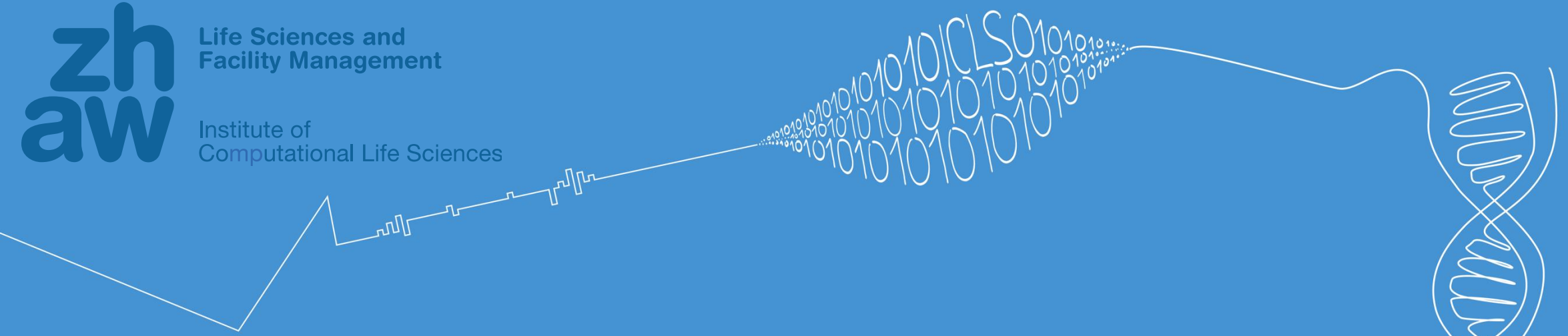
Life Sciences and
Facility Management

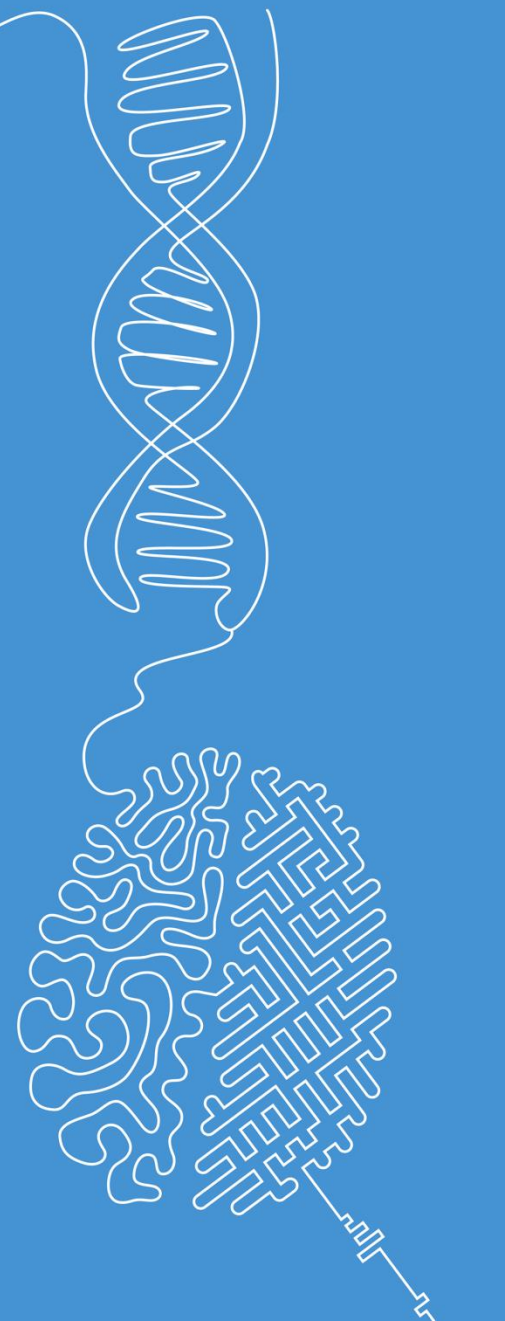
Institute of
Computational Life Sciences



Exploratory data analysis

Machine Learning

Norman Juchler



Miscellaneous

Stuff to discuss before the lecture

Quiz SW04 – Handle missing values

Assume we have a dataset of 1000 samples, each with 10 features stored in a DataFrame `df`. Assign each of the following code snippets with the correct description of what it does.

`df.isnull().any(axis=0).sum()`	Count the number of columns (features) with missing values
`df.isnull().any(axis=1).sum()`	Count the number of rows (samples) with missing values
`df.isnull().any().sum().sum()`	Count the total number of missing values in the DataFrame
`df.dropna(axis=0)`	Remove rows (samples) with missing values
`df.dropna(axis=1)`	Remove columns (features) with missing values
`df.fillna(df.mean())`	Fill missing values with the mean of each column (feature)
`df.drop(col_names, axis=1)`	Remove columns (features) with names in the list col_names

→ **See notebook: SW04-missing-values-TUT.ipynb**

Weekly Moodle quizzes: Updates

- After each lecture, test questions will be posted on Moodle
- The quizzes are open-book, you can work on them at any time
- Deadline for every quiz is the following **Monday at 20h**
- The final grade for all quizzes is calculated as follows:

$$\text{Grade} = 5 \times \frac{\text{Total points achieved}}{\text{Total points possible}} + 1$$

- As a rule, no excuses will be accepted after the deadline
- Each student can **miss a quiz once** (joker)

Exploratory data analysis

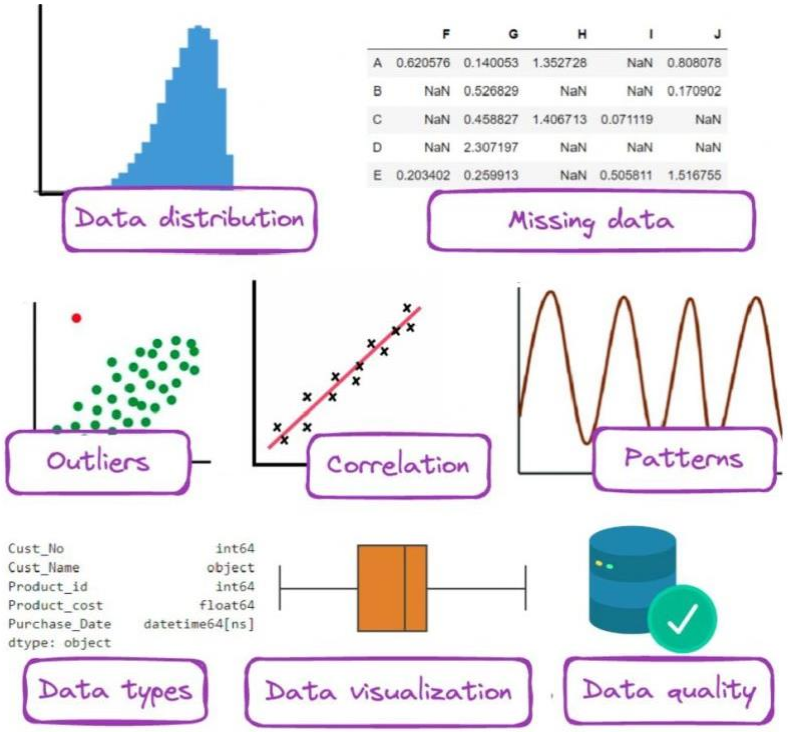
Look at your data!

Purpose of exploratory data analysis (EDA)

- **Understand the data's structure**: Identify patterns, relationships, trends, and anomalies in the dataset.
- **Detect missing or incorrect data**: Uncover issues such as missing values, outliers, or inconsistencies that need to be addressed.
- **Guide feature selection and engineering**: Inform decisions about which features are important or need transformation.
- **Generate hypotheses**: Formulate potential hypotheses and insights for further analysis or modeling.
- **Validate assumptions**: Check if the data fits certain assumptions required by statistical or machine learning models.

Means of EDA

- Summary statistics
- Data visualization
- Correlation analysis (between features)
- Univariate analysis (between features and target)



Data summary

- For **all** variables
  - Number of missing values — `np.isnan(x), pd.isna(x)`
  - Number of unique values — `np.unique(x), pd.unique(x)`
- For **scalar** variables
  - Measures central tendencies (mean, median) — `np.mean(x), np.median(x)`
  - Measures for disparity/spread
    - Standard deviation — `np.std(x)`
    - Range [min, max] — `np.min(x), np.max(x)`
    - Interquartile range [25th percentile, 75th percentile] — `np.iqr(x)`
- For **categorical** variables
  - List of distinct categories — `np.unique(x, return_counts=True)`
  - Absolute and relative frequencies of categories — `pd.value_counts(x)`, `pd.value_counts(x)/len(x)`

Data visualization

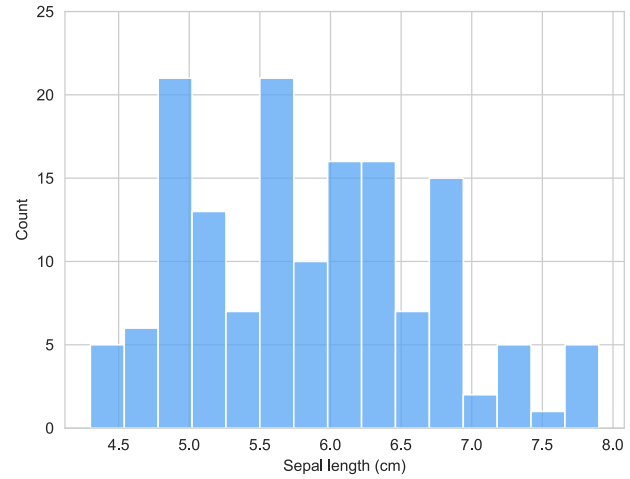
Look at your data once more!

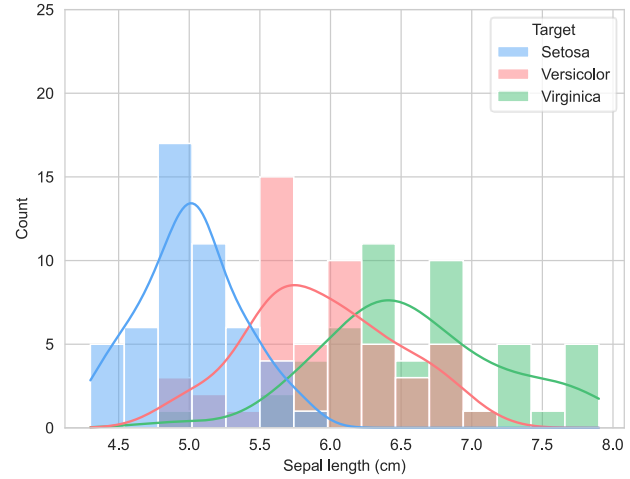
Data visualization

- **Histograms**: Visualize the distribution of a single variable.
- **Density plots**: Show the distribution of a variable in a smoothed manner.
- **Box plots**: Detect outliers and understand the spread of data.
- **Scatter plots**: Explore relationships between two scalar variables.
- **Pair plots**: Analyze pairwise relationships in the dataset.
- **Bar plots**: Compare categorical variables.
- **Heatmaps**: Visualize correlations between features.

Histograms

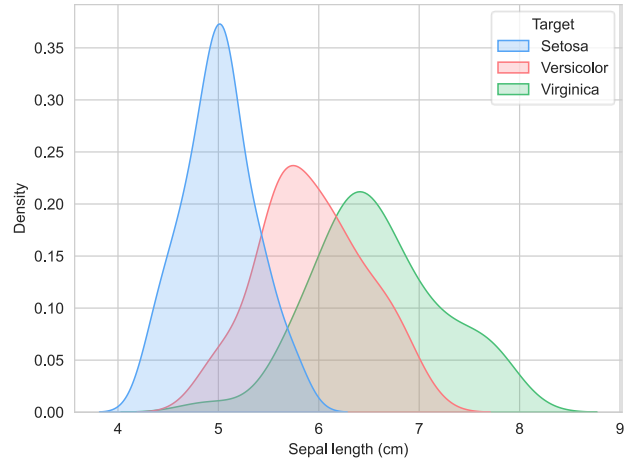
- Visualize the distribution of values by dividing data into bins and counting the data points in each bin
- Assess spread and range of the distribution
- Identify skewness and symmetry properties
- Detect outliers: Identify extreme values
- Are the individual distributions separable?
- Can be used to compare distributions





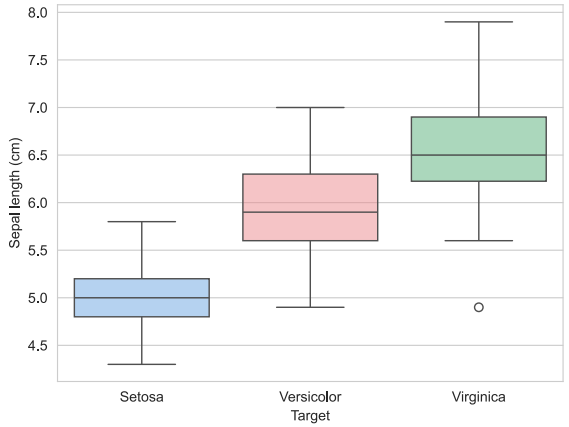
Density plots

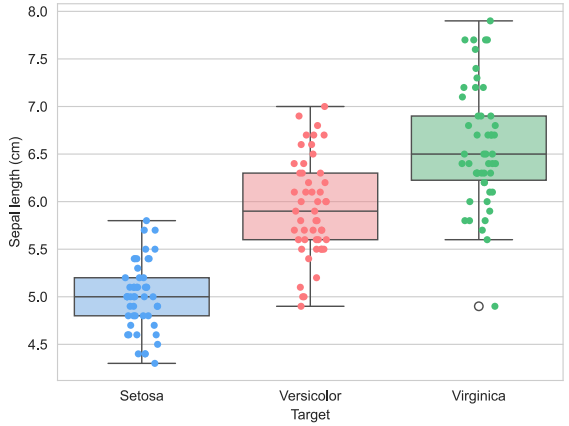
- Visualize the distribution of values using smooth curves estimated from the data (kernel density estimation, KDE)
- Assess spread and range of the distribution
- Identify skewness and symmetry properties
- Detect outliers: Identify extreme values
- Are the individual distributions separable?
- Can be used to compare distributions



Box plots

- Graphical representation summarizing the distribution of dataset through five key descriptive statistics:
  - **Minimum**: 0th percentile
  - **First quartile**: 25th percentile, the lower end of the box
  - **Median**: 50th percentile, central line inside the box
  - **Third quartile**: 75th percentile, the upper end of the box
  - **Maximum**: 100th percentile
- Important elements:
  - **Interquartile range**: Defines the extent of the box
  - **Whiskers**: Smallest and largest values that are within 1.5 times the IQR (identifies inliers)
  - **Outliers**: Data points outside the whiskers
- Recommendation: Overlay with a scatter plot to get a better feeling for the data (for n<1000 samples)

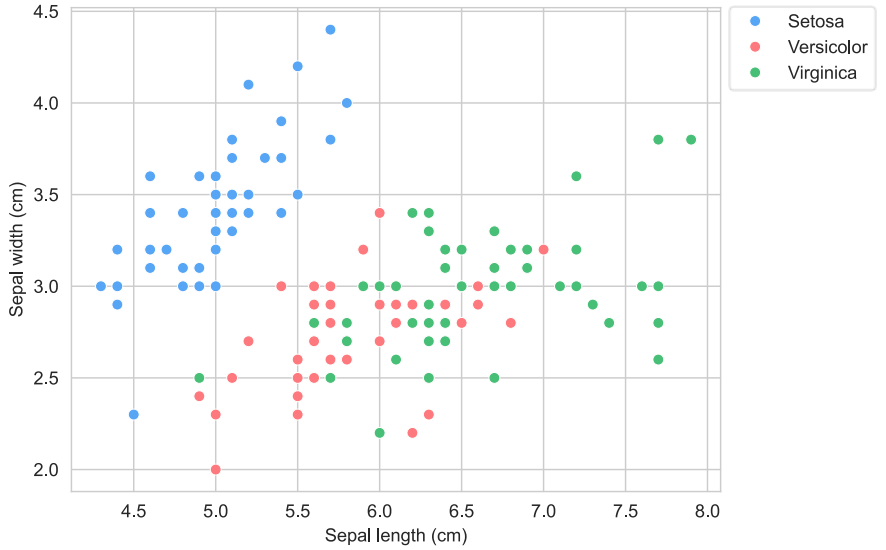




Boxplots for a dataset stratified by the categorical target variable, overlayed with a scatter plot showing the actual data points.

Scatter plots

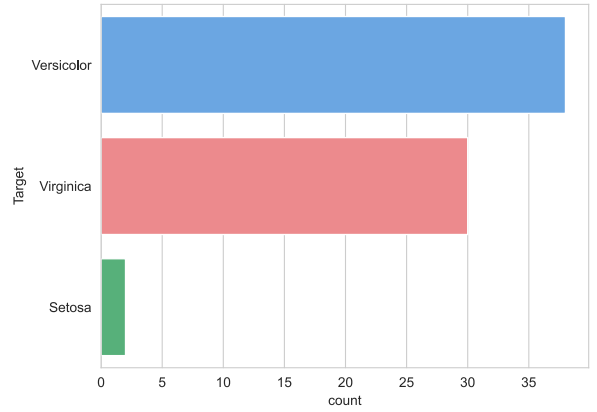
- Plot two variables (features or target variables) against each other
- Useful for
  - Relationship exploration
  - Identifying trends
  - Uncovering clustering patterns

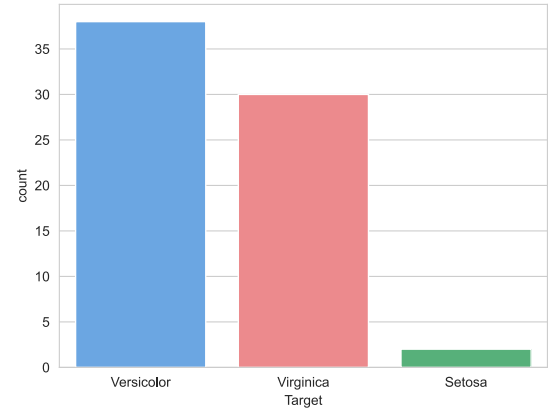


Scatter plot between two feature variables, stratified by the target variable. We can see that with these two features alone, we should be able to distinguish quite reliably between the blue (Setosa) and the other class.

Bar plots

- Visualize a characteristic value per category
- Useful for
  - Displaying counts or frequencies in categorical data
  - Comparing numerical (summary) values for multiple categories
  - Highlight differences





Horizontal and vertical bar plots displaying the absolute frequency (i.e., counts) of the categorical target variable (in a modified version of the iris dataset).

Data visualization

- For more visualization ideas, consider the following sources:
  - From Data to Viz (with a beautiful **decision tree**)
  - The Python graph gallery
  - The R graph gallery (for R / ggplot2)
  - Seaborn examples gallery
  - Plotly express / Plotly graphing examples

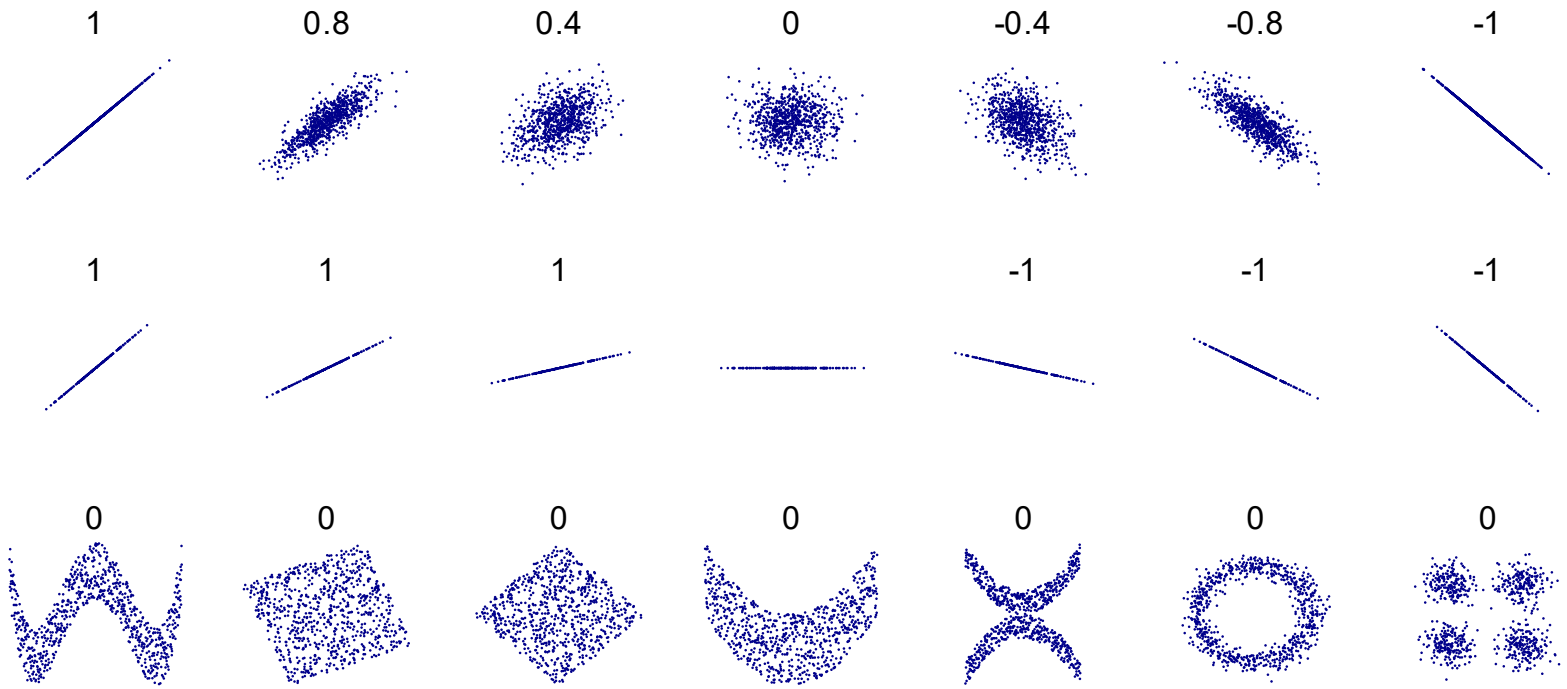
→ See coming notebook for examples...

Correlation analysis

Correlation analysis: Concept

- Another approach to explore how variables relate to each other
- Correlation analysis is used to determine the strength and direction of a *linear* relationship between two (random) variables X and Y.
- It involves a score, the **correlation coefficient ρ**, with the following properties:
  - The correlation coefficient is a value that ranges between -1 and 1
  - A value of 1 indicates a perfect positive correlation
  - A value of -1 indicates a perfect negative correlation
  - A value of 0 indicates no correlation between the variables
- Goal:
  - Identify variables (or cluster of variables) that depend on each other.
  - We can use this information to reduce redundancy in the data:
    If two features X_1 and X_2 are strongly correlated, we may remove one of them
    to reduce the dimensionality of the data, as they provide redundant information.

Correlation: Illustration



Several sets of (x, y) points, with the (Pearson) correlation coefficient of x and y for each set. The correlation reflects the noisiness and direction of a linear relationship (top row), but not the slope of that relationship (middle), nor many aspects of more complex, nonlinear relationships (bottom). Note: the figure in the center has a slope of 0 but in that case, the correlation coefficient is undefined because the variance of Y is zero.

Source: Wikipedia

Correlation: Mathematical definition

- Mathematically, the (Pearson) correlation coefficient for two random variables X and Y is defined as follows:

$$\rho = \frac{\mathrm{cov}(X, Y)}{\sigma_X \sigma_Y}$$

- Here, $\mathrm{cov}(X, Y)$ represents the covariance between X and Y.
  It measures how much the variables change together:

$$\mathrm{cov}(X, Y) = \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X}_1)(Y_i - \bar{Y})$$

- By scaling the covariance with the standard deviations of X and Y, we yield a coefficient that takes values between -1 and 1.

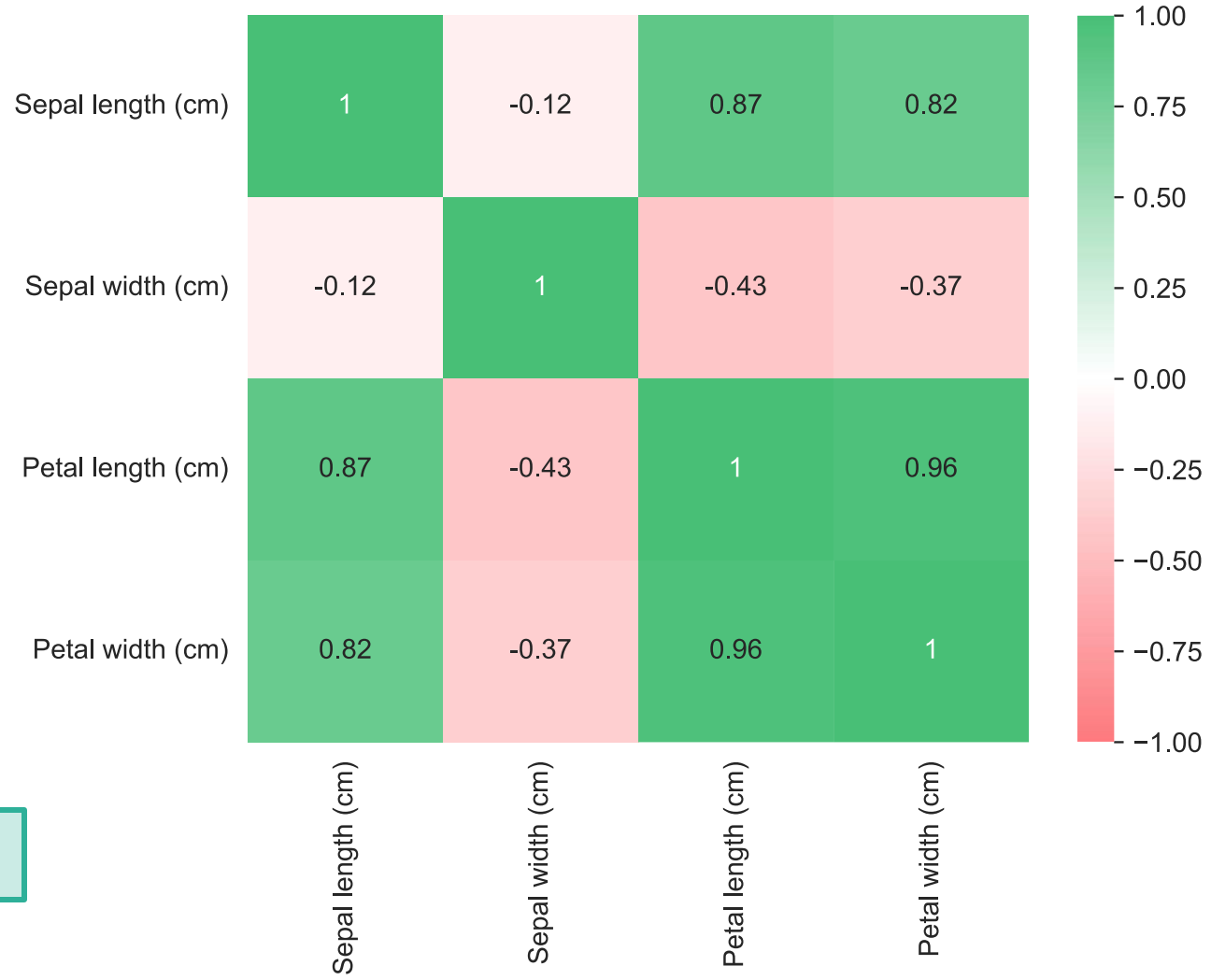
$$\sigma_X = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})^2} \qquad \sigma_Y = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (Y_i - \bar{Y})^2}$$

Correlation analysis in practice

- Luckily, we usually do not have to bother about these formulas! ☺
- We can compute the pairwise correlation between features using the df.corr() method of a Pandas DataFrame.
- This yields the so-called **correlation matrix**
- We can graphically visualize the correlation between features
  - illustrating the correlation matrix using a heatmap
  - using pair plots

Correlation matrix / heatmap

- Visualize the values of the correlation matrix using a color map
- See the strength and direction of the (linear) relationship between two variables

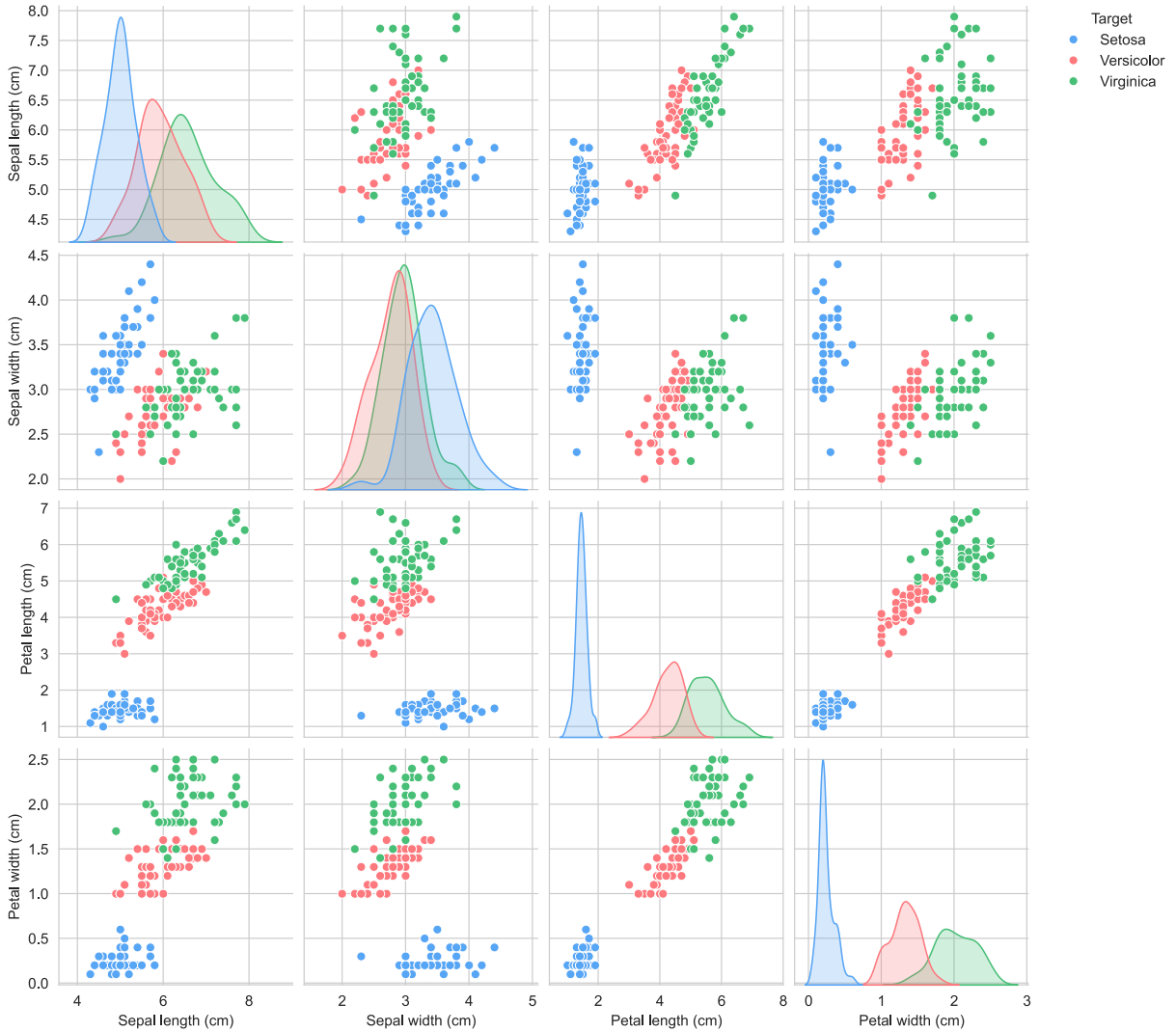


Why are the diagonal entries always 1?

Pair plot (or scatter plot matrix)

- A grid of plots showing pairwise relationships between variables in a dataset.
- **Scatter plots** for variable pairs: Each off-diagonal plot shows the scatterplot between two different features X_i and X_j with i≠j.
- **Density plots** (or histograms) in the diagonal, showing the value distribution of the features

How would the scatter plot look like between X_i and X_j if i=j?



Univariate analysis

Univariate analysis: A brief outline

- Focus on analyzing a single variable or feature at a time
- For each feature
  - Describe the variable (mean, standard deviation, etc.)
  - Visualize the distribution of values
  - Examine the univariate relationship with the target variable
- This may include statistical tests!

Shortcuts and tools

Automated exploratory data analysis

Dataset summary using pandas

- Pandas DataFrames offer the df.describe() method to generate basic descriptive statistics

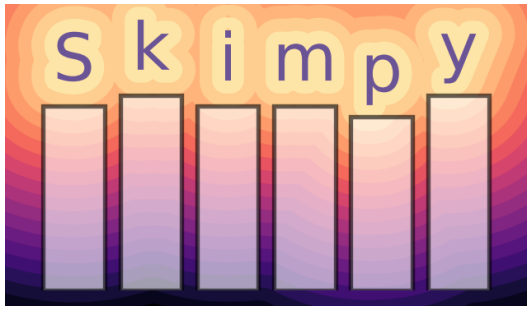
```
import pandas as pd
from sklearn.datasets import load_iris

df = load_iris(as_frame=True).frame
df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

Dataset summary using skimpy

- Skimpy: A light-weight Python package for creating summary statistics from Pandas DataFrames.
- A supercharged version of df.describe()




```
# Make sure to previously install the skimpy package.
# %pip install skimpy

from skimpy import skim
from sklearn.datasets import load_iris

df = load_iris(as_frame=True).frame
skim(df)
```

skimpy summary

Data Summary

dataframe	Values
Number of rows	150
Number of columns	5

Data Types

Column Type	Count
float64	4
int64	1

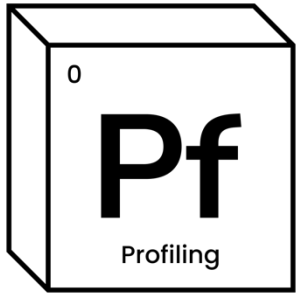
number

column_name	NA	NA %	mean	sd	p0	p25	p50	p75	p100	hist
sepal length (cm)	0	0	5.843	0.8281	4.3	5.1	5.8	6.4	7.9	
sepal width (cm)	0	0	3.057	0.4359	2	2.8	3	3.3	4.4	
petal length (cm)	0	0	3.758	1.765	1	1.6	4.35	5.1	6.9	
petal width (cm)	0	0	1.199	0.7622	0.1	0.3	1.3	1.8	2.5	
target	0	0	1	0.8192	0	0	1	2	2	

End

EDA using ydata-profiling

- ydata-profiling: A Python package that provides a simple way to automatically **profile data** and **generate reports**.



- **Features**:
  - Summary statistics
  - Data visualizations
  - Missing data analysis
  - Outlier detection
  - Correlation analysis
  - Detection of duplicates
  - Interactive and customizable reports

EDA using ydata-profiling

- ydata-profiling: A Python package that provides a simple way to automatically **profile data** and **generate reports**.

```
# Make sure to previously install the package.
# %pip install -U ydata-profiling

from sklearn.datasets import load_iris
from ydata_profiling import ProfileReport

df = load_iris(as_frame=True).frame

profile = ProfileReport(df,
title="Report: Iris dataset",
sort=None,
sensitive=False,
explorative=False)

# Create and display the report
profile.to_notebook_iframe()   # Integrate into a notebook
profile.to_widgets()           # Integrate into a notebook, compact
profile.to_file("output.html") # Save the report to an HTML file
```