Life Sciences and
Facility Management

Institute of
Computational Life Sciences

Data problems

Machine Learning

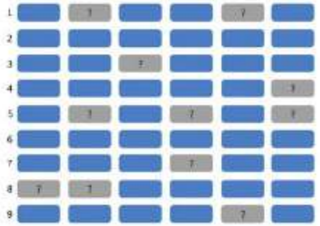
Norman Juchler

Potential problems?

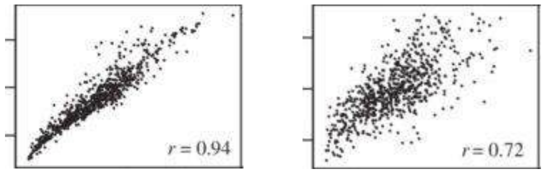
...with the data

Potential problems with data

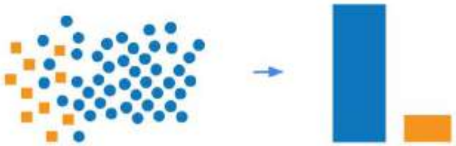
- **Missing data** Data points may be missing → Samples may have to be excluded, as some ML algorithms cannot handle it, which may lead to selection biases.



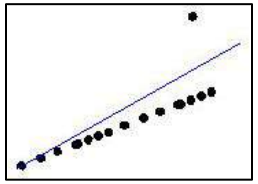
- **Noisy data** Data may contain irrelevant, imprecise or erroneous entries that introduce noise → harder for a model to learn meaningful patterns.



- **Imbalanced data** A class or outcome may be overrepresented in the dataset → classification models may be biased toward the dominant class and fail to detect rare events.



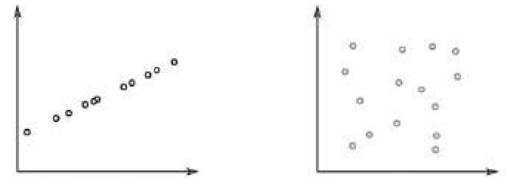
- **Outliers** Unrealistic or extreme values → can skew or destabilize the model's learning process.



- **Duplicated data** Some samples are overrepresented → model may give too much weight to certain patterns.

Potential problems with data

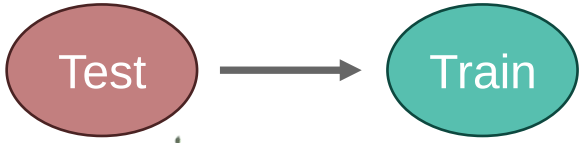
- **Irrelevant or redundant features** Features that do not contribute to the target variable or are highly correlated with other features → reduced model performance or overfitting



Perfect correlation ⇔ Redundancy

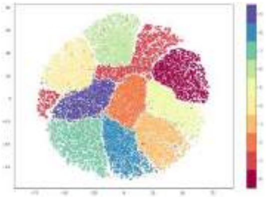
No correlation ⇔ No redundancy

- **Data leakage** Test data may overlap with training data → unrealistic performance estimates, data overfitting goes unnoticed



- **Inconsistent data** Data from different sources or time periods may be inconsistent → model assumptions don't hold for the entire dataset

- **High dimensionality** If dataset has too many features compared to the number of samples → curse of dimensionality, model more prone to overfitting



Projection of high dimensional data onto a lower dimensional embedding (t-SNE)

- **Unstructured or unformatted data** Raw, unstructured data (e.g., text, images) can be difficult to process directly → preprocessing required

20241008-101026 Tue 8/10 2024 10:10AM 1728374982.745886	→	08.10.2024 10:10 08.10.2024 10:10 08.10.2024 10:10

Know where your data comes from!

Possible issues resulting from unclear data provenance:

- **Biases**
  - Sample bias: The data might not be representative.
  - Exclusion bias: Important data is left out of the analysis.
  - Observer bias: See what you expect to see in the data.
  - Prejudice bias: Models highlights stereotypes and/or faulty social assumptions.

Example: X-rays for babies are taken differently than for adults.

- **Confounding factors** Potentially unknown factors that influence both the features and target variable.

Example: People who smoke have a higher risk to suffer from liver cirrhosis. However, this is not due to a direct biological effect of smoking on the liver but because smokers, on average, also drink more alcohol than non-smokers.

Know where your data comes from!

Possible issues resulting from unclear data provenance:

- **Data set merging** If data sets are merged in several steps, they might contain the same samples several times.
- **Hidden clues** Metadata (which are accessible to the ML-model) may contain hidden clues that correlate with the target variable.
- **Derived ground truth** The use of ground truth, which was itself derived from the samples instead of independently assessed, is problematic
- **Undocumented preprocessing** Preprocessing can remove or introduce correlations in the data.

Examples:
- X-ray image names may contain a clue about the patient's condition
- The X-ray device from emergency unit creates an artefact on the top left corner, which is not the case for X-ray images of other units

Example: A radiologist creates target labels by looking at the same X-ray images that will be used by the ML model.

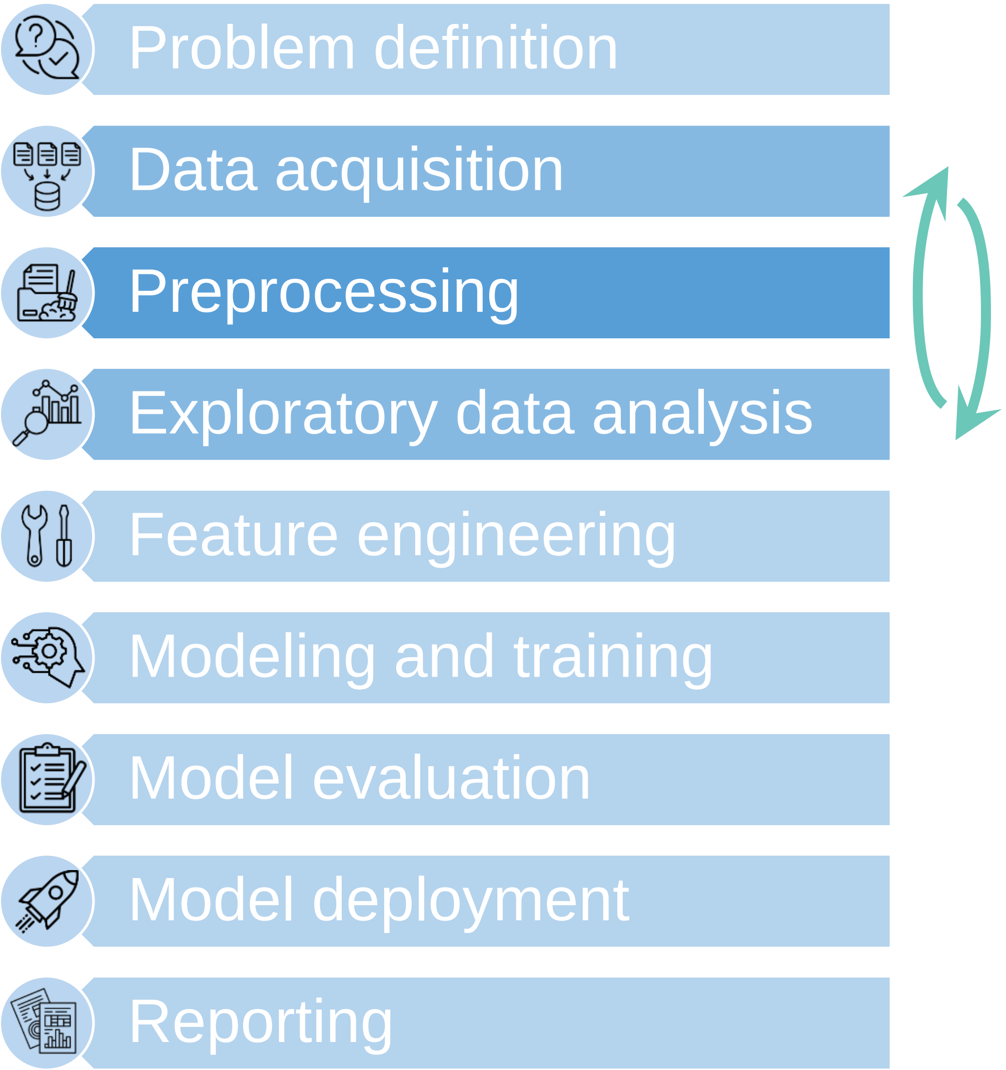
Summary: The main (data) challenges of machine learning

- Insufficient quantity of training data
- Poor-quality training data
- Non-representative training data
- Irrelevant features
- Overfitting the training data
- Underfitting the training data

Insufficient data

How to deal with it?

How to deal with data problems?



- Clean, transform, and organize the data to ensure that it's suitable for modeling and analysis.
- This may involve
  - Handle missing data, duplicates, and outliers.
  - Convert data into a usable format (e.g., handling dates, strings, and categorical variables).
  - Normalize or standardize data if necessary.
  - Remove or correct inconsistent or erroneous entries.

Challenge: Limited data quality

- How to know it is a problem?
  - Identify low-quality samples (by looking at the data)
  - Observe a high model bias (when using the model)

- Approaches to improve data quality:
  - Denoising the data
  - Removing bad samples
  - Improving the data labeling process

Approaches to ensure quality on the level of database

Challenge: Limited amount of data

- How to know it is a problem?
  - When plotting performances versus training set size shows strong increase.
  - Reason: The model can still benefit from seeing additional data. If enough data is available, learning flattens out.

- Approaches to deal with insufficient training data:
  - Collection of additional training data
  - Reduction of model complexity
  - Data augmentation
  - Introduction of noise into existing samples
  - Transfer learning

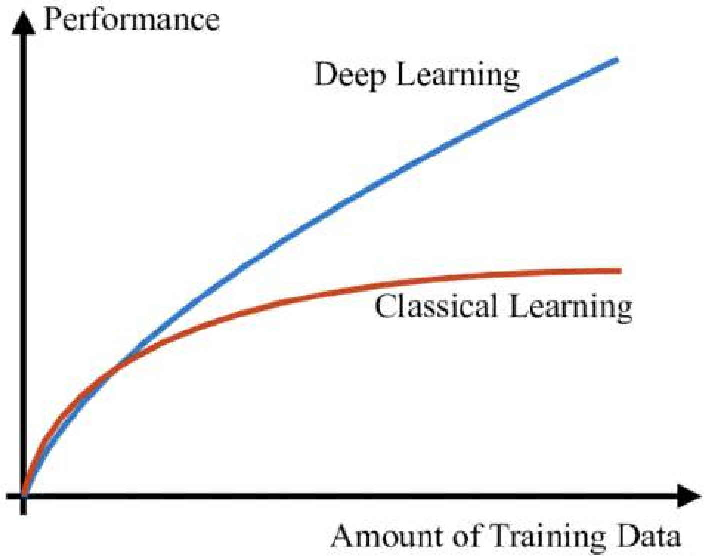


Illustration: Classical machine learning approaches tend to require a lower amount of data, as their “learning curves” flatten out earlier.

Data augmentation

- Artificially increase the size and diversity of the training data.
- How: Add slightly modified copies of already existing data.
  - Apply transformations to input data (like rotations, flips, scaling for images)
  - Add noise to the original data
  - Generate synthetic data using generative models
- Goals:
  - Prevent overfitting
  - Improve model generalization / enhance performance on unseen data

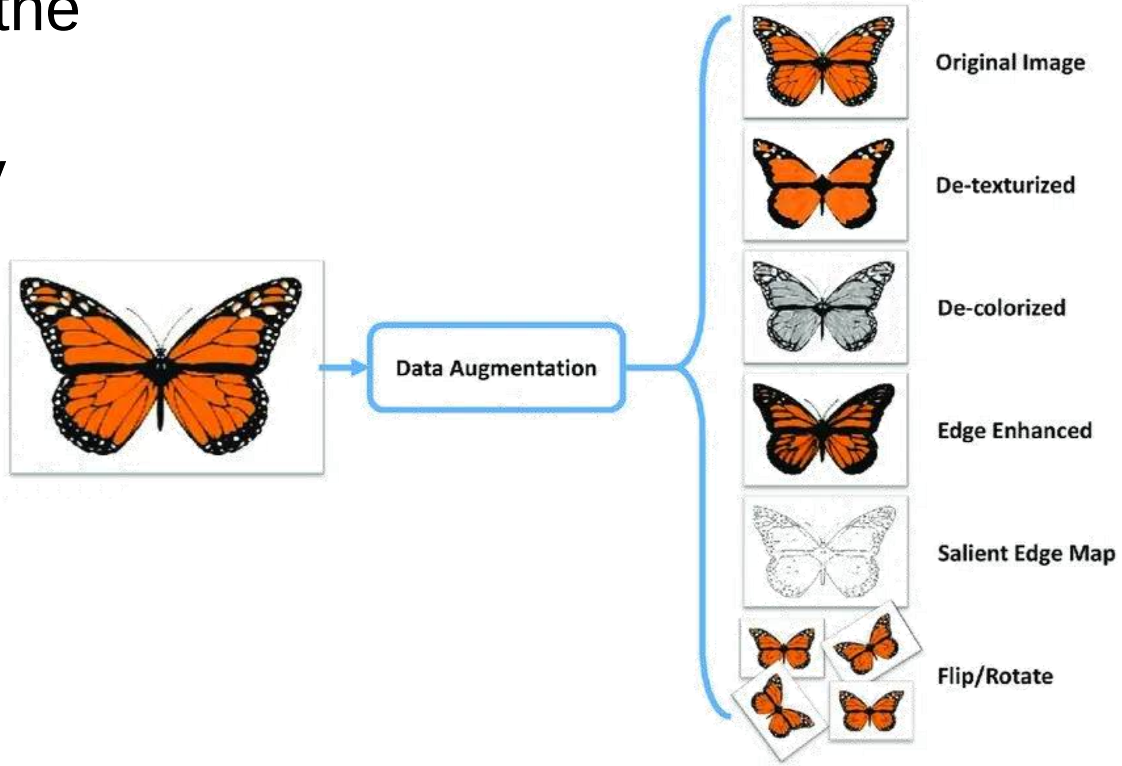
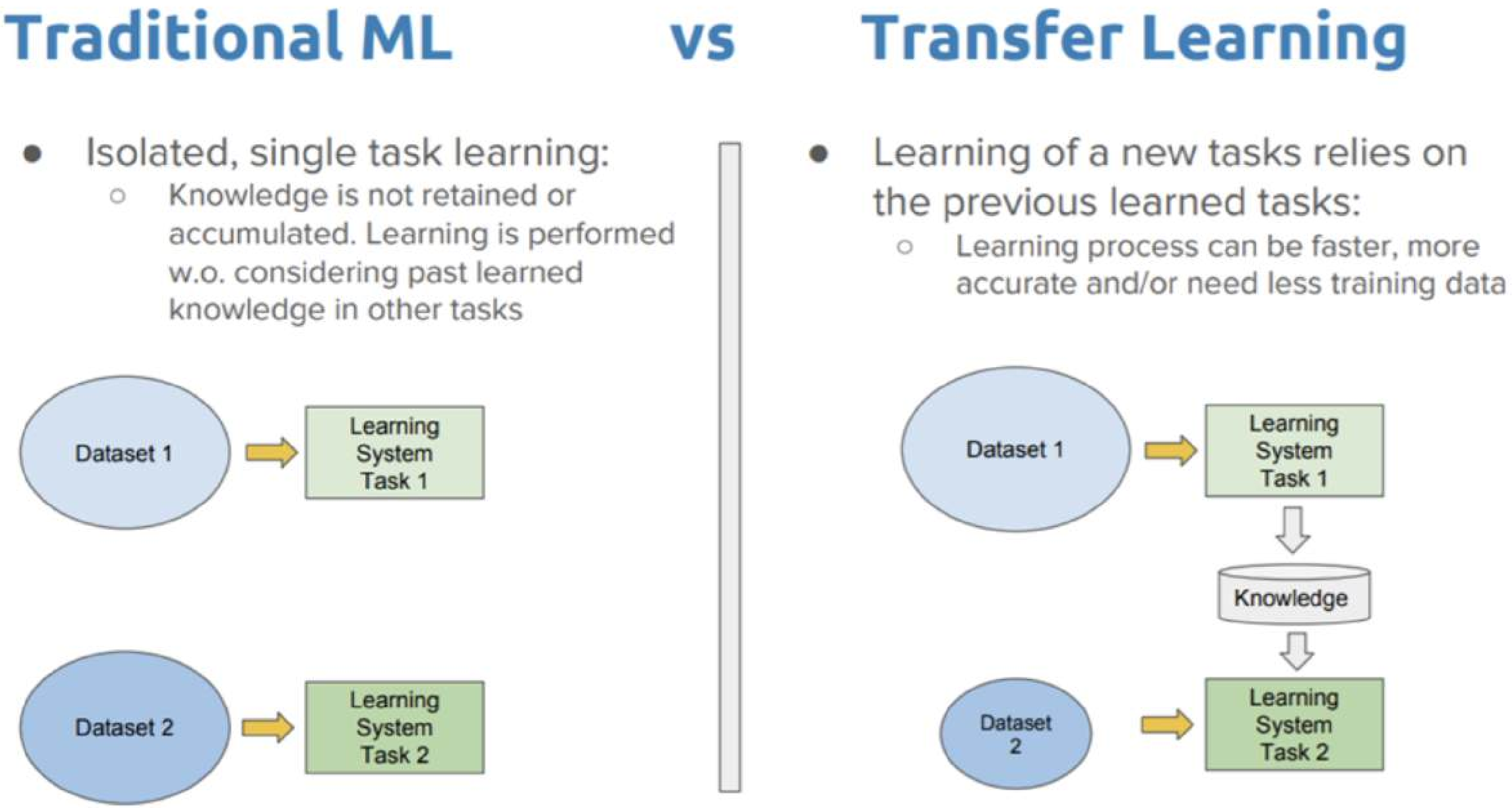


Image source: Link

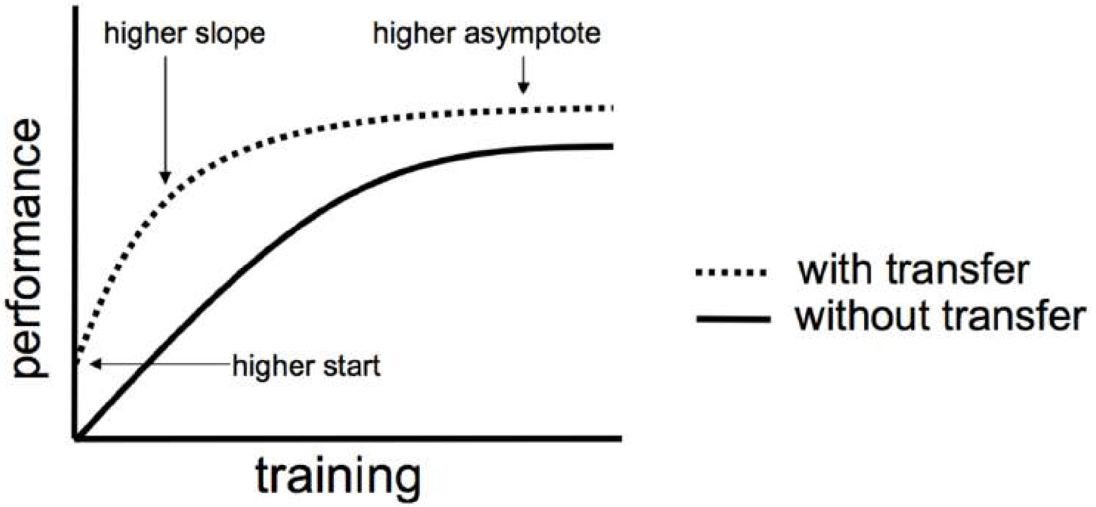
Transfer learning



Transfer learning

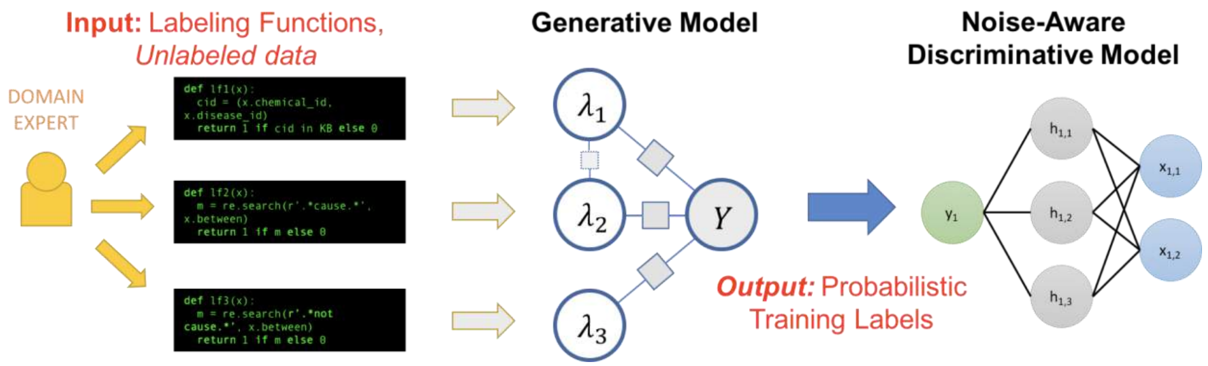
- Transfer learning allows rapid progress or improved performance for the second task (for which we may have only a small data set).

- Ways to transfer knowledge:
  - Pre-trained models
  - Fine-tuning on new data
  - Input Embeddings



Weak supervision

- Idea: Use noisy, or imprecise sources to provide supervision signal for labeling large amounts of training data.
- It reduces the requirements on the training data
  - Reduce amount of hand-labeled (expensive) samples
  - Increase the amount of available training data



- Approaches:
  - Data programming (Snorkel)
  - Confident learning (CleanLab)

Preprocessing

Techniques and methods

Encoding of categorical features: Ordinal encoding

- **Problem**: Many machine learning methods require numerical data. However, we might encounter categorical predictors in our data.
- **Solution**: Convert categorical variables into a numerical format by mapping each each category onto a numerical value

SAFETY-LEVEL (TEXT)	SAFETY-LEVEL (NUMERICAL)
None	0
Low	1
Medium	2
High	3
Very-High	4

Encoding of categorical features: One-hot encoding

- **Problem**: Many machine learning methods require numerical data. However, we might encounter categorical predictors in our data.
- **Solution**: Convert categorical variables into a numerical format by mapping each each category onto a binary vector (a.k.a. **dummy variables**).
- **Issues**:
  - **Problem**: A mapping of m categories to m features Introduces collinearity.
    Solution: Only use m-1 variables
  - **Problem**: Increases the dimensionality of the feature space (especially for many categories)
    Solution: Reduce categories, or apply dimensionality reduction techniques

Source / good read: https://www.scaler.com/topics/data-science/one-hot-encoding/

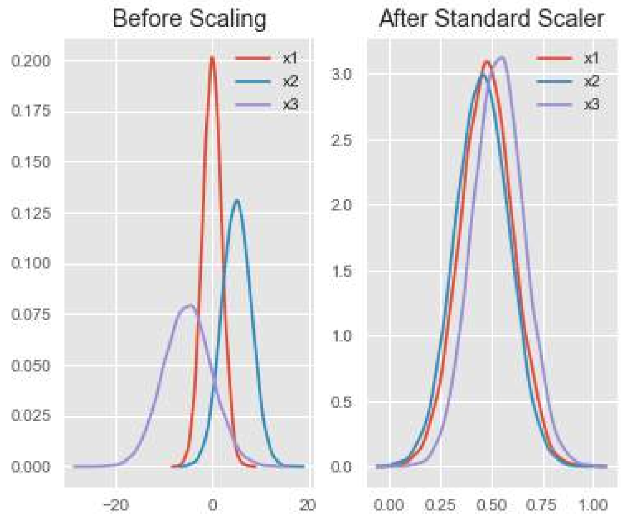
Preprocessing: Feature scaling

- **Problem**:
  - Some algorithms are sensitive to the scale of the data.
  - Features with different ranges can dominate the learning process, leading to biased models.
  - Numerical problems may arise for improperly scaled features, affecting the convergence speed for optimization algorithms (like gradient descent)
- **Solution**: Normalize or standardize features
- **Methods**:
  - **Min-Max scaling** (normalization):
    - Rescale the features such that all values are in the range [0, 1]

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

  - **Z-score normalization** (standardization):
    - Rescale the features such that the features have a mean μ=0 and a standard deviation σ=1

$$X_{scaled} = \frac{X - \mu}{\sigma}$$



Preprocessing: Feature scaling

```
# Example for MinMaxScaler
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = MinMaxScaler()
scaler.fit(data)

# Parameters of the MinMaxScaler
print(scaler.data_max_)
# >>> [ 1. 18.]

# Apply the transformation to the data
print(scaler.transform(data))
# >>> [[0. 0. ]
#      [0.25 0.25]
#      [0.5 0.5 ]
#      [1. 1. ]]
```

Source / good read: https://www.i2tutorials.com/what-is-feature-scaling/

Find and remove duplicates

- Duplicate samples may:
  - Drive the learning in unintended directions
  - Invalidate your performance measure

In Python/Pandas:

- Find duplicates:

```
>>> df.duplicated()
```

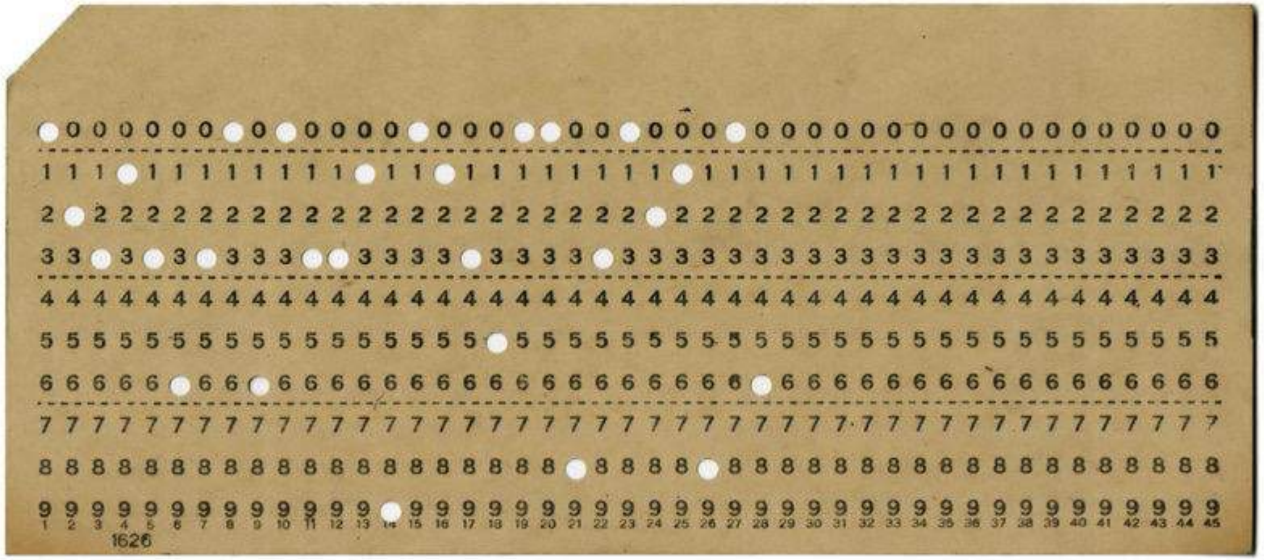
- Remove duplicates:

```
# Use keep='last' to keep the last occurrence
df.drop_duplicates(keep='last')
```

Finding duplicates can be tricky for certain types of data or problems....

Handle missing values

- Most ML algorithms will not work if data entries are missing or give wrong results!
- **Types of missing data**
  - Missing at random
  - Missing not at random



Handle missing values

- Most ML algorithms will not work if data entries are missing or give wrong results!

- **Types of missing data**
  - Missing at random
  - Missing not at random

- **Option 1: Removal**
  - Entire sample
  - Entire feature

- **Option 2: Imputation**
  - Replace with a default value
  - Replace with average
  - Replace with a predicted value
    - via regression models
    - via similarity (hot-deck imputation)

Column 0	age	years_seniority	income	parking_space	attending_party	entree	pets	emergency_contact
Tony	48	27		1	5	shrimp		Pepper
Donald	67	25	86	10	2	beef		Jane
Henry	69	21	95	6	1	chicken	62	Janet
Janet	62	21	110	3	1	beef		Henry
Nick		17		4				
Bruce	37	14	63		1	veggie		NA
Steve	83		77	7	1	chicken		n/a
Clint	27	9	118	9		shrimp	3	None
Wanda	19	7	52	2	2	shrimp		empty
Natasha	26	4	162	5	3			_
Carol		3	127	11	1	veggie	1	""""
Mandy	44	2	68	8	1	chicken		null

Handle missing values

- Most ML algorithms will not work if data entries are missing or give wrong results!

- **Types of missing data**
  - Missing at random
  - Missing not at random

- **Option 1: Removal**
  - Entire sample
  - Entire feature

- **Option 2: Imputation**
  - Replace with a default value
  - Replace with average
  - Replace with a predicted value
    - via regression models
    - via similarity (hot-deck imputation)

```python
# We can use the SimpleImputer class from sklearn to impute missing values.
from sklearn.impute import SimpleImputer
df_clean = df.copy()

# Imputation action 1: In the 'Age' column, we can replace all NaNs with
# the median of all the Age values in the column.
imputer = SimpleImputer(strategy='median')
columns_to_impute = ['Age']
imputer.fit(df[columns_to_impute])
df_clean[columns_to_impute] = imputer.transform(df[columns_to_impute])

# Imputation action 2: In the 'Embarked' column, there are only 12 missing
# values. Lets replace them with the value that is most frequent in this column.
imputer = SimpleImputer(strategy='most_frequent')
columns_to_impute = ['Embarked']
imputer.fit(df[columns_to_impute])
df_clean[columns_to_impute] = imputer.transform(df[columns_to_impute])
```

Example: How to apply imputation with scikit-learn