# Machine Learning I

## Chapter 2 - Data Preprocessing

Prof. Dr. Sandra Eisenreich

WS 2024/25

Hochschule Landshut

I. Data Preprocessing: Collect and clean up data.

II. Choosing/Training Model(s): see below.

III. Evaluating the Model(s): Check the performance on new, unseen data.

This is a summary of the steps done in the Data Preprocessing Notebook plus a little theory.

# Data Challenges in ML

**Insufficient quantity of data**
solutions: get more data, create more data (simulations, generative AI), use unsupervised or semi-supervised machine learning (ML models which doesn't need labels; see soon)

**Poor quality data**: e.g. due to sensor malfunction, human error, poor calibration, false data...
solutions:

- Find outliers and errors and remove them from the dataset.
- Find missing values and drop these data or replace the missing values.

# Data Challenges: Irrelevant or insufficient features

Irrelevant information in the data can "confuse"the model, insufficient information can make good prediciton impossible.
solutions:

- Feature selection is the process of selecting the most useful features.
- Feature extraction is the process of combining features to produce more useful ones.
- Gather more data.

# Data Challenges: Class/Data Imbalance

e.g. class imbalance: if a feature is categorical and the amount of data from each category is unbalanced. Solutions (see later): Balancing the classes by

- Subsampling: Choosing only a subset of the data for overrepresented classes
- Oversampling: Supplementing the underrepresented classes with multiple copies of instances.

# Data Challenges: Nonrepresentative Data

- Too few data: If you only flip a coin twice and the result is always head, your model will predict 100% head results. This effect is called sampling noise.
- Biased data from only a fraction of cases:
  - e.g.: If you conduct a poll before a vote and only question people who play golf, you will not get a result representative of the entire population. This effect is called sampling bias.
  - e.g.: If you conduct a poll about your product by email, you will probably mainly get responses from those who care strongly about the matter (five stars or no stars), you won't get the opinion of the average customer. This is called nonresponse bias.

# Steps of data preprocessing

- Getting to know the data; visualizations (Data Science 2)
- Feature engineering
- Creating a train-test split (see later)
- Data Cleaning
- Handling Categorical Attributes
- Feature Scaling
- Creating class balance, if necessary

You often combine the last four in a Transformation Pipeline. See the jupyter Notebook on Data Preprocessing for details.

# Step 1: Getting to know data; Visualizations

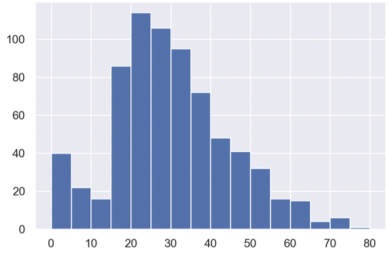
Possible steps:

- 1.1 Get to know the data structure: features (df.head()), non-null and null values (df.info(), df.isnull().sum()), statistical values (df.describe())
- 1.2 Plot Feature distributions and check class balance: plot a historgram of numerical features (df.hist()) - what are the distributions?, plot categorical values - class (im)balance?
- 1.3 Correlations and Scatterplots:
  - correlation matrix: how strongly are features **linearly** related?
  - scatterplots: plot the relationship (indepentent of linear or not) between features.
- 1.4 Gain insights wrt target value: Focus on the feature you want to predict for further visualisation.

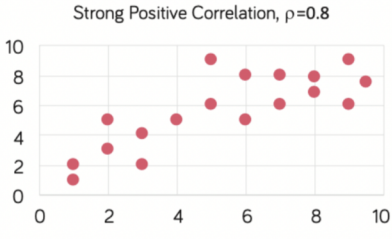
# Remember: Histograms

**Histograms**:

- divide the data into discrete bins;
- height of each bar = number of instances in the bar/width of the bar (theory); in pandas/matplotlib: number of instances



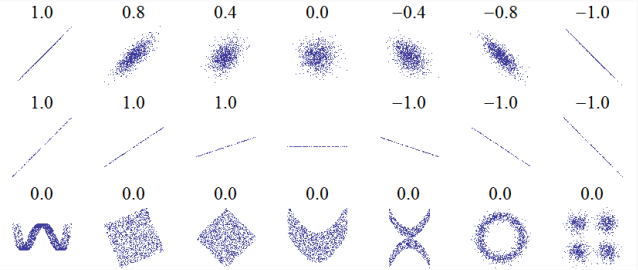

**Scatterplots**:
to visualize relationship between features.
To compute how linearly correlated features are use Pearson's correlation value.



# Remember: feature correlations

Correlation = a **linear** relationship between two features. **Pearson's correlation coefficient** describes the strength of the correlation.

- close to 1/-1: strong correlation with positive/negative slope
- close to 0: weak/no correlation (there still can be a relationship, just not a linear one)




Quelle: Wikimedia Commons

Example of a correlation matrix (correlation coefficients for each pair of features):

| | a | b | c |
|---|---|---|---|
| a | 1 | 0.7 | -0.1 |
| b | 0.7 | 1 | -1 |
| c | -0.1 | -1 | 1 |

# 2nd step: Feature Engineering

- Feature selection: Drop irrelevant features, if there are any, or those with too many missing entries to be of use.
- Feature extraction: combine existing features to produce a more useful one. Experiment with attribute combinations and find out if they are useful by repeating step 1 (Data Visualizations, correlation matrix).

# 3rd step: Train-Test Split and Input-Label Split

- Divide data into training and test set BEFORE cleaning or transforming data in a way that might mix information from the train and test set (like computing the mean or median or most frequent category), to keep the model unbiased towards the test data.
- Always permute data before the split (to mix classes)
- Use a stratified split to make sure classes are evenly distributedïnto test and train set (i.e. the data distribution is the same as in the overall data set).
- Split the target feature from the rest of the dataset.

# 4th step: Data Cleaning

- drop irrelevant features,
- clean missing values by either dropping instances/features or imputing "good" values like
  - for numerical features: mean or median or a constant value
  - for categorical features: most frequent category or a constant value

  Alternatively, one could train an additional model predicting missing values.
- Get rid of outliers (via box plots or the underlying calculations, or anomaly detection, see later).

# 5th step: Handling Categorical Features

like (Green, red, . . . , yellow...). Two possibilities:

- Ordinal encoding: Convert the categories to numbers (Green $= 1$, Red $= 2$, . . . Yellow $=$ C). Disadvantage: only makes sense for ordinal data (with a natural ordering)
- One-hot encoding: Convert the $C$ categories to $C$-dimensional one-hot vectors (Green $= e_1$, Red $= e_2$, . . . )

# 6th step: Feature Scaling

Most ML algorithms perform badly when the numerical attributes have different scales.

Solutions:

- Min-max scaling: values are shifted and rescaled so that they end up in the range of 0 to 1, by substracting the min value and dividing by the max minus the min:

$$x \mapsto \frac{x - \min}{\max - \min}$$

- Standardization: substract the mean value $\mu$ and divide by standard deviation $\sigma$ so values have mean 0 and 1 variance.

$$x \mapsto \frac{x - \mu}{\sigma}$$

**Important:** If you preprocess training data one way, it's important to do exactly the same to the test data, i.e. don't compute mean and deviation or min/max of the test data for standardization, but those of the *training data*!

# 7th step (ONLY for training data): Creating class balance if necessary

If there is a major class imbalance (particulary in the labels), create class balance for the target values only on the training set via under- or oversampling.

# Transformation Pipelines, Column Transformer

To keep it simple, combine 4-6 into a pipeline which deals with numerical and categorical data separately:

- divide features into numerical and categorical features
- define a transformation pipeline for numerical features (impute e.g. mean or median for missing data, scale data)
- define a transformation pipeline for categorical features (impute e.g. most frequent category, encode categories)
- combine the two pipelines in a single Transformer.

# How to perform 4-6 on test data

Note: We want to transform the test data with the parameters (e.g. mean or median or standard deviation) fit on the **training data**, not the test data! Therefore, the process is as follows:

- Define a transformation pipeline for steps 4-6
- Fit-transform it on the training data
- **only transform** (not fit!) the test data with trained pipeline.

# Chapter II Summary: Data Preprocessing Steps

- Getting to know the data and visualizations: look at data and summary statistics, plot histograms and scatterplots, compute correlation matrix.
- Feature engineering: feature selection, feature extraction (create new features from old )
- Create a train-test split (after a random permutation of the dataset): simple split or stratified split (equal distribution of categories).
- Data Cleaning: drop columns/instances with missing values, or insert "good"values for missing values:
  - for numerical features: mean or median or constant
  - for categorical features: most frequent category or constant
  - alternative: prediction model to insert missing data
- Handling Categorical Attributes: ordinal or one-hot encoding.
- Creating class balance, if necessary: If there is a major class imbalance (particulary in the labels), create class balance for the target values only on the training set via under- or oversampling.
- Feature Scaling to make all features have similar size and variance: MinMaxScaling or Standardization.
- Important: use the transformation parameters from the training set (mean, standard deviation, most-frequent-categories...) for the transformation of the test set.