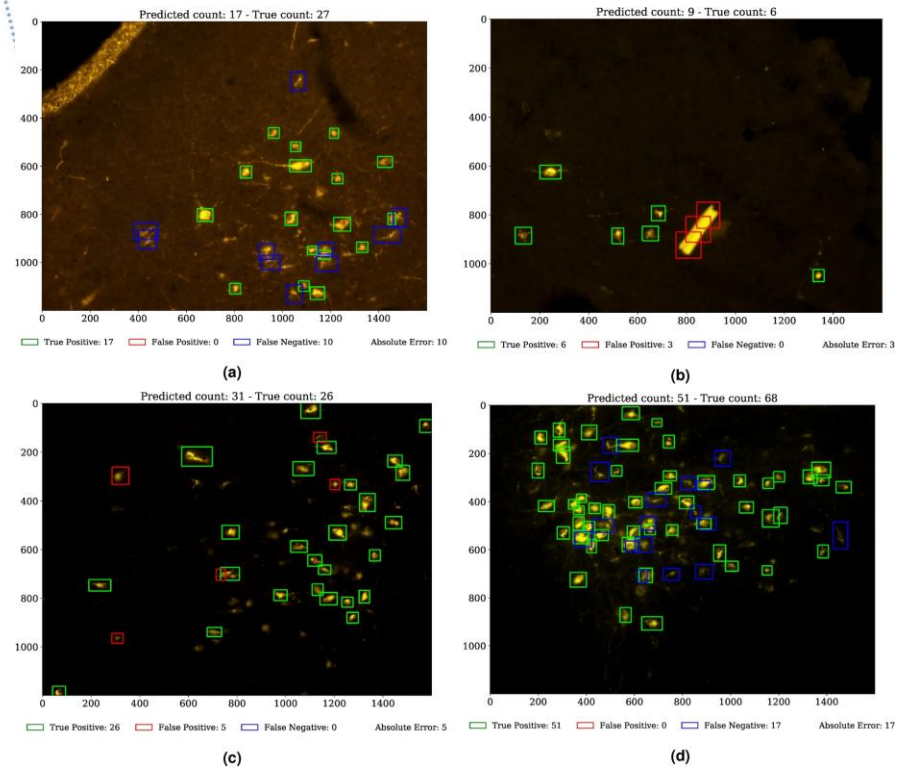




Scientific data types ≠ Python data types!!







DNA Blot: By Mnolf - Photo taken in Innsbruck, Austria, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1131449
Cell detection: R. Morelli, L. Clissa, R. Amici, *et al., Sci Rep* **2021**, *11*, 22920, https://doi.org/10.1038/s41598-021-01929-5
Computations: J. Martín, J. Schörgenhumer, C. Nevado, *JACS Au* **2025**, *5*(3), 1439–1447, https://doi.org/10.1021/jacsau.5c00056

Tabular data

- Very commonly, data is structured in a tabular form.
- Data elements are arranged in vertical columns (**features, labels**) and horizontal rows (**samples**).
- Each column and row is uniquely numbered (**implicit indices**).
- Tabular data has a virtually infinite range for mass data storage (you can always add rows).

- Tabular databases include the following key properties:
  - Share the same set of properties per record, i.e., every row has the same column titles.
  - Each column is (usually) assigned with a header title (**explicit index**, metadata).
  - Access through identifiers, i.e., each object can be retrieved by a query through key values.

Tabular data: Example

— E.g. Iris data set

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

— Four **features** (**d = 4**)

— Every flower entry is referred to as **sample**

— Each sample is described by four features, which can be represented as **feature vector,** e.g. **x = (5.1, 3.5, 1.4, 0.2)**

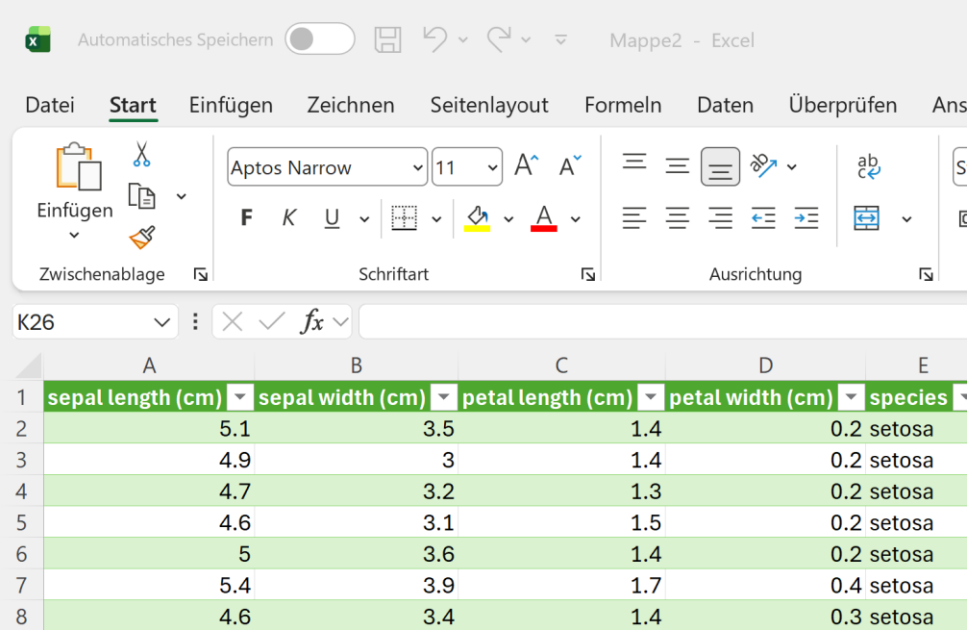
— Three flower species, i.e. three **classes**

— Every sample lists the species / classes via its **label**, e.g. **y = setosa**

E. Anderson, “The Species Problem in Iris.”, *Annals of the Missouri Botanical Garden* **1936**, *23*, 457–509, https://doi.org/10.2307/2394164.
R. A. Fisher, “The Use of Multiple Measurements in Taxonomic Problems”, *Annals of Eugenics* **1936**, *7,* 179-188, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x

Tabular data: Handling

- Excel not ideal for processing:
  - Slow for big data
  - Binary format!
  - Commercial license
- Store data in easily accessible file format, e.g. **.csv** (and similars), **.json**
- Work with **Pandas dataframes!**



	A	B	C	D	E
1	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   species            150 non-null    object
dtypes: float64(4), object(1)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Handling data: Pandas

```
import pandas as pd
```

- Pandas is the bread and butter of data science in python!
- Built on Numpy
- Two objects: **Series** (1D) and **Dataframes** (2D) for tabular data including **column** headers (= **axis 1**) and row **indexes** (= **axis 0**)

- The crucial pandas functionalities that we are going to use:
  - Importing and inspecting data (**read_csv**, **info**, **describe**, **head**)
  - Indexing & selection (**loc**, **iloc**, **boolean filtering**)
  - Data cleaning & preprocessing (**isna**, **duplicated**, **astype**, **dropna**, **drop_duplicates**)
  - Grouping & aggregation (**groupby**)
  - Merging & joining datasets (**merge**, **concat**)
  - Reshaping data (**melt**, pivot)

columns (axis 1)

index (axis 0)

	sepal length (cm)	sepal width (cm)
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

Pandas - exercise

1) Quick pandas demo: *pd-demo* notebook
2) Pandas puzzles: *pd-puzzles* notebook

Understanding data: EDA Recap

Key questions in Exploratory Data Analysis (EDA):

— What does each feature look like?

— Are there outliers?

— Are features correlated?

— Are there batch effects?

Typical tools

— Basic information, summary statistics

— Visualisation:

  — Histograms

  — Scatter plots

  — Correlation matrices

— Dimensionality reduction

https://www.geeksforgeeks.org/data-analysis/what-is-exploratory-data-analysis/

Visualisation – brief excurse

- Gaining **insights** into your data is essential and the first step towards a successful model
- Besides basic information and summary statistics, looking at the raw data is often infeasible or does not help (too much data, too many features).
- **Visualisation** is a quick and powerful tool in this regard.
- Which type of visualization works best is highly dependent on the data:
  - Individual features: histograms, box plots
  - Time series data: line plots
  - 2D data: scatter plots, e.g. with colour-encoding of different labels / classes
  - …

- **Visualisation is also key for communicating results!!**
- **Follow good practices for representative plots!**
- Find more information in dedicated resources!

```
import seaborn as sns
import matplotlib.pyplot as plt
```

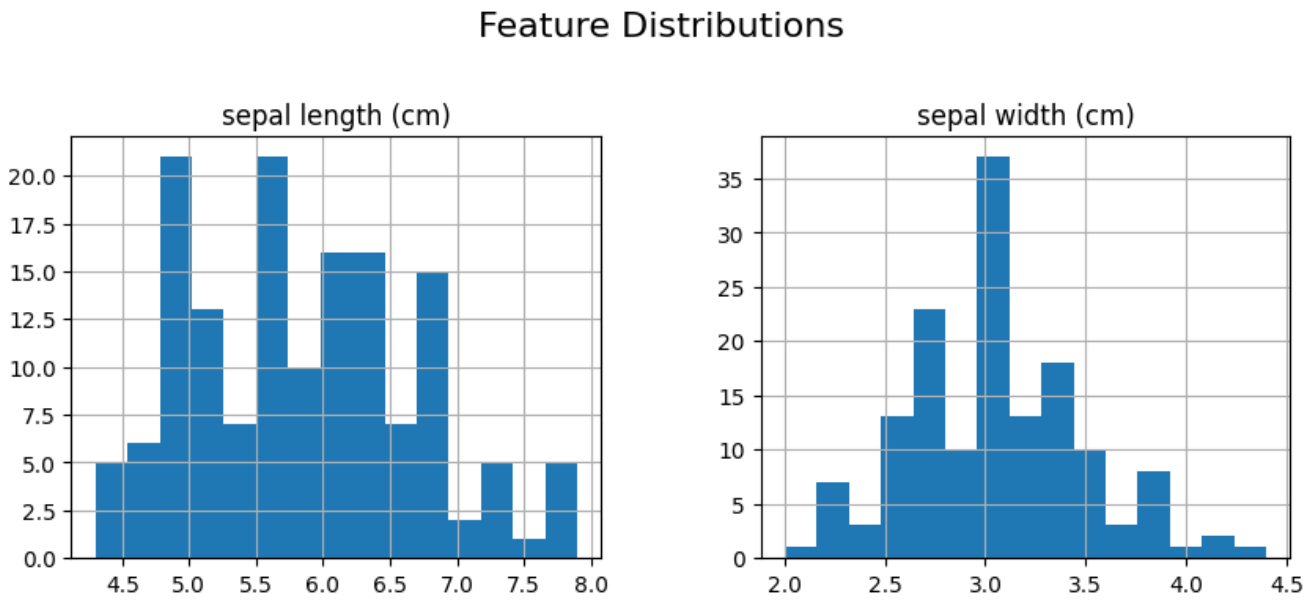
Visualisation – Histogram

A histogram shows the statistical distribution of the features.

- **Matplotlib:**

  E.g. plot for all features

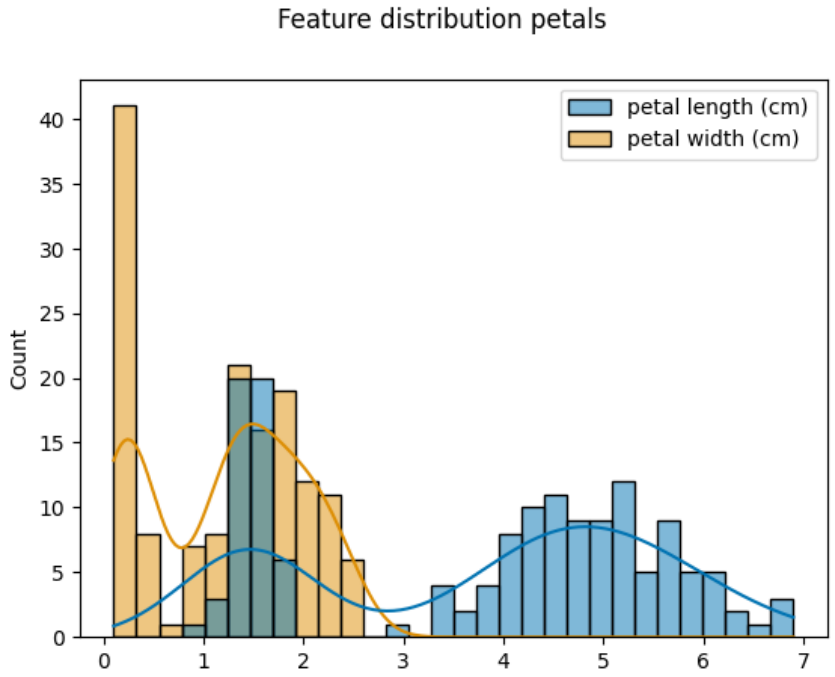
```
# Histograms: Distribution of each numerical feature
iris.hist(figsize=(10, 8), bins=15)
plt.suptitle('Feature Distributions', fontsize=16)
plt.show()
```



- **Seaborn:**

  E.g. plot for some selected features

```
sns.histplot(
    data=iris[["petal length (cm)", "petal width (cm)"]],
    bins=30,
    kde=True,
    palette="colorblind")
plt.suptitle("Feature distribution petals")
plt.show()
```



https://seaborn.pydata.org/generated/seaborn.histplot.html

Visualisation – Box Plots

Box plots are a great method to provide an **overview** on the locality, spread and skewness groups of numerical data through their quartiles. Box plots also reveal potential outliers.

Maximum (Q4 / 100th percentile): highest data point (outliers excluded) indicated by “whiskers”

Upper quartile (Q3 / 75th percentile): median of upper half of the dataset, indicated by upper box boundary

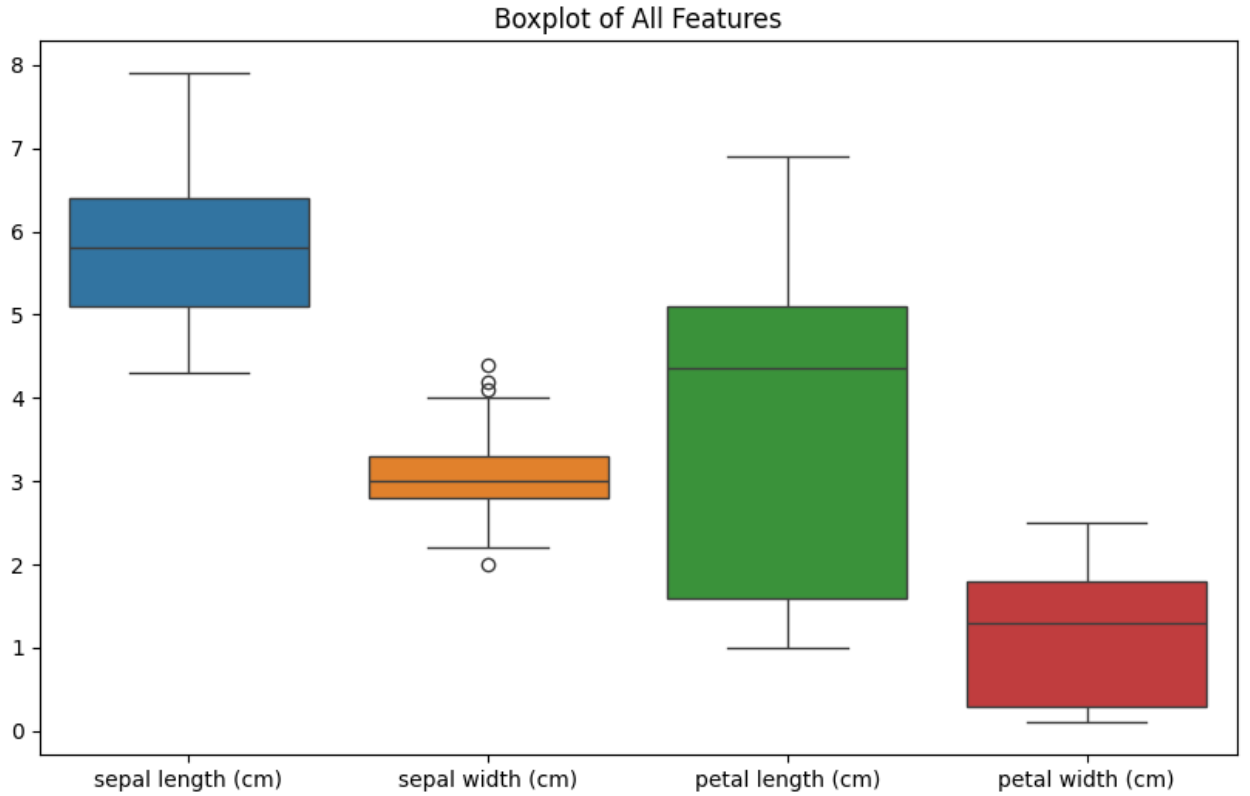
Median (Q2 / 50th percentile): indicated by line in the box

Lower quartile (Q1 / 25th percentile): median of lower half of the dataset, indicated by upper box boundary

Minimum (Q0 / 0th percentile): lowest data point (outliers excluded) indicated by “whiskers”

Interquartile range (IQR) = Q3 – Q1. Usually, 1.5*IQR is considered the threshold for **outliers** (marked with small circles)

https://seaborn.pydata.org/generated/seaborn.boxplot.html




```
# Boxplots to check for outliers
plt.figure(figsize=(10, 6))
sns.boxplot(data=iris)
plt.title('Boxplot of All Features')
plt.show()
```

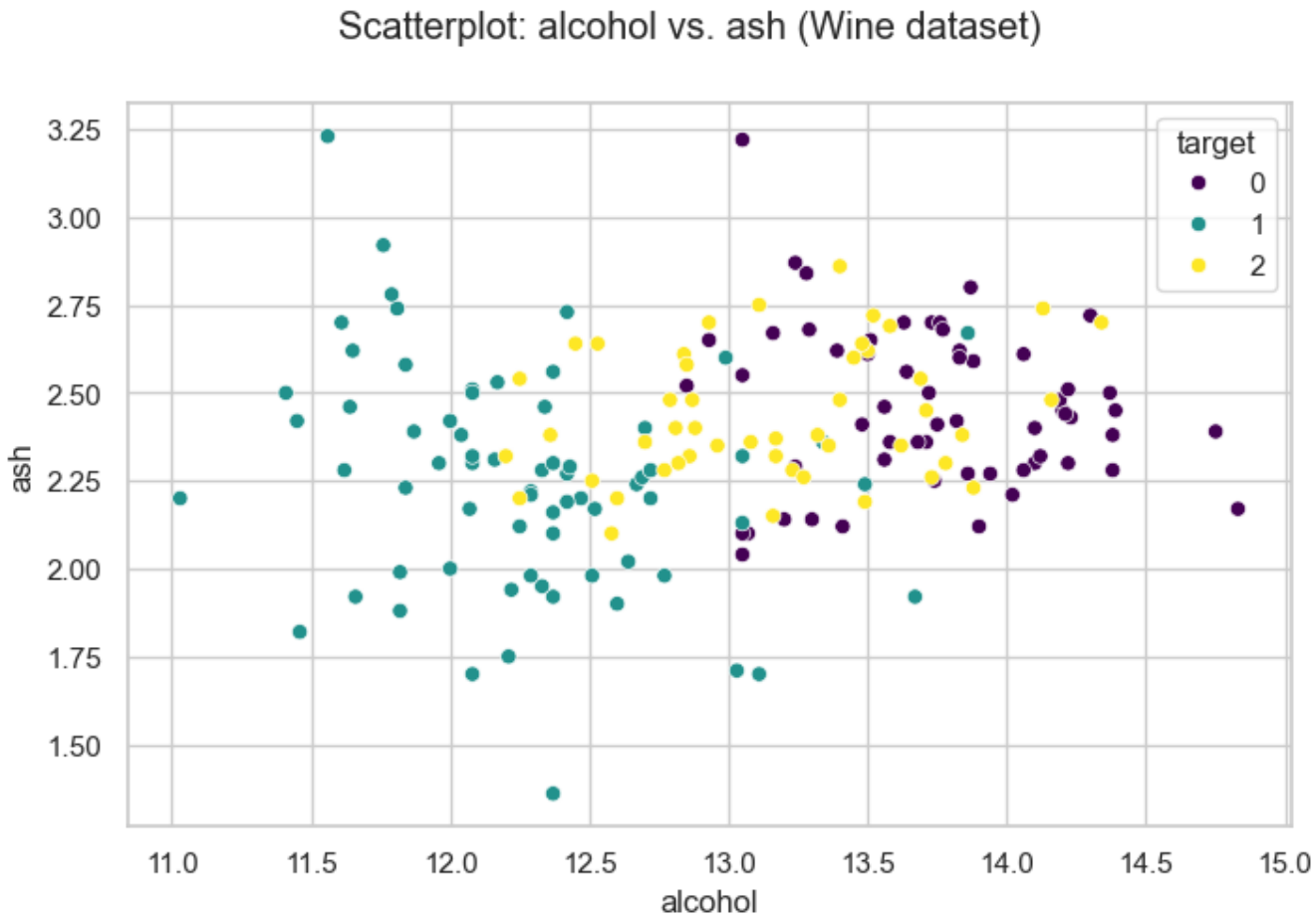
Visualisation – Scatter Plot

Scatter plots are the classic plot to **compare two features** on two axes (**x = “feat1”**, **y = “feat2”**)

A third dimension can be added by adding a coloured hue (**hue = “feat3”**)

A good choice in colour palette helps visualization if a hue is set.

```
plt.figure(figsize=(8,5))
sns.scatterplot(
    data=df,
    x="alcohol",
    y="ash",
    hue="target",
    palette="viridis"
    )
plt.suptitle("Scatterplot: alcohol vs. ash (Wine dataset)")
plt.show()
```



https://seaborn.pydata.org/generated/seaborn.scatterplot.html

Visualisation – Pair Plot

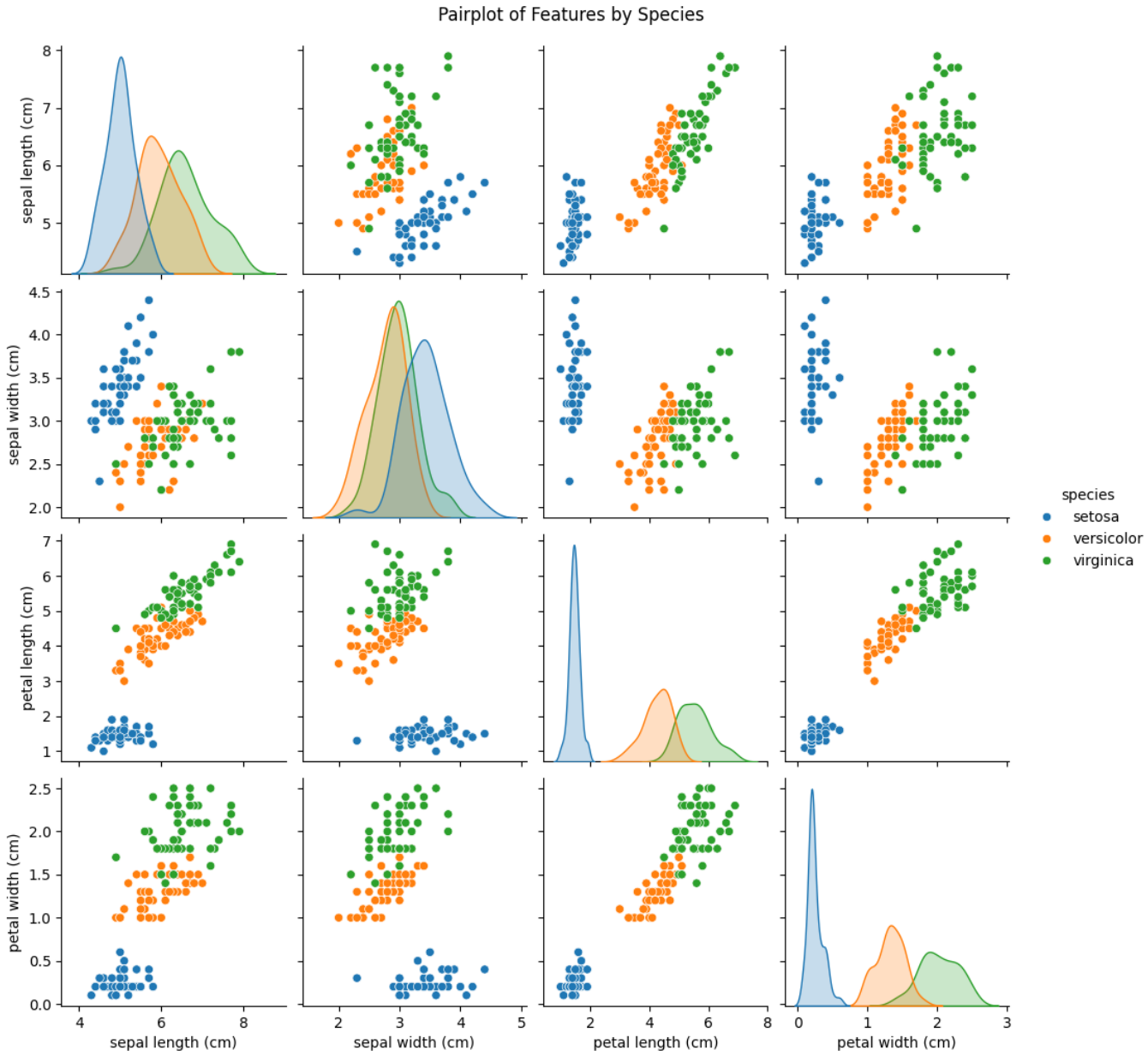
Pair plot is a method for visualizing **relationships between multiple variables** in a dataset in order to visualize patterns easily

Essentially a grid of scatter plots (other plot types can be defined with **kind**)

The diagonal can be chosen, e.g. as histograms or kde (kernel density estimate)

Mirrored around diagonal – redundancy (can be avoided by corner=True)

```
# Pairplot: scatter plots between all numeric variables
sns.pairplot(iris, hue='species', diag_kind='kde')
plt.suptitle('Pairplot of Features by Species', y=1.02)
plt.show()
```



https://seaborn.pydata.org/generated/seaborn.pairplot.html

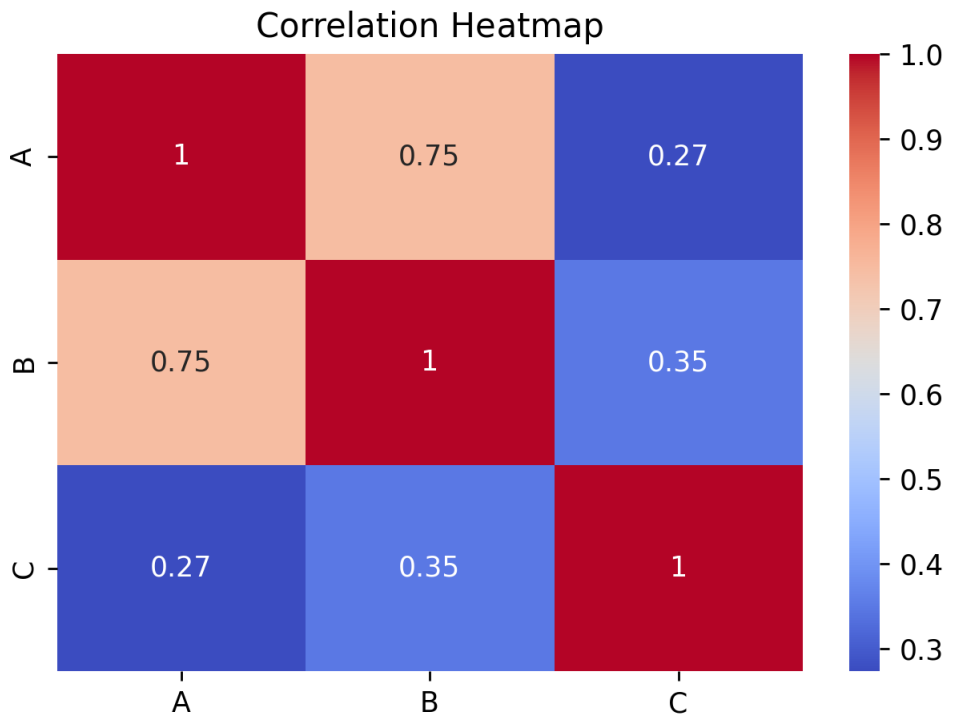
Visualisation – Heatmap

Heat maps plot rectangular data as a colour-encoded matrix where the function for the hue (or intensity) is depending on the purpose.

Very diverse use cases, e.g.

- Correlation between features (here)
- Temperature maps
- Visualising quality of a result (e.g. reaction yield)
- …

```
# correlation heatmap
plt.figure(figsize=(6, 4))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



https://seaborn.pydata.org/generated/seaborn.heatmap.html

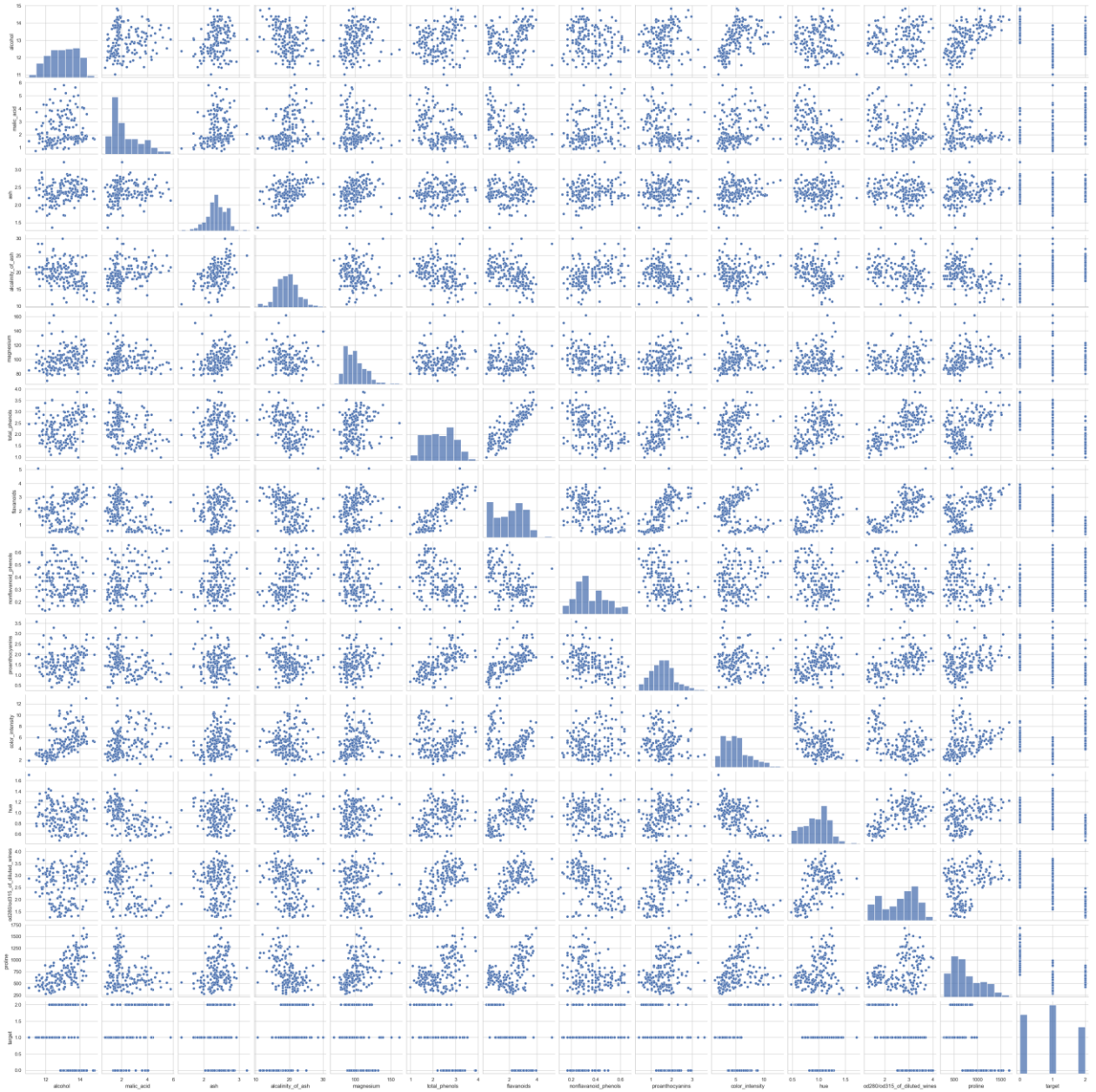
The visualisation problem

E.g. Wine data set: 13 features.

We can still look at all of them manually and make comparisons with each other.

What about bigger data sets? Let's say 500 features?

Of course, we could still look at all features individually or compare features pair-wise, or…



Dimensionality reduction

High dimensional data has some practical drawbacks:

- Difficult visualisation
- Slower training for ML models (more features)
- Overfitting (more noise than patterns)
- Multicollinearity (quite a few features may be correlated)

Dimensionality reduction techniques, reduce number of features while preserving as much information as possible. Basically, two types:

1) Feature selection: Original features are preserved, but fewer of them will be taken into account (e.g. Filters)
2) Feature extraction: new, compressed representations of the data are obtained by transforming or combining original features to capture most of the variance / information in the data. E.g.:
   - **PCA** (Principal Component Analysis): most popular method, linear technique
   - **t-SNE** (t-distributed Stochastic Neighbor Embedding): non-linear technique
   - Autoencoders: neural network-based

Dimensionality reduction: PCA

What PCA is:

— Variance compression to n components (maintaining maximum variance)

— “Finding the main axes of variation”

— Unsupervised **transformer model** (learns parameters from X without using labels y)

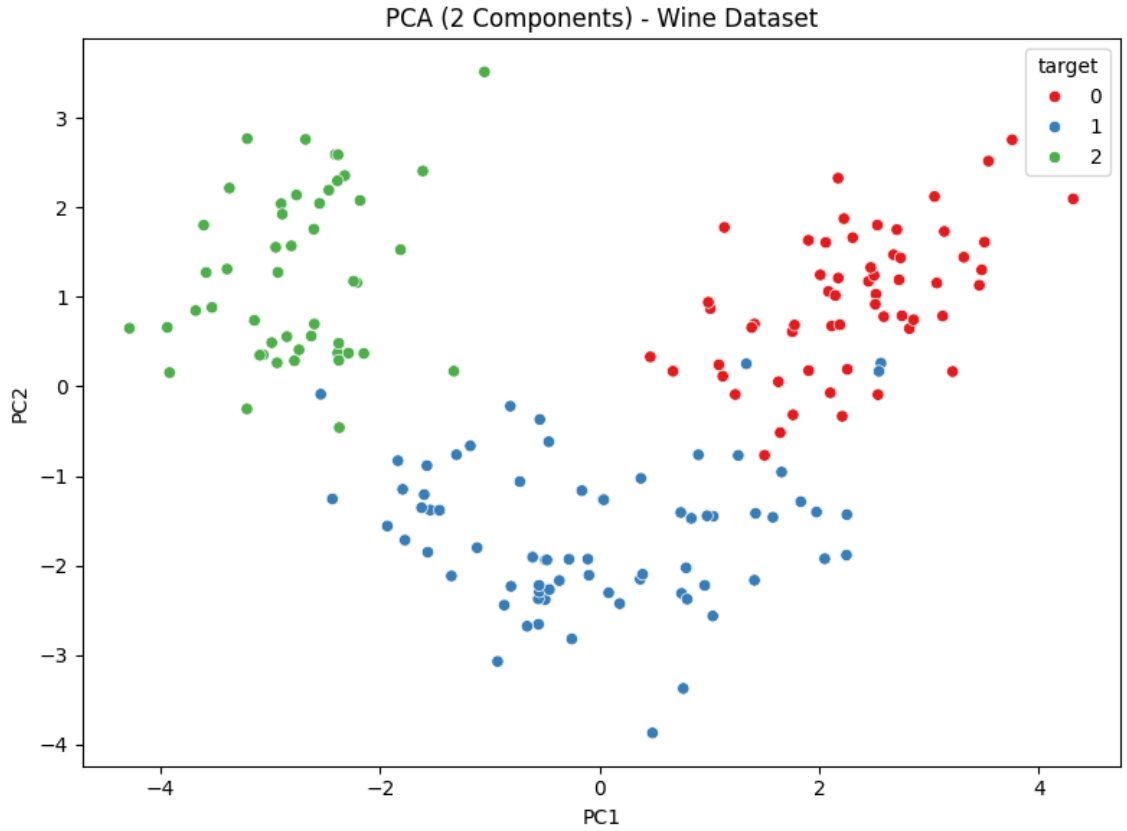
— Linear: dimensional information retained (still limited interpretability)

What it’s *not*:

— Predictive

— A classifier or clustering technique

— Magic feature selection



Working principle:

1) Standardizing the data (so each feature has equal importance)

2) Calculate covariance matrix to check how features relate to each other. Finding directions (“principal components”, **PCs**) where the data varies the most.

3) After finding the components that capture the **maximum variance**, the original data is projected onto these new axes

Exercise: EDA on wine dataset

This dataset contains:

— 178 samples

— 13 numeric features describing chemical properties of wines

— 3 target classes (wine cultivars)

Checklist:

1) Import and load data
2) Get an **overview** about information, shape and noise in the data: head, size, shape, info
3) Do some **quick statistics** and **summary**: value_counts, describe, nlargest, nsmallest, nunique
4) Look if data is **missing** (isna, notnull) or if there are **duplicates** (duplicated)
5) Visualisation of features: histogram (pick a feature), box-/violinplot (pick a feature), pairplot, correlation heatmap
6) PCA & scatter plot

7) Compare the code for the PCA with the code for the Regressions in the dht-data notebook.