



SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# Machine-learning based research software: Best practices

Dr. Inga Ulusoy, Scientific Software Center  
June 2024

# *What this course is not*

- An introduction to data science
- An introduction to machine learning
- A course about different ML algorithms
- A course about different training approaches and libraries
- ...

# *What this course is*

- A best practices guide to creating machine learning based research software (MLBRS)
- A recommendation on how to prepare your data
- A recommendation on how to train your models
- A guideline on how to generate independently reproducible scientific results using data-based approaches
- A guideline on how to publish your data and your models



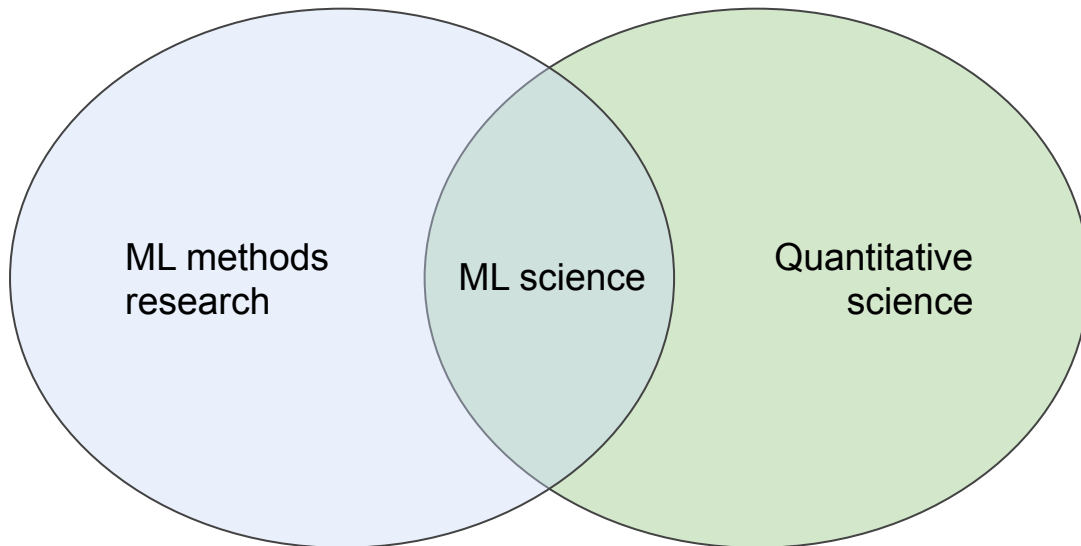
# What is special about research software based on data? (“ML-based science”)

# ML science

- Scientific research that uses machine learning models to extend scientific knowledge
- Answers a scientific question by using ML
- No restriction on algorithm, method, library, domain

*Contrary to:*

- ML methods research: Research on ML methods and algorithms with the goal to improve the field of ML



# Research software

“... software that is developed and used in the context of research...”

- Shifting requirements: A scientific question is answered using computation/simulation, but the way the problem is solved changes as part of the research process
- Passed along researchers
- Often created by researchers that have no fundamental training in software engineering

# MLBRS: Data

- Data is used to train models, to make decisions or predictions
- Different kinds of data: Numerical data, textual data, images, audio, video,...
- Legal considerations: Where does the data come from? Is it licensed? Is it public or private data?
- Ethical considerations: Does the data exploit work of others? Does it break some sort of confidentiality?
- Bias: Is there an inherent bias in the data itself, due to the data collection approach, or other reasons?
- Will the data be publicly available to the community? What license does/will the dataset have?
- What metadata needs to be included on the data card?

# MLBRS: Software

- Does the software incorporate third-party models? If so: Legal considerations (license!)
- Does the software follow software engineering best practices (version control, testing, documentation, ...)?
- Does the software include documentation on how models can be trained, and keeps track of training parameters?
- Does the software help generate model cards?
- What license are models published under?
- Does the software create robust and consistent results, even though it is based on a non-deterministic process?
- Is the software secure against data injection?



# Reproducibility

- Provide data to enable others to reproduce findings
- Provide code to enable others to reproduce findings

→ ***Computational reproducibility (i)***

- Make sure your findings are true findings, and do not arise from problems with your data/code

→ ***Independent reproducibility (ii)***

RSE generally targets (i), but with MLBRS we target (ii)

## ***Why should you care?***

Your research integrity, scientific best conduct (malpractice), can have long-lasting detrimental effect on science!!! (impact on others and the field)

# Key aspects

Legal aspects

Reproducibility  
of the model's  
predictions

Reproducibility  
of the model  
training

Documentati  
on on data  
collection,  
data  
cleaning,  
feature  
selection

Documentation on  
model training,  
hyperparameter  
tuning, model  
testing

Robustness of  
the model(s)

Ethical  
aspects

Model bias

Software  
security

Data bias

Data leakage



# How to prepare your datasets

# Sample dataset

Journal ranking dataset

<https://www.kaggle.com/datasets/xabirhasan/journal-ranking-dataset>

Accept the Github classroom assignment and follow along!

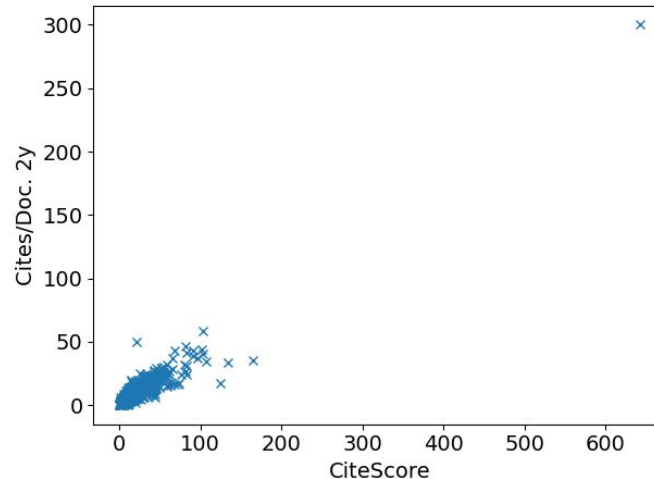
<https://classroom.github.com/a/EUizrQ2H>

# Collecting data

- Data must contain **all ranges of the condition** that is to be sampled
  - For example: To predict the impact of temperature on reactivity, all temperatures that are of interest need to be sampled (predictions only interpolate between data points but cannot extrapolate).
- Data must be **homogeneous throughout feature space**
  - For example: If temperature and pressure are both sampled, all combinations of features must be recorded for a homogeneous distribution of data points.
- Data must be of **good quality**
  - Whether it is real or synthetic data, the model can only make accurate predictions if the data itself is accurate.
- Data **volume** must be sufficient
  - Only with enough data can a model be trained to make accurate predictions.
- Depending on the type of learning, data must be labeled and labeled correctly
  - Incorrect labelling interferes with the learning process.

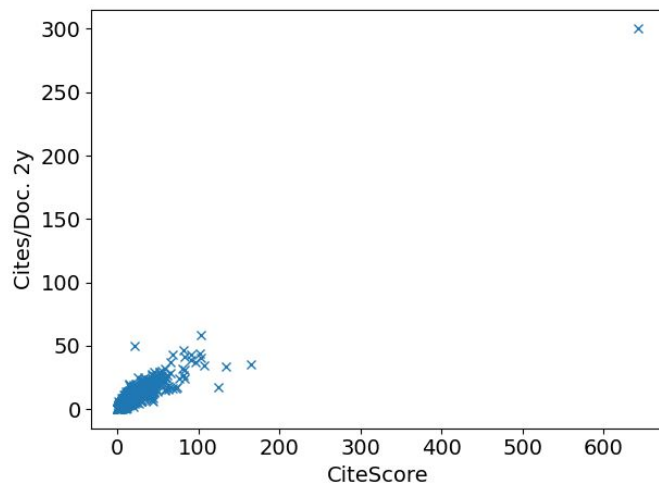
# Collecting data

- Data must contain all ranges of the condition that is to be sampled
  - For example: To predict the impact of temperature on reactivity, all temperatures that are of interest need to be sampled (predictions only interpolate between data points but cannot extrapolate).
  - For example: CiteScore (Scopus citation index) vs. citations over all documents from last 2 years, for scientific journals.



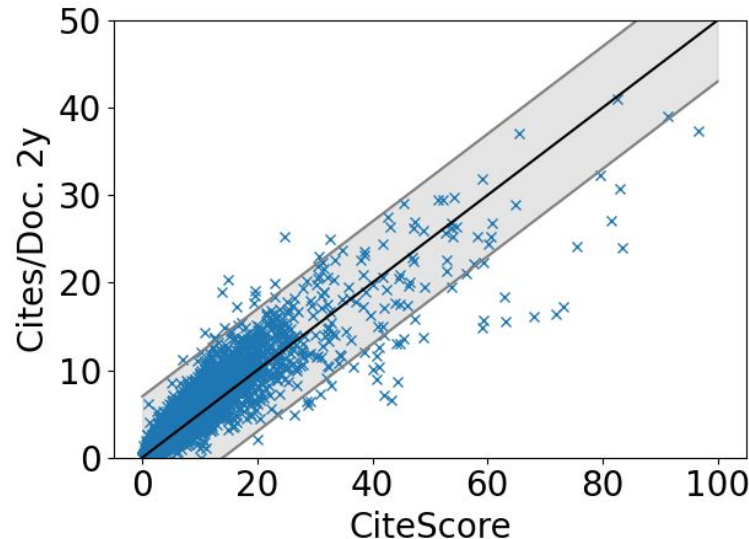
# Collecting data

- Data must be homogeneous throughout feature space
  - For example: If temperature and pressure are both sampled, all combinations of features must be recorded for a homogeneous distribution of data points.
  - For example: CiteScore (Scopus citation index) vs. citations over all documents from last 2 years, for scientific journals.



# Collecting data

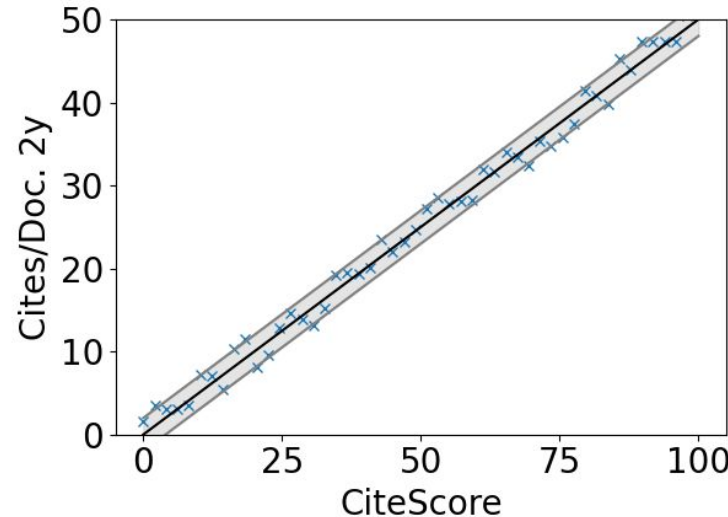
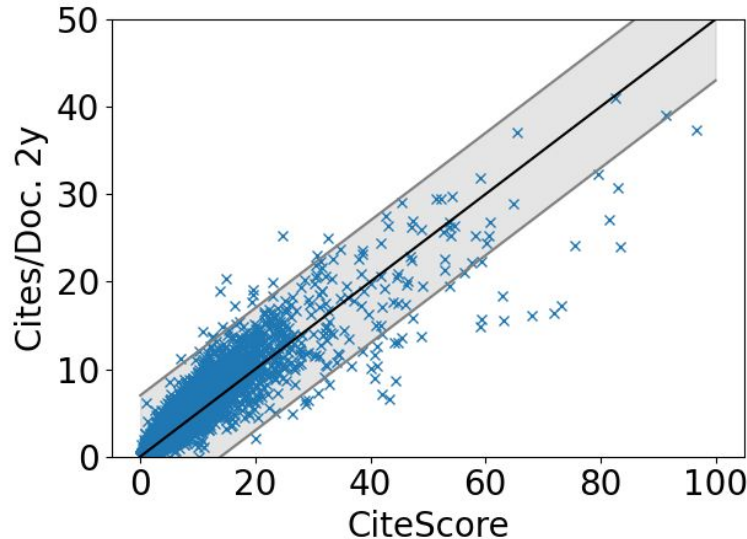
- Data must be of good quality
  - Whether it is real or synthetic data, the model can only make accurate predictions if the data itself is accurate.
  - For example: CiteScore (Scopus citation index) vs. citations over all documents from last 2 years, for scientific journals





# Collecting data

- Data volume must be sufficient
  - Only with enough data can a model be trained to make accurate predictions.
  - For example: Complex data - more data points required; simpler data - fewer data points required



# Collecting data

- Depending on the type of learning, data must be labeled and labeled correctly
  - Incorrect labelling interferes with the learning process.



Photo by nishizuka:  
<https://www.pexels.com/photo/brown-chihuahua-485294/>



Photo by Maksim Goncharenok:  
<https://www.pexels.com/photo/a-chocolate-muffin-on-blue-surface-5994864/>

# Data preparation

- Make sure data is clean.
  - Correct typos, misidentified data types
- Make sure data is homogeneous.
  - Visualize the data and use clustering analysis to identify outliers.
- Remove duplicates.
  - Duplicates introduce bias.
- Feature Engineering: Select influential features, remove unnecessary ones.
  - Unimportant features increase the complexity and reduce robustness.

# Data preparation

- Make sure data is clean.
  - Correct typos, misidentified data types

*Chihuahuah* → *Chihuahua*

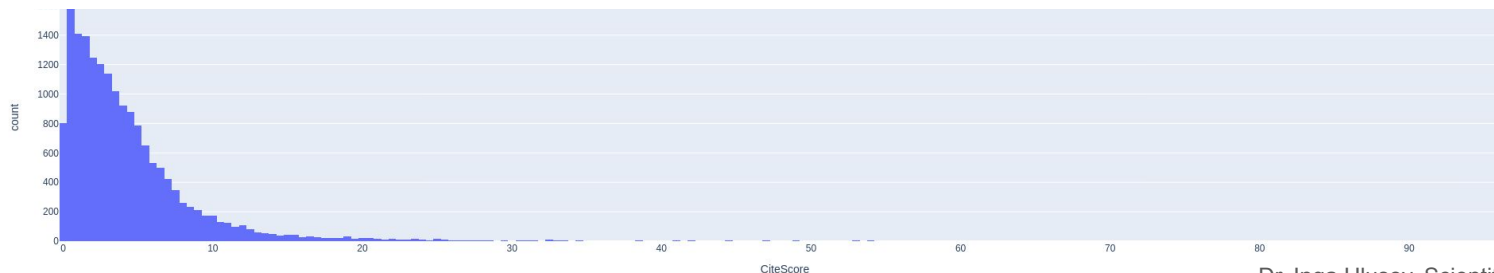
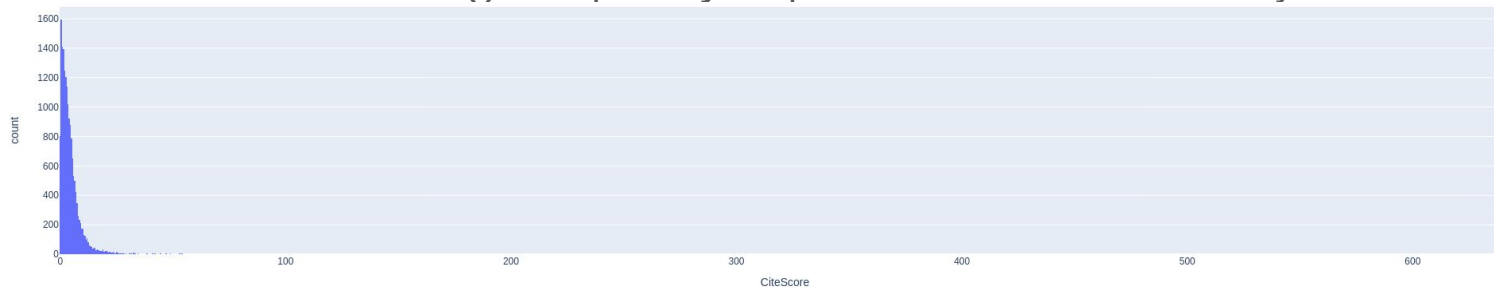


Photo by nishizuka:  
<https://www.pexels.com/photo/brown-chihuahua-485294/>

“26-04-24” → 2024-04-26

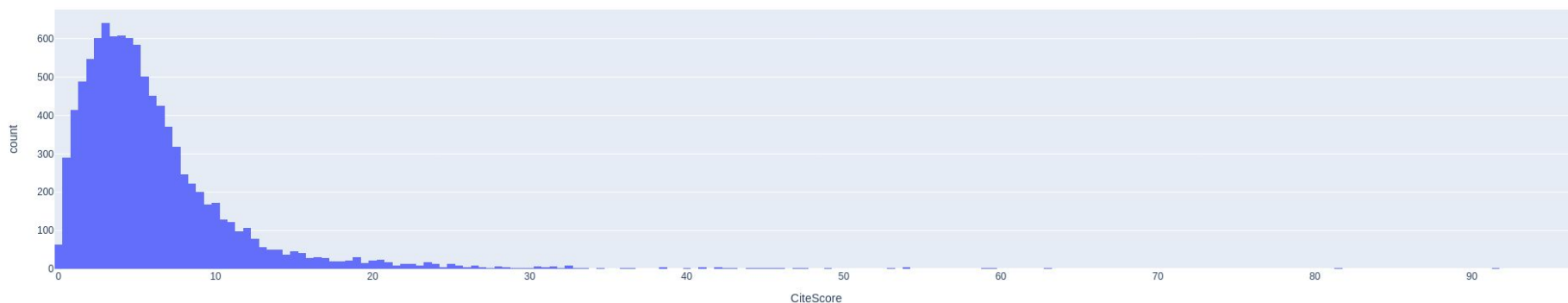
# Data preparation

- Make sure data is homogeneous.
  - Visualize the data and use clustering analysis to identify outliers.
  - Use `df.describe()` and `plotly.express` to better understand your data



# Data preparation

- Remove duplicates.
  - Duplicates introduce bias.
  - Use `df.drop_duplicates()`



# Data preparation

- Feature Engineering: Select influential features, remove unnecessary ones.
  - Unimportant features increase the complexity and reduce robustness.
  - For example: only choose features that are clearly correlated

	Rank	OA	SJR-index	CiteScore	H-index	Best Subject Rank	Total Docs.	Total Docs. 3y	Total Refs.	Total Cites 3y	Citable Docs. 3y	Cites/Doc. 2y
Rank	1.000000	0.111300	-0.503617	-0.485568	-0.625403	0.558208	-0.192069	-0.196795	-0.196338	-0.243070	-0.185484	-0.560625
OA	0.111300	1.000000	-0.069304	-0.056997	-0.178146	0.114037	0.061870	0.046403	0.058683	0.024084	0.048485	-0.045120
SJR-index	-0.503617	-0.069304	1.000000	0.878000	0.565015	-0.281225	0.091092	0.102424	0.094227	0.270083	0.081086	0.828618
CiteScore	-0.485568	-0.056997	0.878000	1.000000	0.527957	-0.279983	0.112000	0.127705	0.122350	0.285965	0.110357	0.943584
H-index	-0.625403	-0.178146	0.565015	0.527957	1.000000	-0.362788	0.331053	0.393130	0.313698	0.505095	0.362266	0.512423
Best Subject Rank	0.558208	0.114037	-0.281225	-0.279983	-0.362788	1.000000	-0.114754	-0.117089	-0.132615	-0.150247	-0.118463	-0.334142
Total Docs.	-0.192069	0.061870	0.091092	0.112000	0.331053	-0.114754	1.000000	0.934468	0.968011	0.806830	0.932626	0.150987
Total Docs. 3y	-0.196795	0.046403	0.102424	0.127705	0.393130	-0.117089	0.934468	1.000000	0.887417	0.854647	0.995085	0.148272
Total Refs.	-0.196338	0.058683	0.094227	0.122350	0.313698	-0.132615	0.968011	0.887417	1.000000	0.802696	0.893789	0.173401
Total Cites 3y	-0.243070	0.024084	0.270083	0.285965	0.505095	-0.150247	0.806830	0.854647	0.802696	1.000000	0.844114	0.308644
Citable Docs. 3y	-0.185484	0.048485	0.081086	0.110357	0.362266	-0.118463	0.932626	0.995085	0.893789	0.844114	1.000000	0.139525
Cites/Doc. 2y	-0.560625	-0.045120	0.828618	0.943584	0.512423	-0.334142	0.150987	0.148272	0.173401	0.308644	0.139525	1.000000
Refs./Doc.	-0.390894	-0.064572	0.267383	0.306913	0.247259	-0.299281	0.032381	0.019822	0.109949	0.076826	0.030626	0.382891
Life Sciences	-0.166150	0.073645	0.071380	0.125088	0.210414	-0.183625	0.044939	0.050866	0.068018	0.051387	0.051948	0.114416

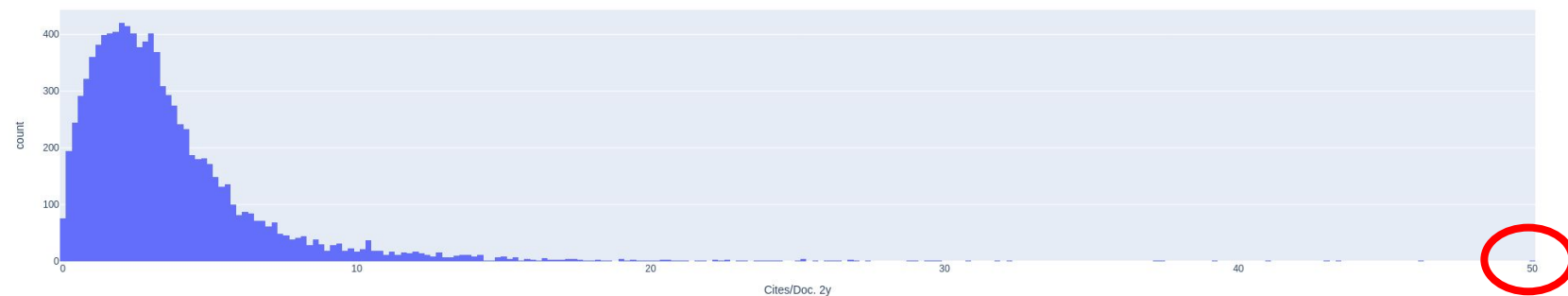
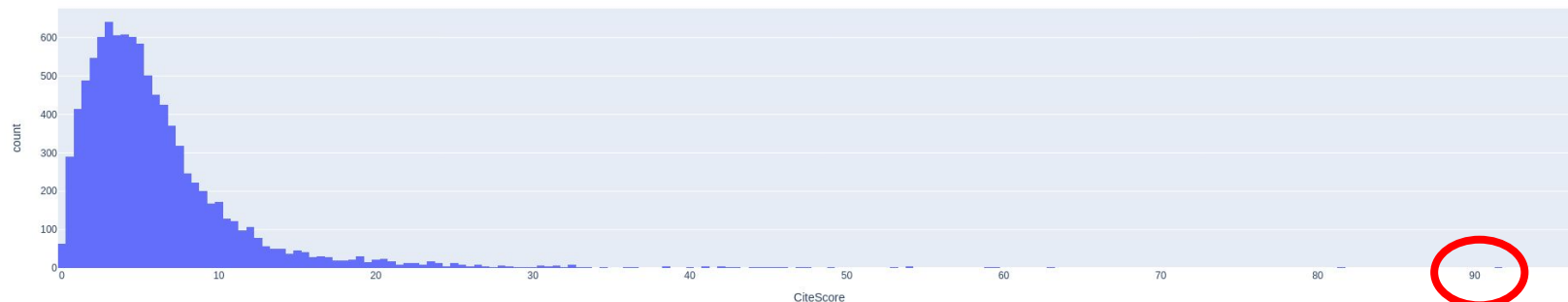
# Data preparation

- Feature Engineering: Normalize features.
  - Features should have similar data ranges for the weights to be in similar ranges, and improved model robustness and faster training.
- Make sure to randomize your data.
  - Otherwise, your train and test data could contain more/less data of a certain kind (inhomogeneous data)
- Feature engineering: Make sure your dataset is balanced.
  - For classification tasks, all classes should have comparable sizes (similar numbers of examples).
- Feature engineering: Pick the right scale.
  - Visualize your data to see if you need to transform ie. onto a log scale.



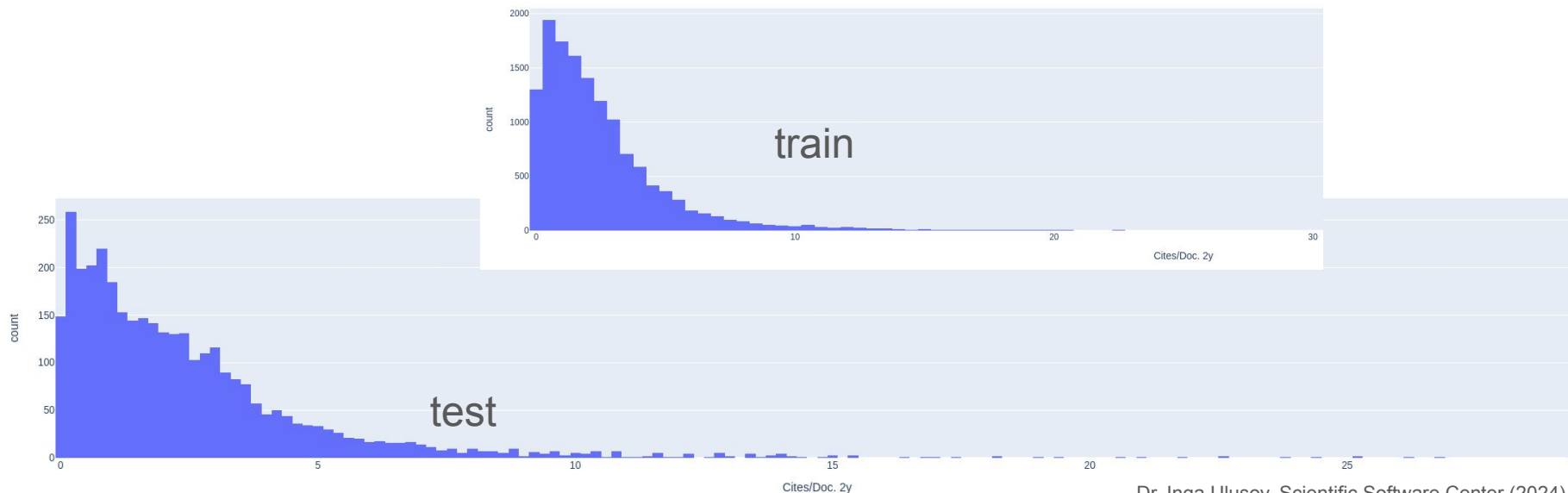
# Data preparation

- Feature Engineering: Normalize features.
  - Features should have similar data ranges for the weights to be in similar ranges, and improved model robustness and faster training.



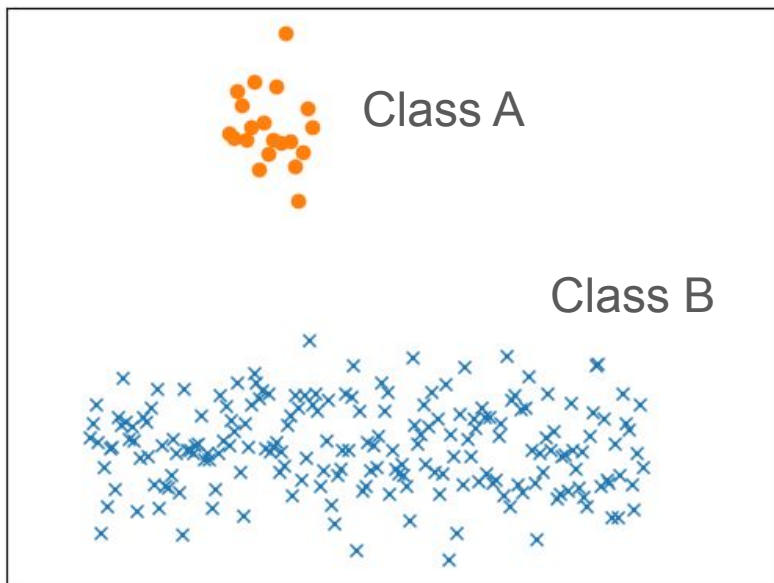
# Data preparation

- Make sure to randomize your data.
  - Otherwise, your train and test data could contain more/less data of a certain kind (inhomogeneous data)



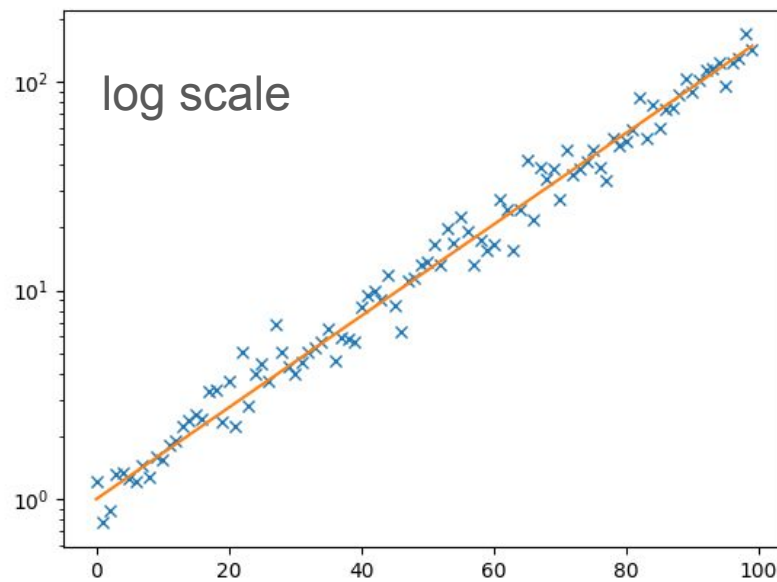
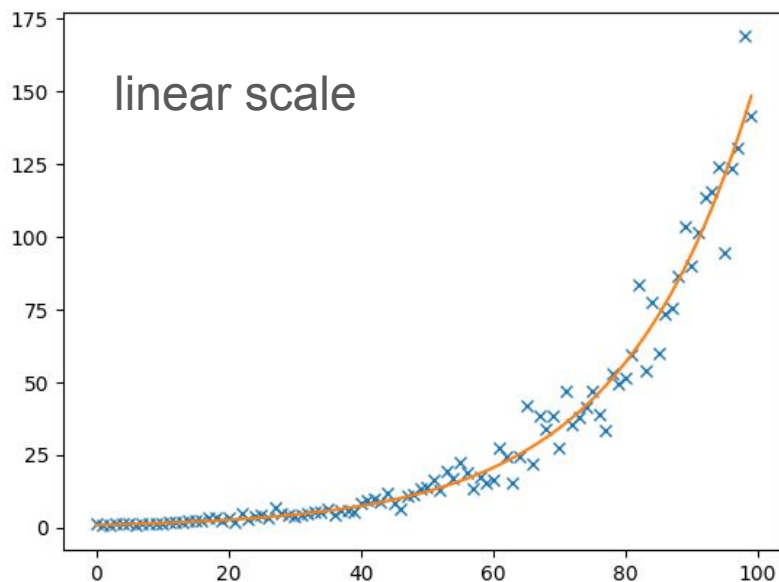
# Data preparation

- Feature engineering: Make sure your dataset is balanced.
  - For classification tasks, all classes should have comparable sizes (similar numbers of examples).



# Data preparation

- Feature engineering: Pick the right scale.
  - Visualize your data to see if you need to transform ie. onto a log scale.





SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# Where to share your datasets

# Why share your data?

- Create a larger impact of your scientific work
- Increase transparency and trust in your work
- Enable others to reproduce and build upon your findings
- Contribute to science as a whole
- BUT: Be careful with legal (copyright) and ethical issues! First clarify with your advisor/institution.
- Follow the FAIR principle: Findable, Accessible, Interoperable, Reusable

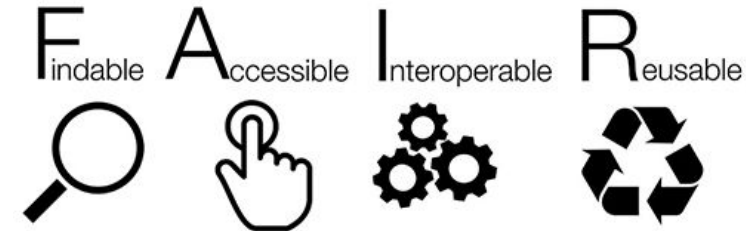


Image credit: SangyaPundir shared under CC-BY-SA 4.0  
The FAIR Guiding Principles for scientific data management and stewardship  
<https://doi.org/10.1038/sdata.2016.18>

# How to share your data

- Get a digital object identifier (DOI) that makes your data citable and identifies your work permanently
- DOI also links your data to metadata, providing all relevant information



# Where to share your data

- HeiDATA
- Open Science Framework
- Zenodo
- Figshare
- Dryad
- Hugging face
- Kaggle: no DOI!
- Domain-specific repositories
  - i.e. <https://lindat.mff.cuni.cz/repository/xmlui/>
  - see <https://www.nature.com/sdata/policies/repositories> for a list
- For a comparison, see DOI: 10.5281/zenodo.3946720





SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# How to train your models

# Choose ML model

- supervised/unsupervised learning (data is/is not labeled?)
- regression/classification (predict numerical values or discrete classes?)
  - Sometimes, a task can be modeled both by a regression or classification model (for example, acidity of an acid could either be predicted as a numerical value, or as a category “very acidic”, “acidic”, “mildly acidic”, etc.)
- Choose algorithm
  - Polynomial / linear / logistic regression, ANN, CNN, K-means,.. - depends on number of features and correlations between the features
- Requires practice (intuition)
  - Simple is better than complex
  - Interpretability is better than obscurity

# Learning

- Cost function  $J$ 
  - Performance of the model as a whole
  - For example, mean squared error (MSE) for linear regression
  - For example, log loss for logistic regression
- Error of the model
  - Compare the mean (median if there are outliers) between models (indicates bias)
  - Standard deviation (spread of values around mean)
  - Range of prediction (minimum and maximum predicted value)
  - $R^2$  (how well model replicates outcomes based on variation of outcomes)
  - Mean absolute error (MAE) gives error in units
  - Relative absolute error (RAE) gives error between two models
  - Root mean squared error (RMSE) gives error in units
  - Mean absolute percentage error (MAPE) a percent estimation of the error

# Hyperparameter tuning

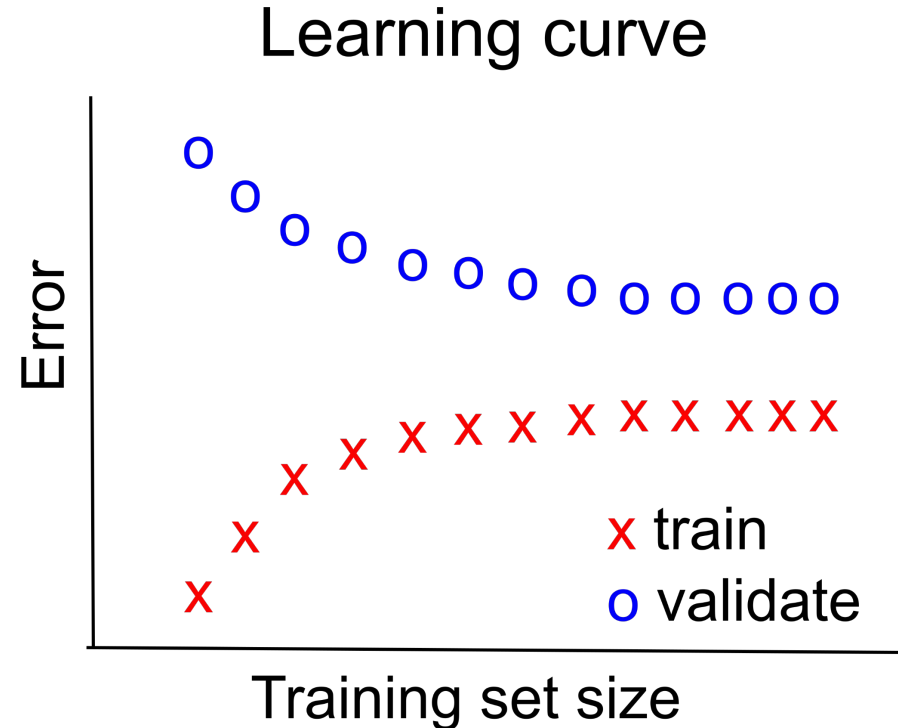
- Define how the model learns from the data
- Choose activation function and architecture (number of hidden layers/neurons)
- Split your data into train and test set (ie., 70:30)
- Make sure your model never sees your test data until you are ready to test!
- Cross validation: Split your train set further into train and evaluate (ie., 80:20), for example, for five different folds
  - Test and validation set must be different! Otherwise hyperparameters will be tuned to reflect test set.

# Evaluate the model

- Define required accuracy
  - MSE: hard to interpret
  - RMSE error has a unit - a bit more interpretable
  - Mean average percentage error:  $\pm$  % deviation on y
- Define expected accuracy - **KEY STEP**
  - Lower bound of the possible accuracy due to measurement uncertainties/intrinsic variation of the data in itself
  - Level of confidence of the data
- If not achieved: what needs to be changed?
  - Not enough data -> Collect more data
  - Not the right features -> Prepare data in a better way (feature engineering)
  - not the appropriate method -> change the algorithm
  - Too simple model -> increase complexity
  - Too complex model -> decrease complexity

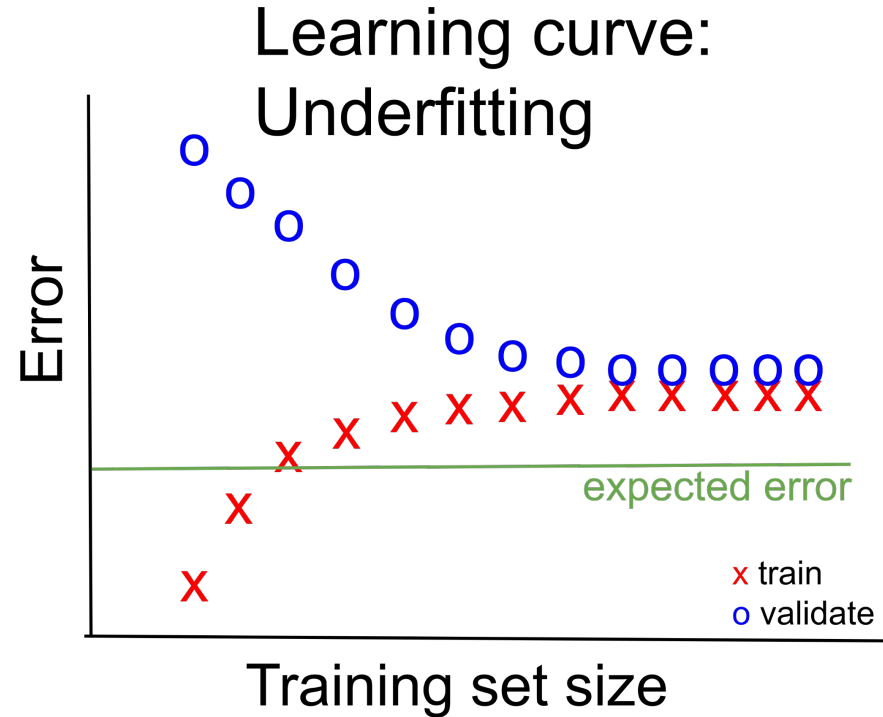
# Learning curve

- Train on more and more data (5%, 10%, 15%, ...) with the selected hyperparameters/features
- Keep the validation set constant
- Measure the error on the train and validation set and plot over size of the training set



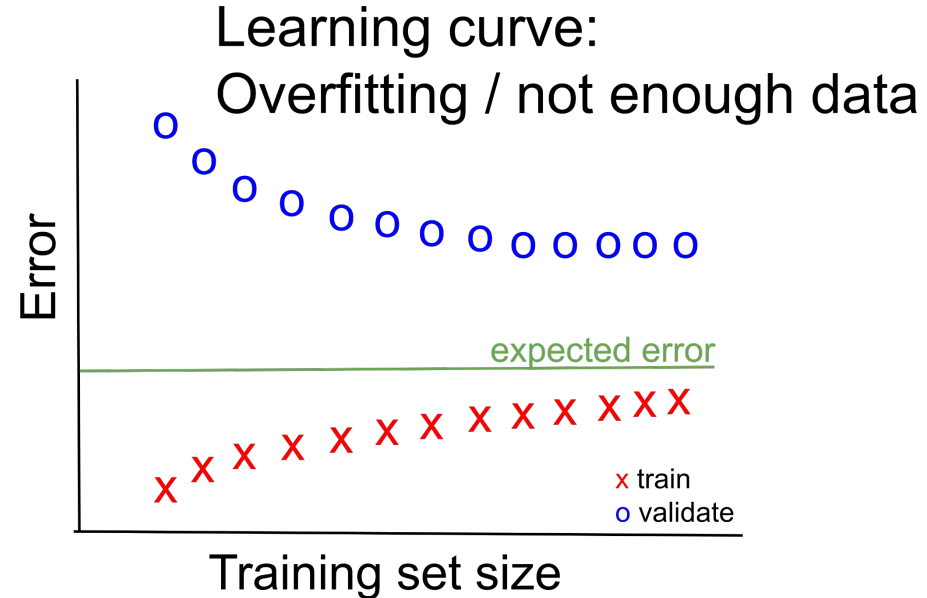
# Learning curve: Underfitting

- Error is too large for both train and validation
- Error is similar for train and validation
- Model learns fast and error converges quickly (stops learning)
- Solution: increase complexity of the model or include additional features



# Learning curve: Overfitting / not enough data

- Error smaller than expected error (train)
- Error larger than expected error (validate)
- Error from train and validate is different
- Model is slow to learn, error does not converge
- Solution: Regularization, remove features, simpler architecture, add more data (complex data requires complex model requires a lot of data - extrapolate learning curve)

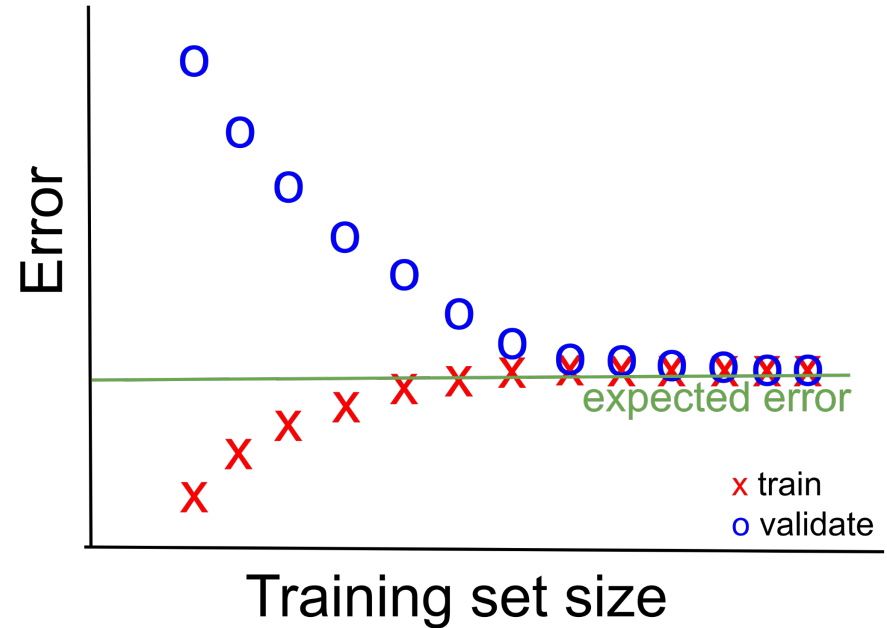




# Learning curve: A good model

- Error from train and validate is identical and close to expected error
- Error converges for both train and validate

## A good learning curve



# Test out learning on the example dataset

Try out different scikit-learn algorithms on the example dataset in the assignment repo.

# Use the model to make predictions

- Use the model on actual data to make predictions
- Evaluate model performance
  - i.e. `sklearn.metrics`

## Regression

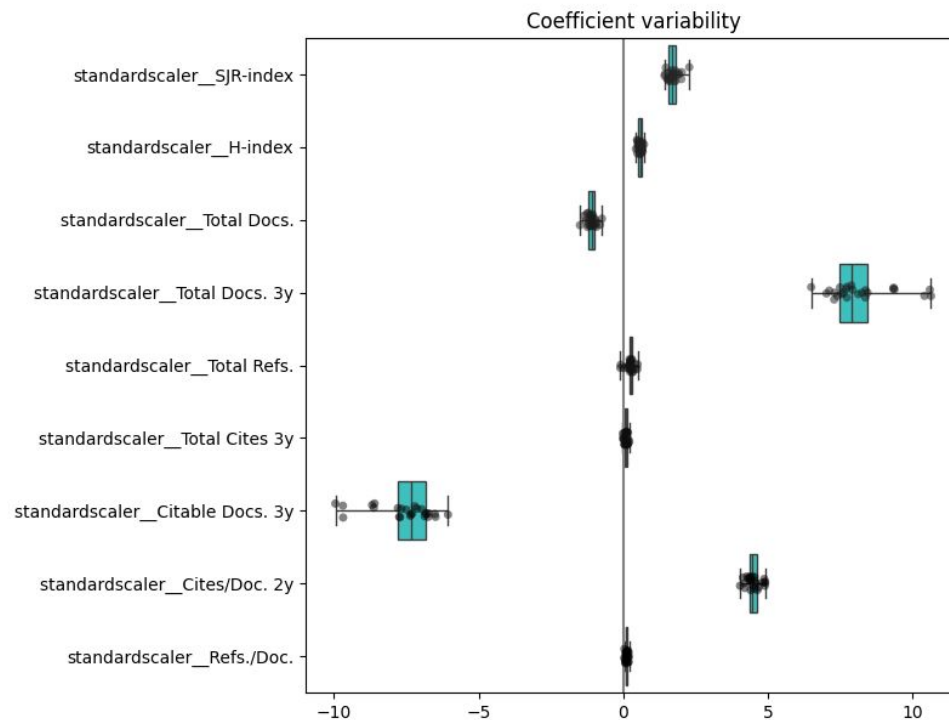
'explained_variance'	<code>metrics.explained_variance_score</code>
'max_error'	<code>metrics.max_error</code>
'neg_mean_absolute_error'	<code>metrics.mean_absolute_error</code>
'neg_mean_squared_error'	<code>metrics.mean_squared_error</code>
'neg_root_mean_squared_error'	<code>metrics.root_mean_squared_error</code>
'neg_mean_squared_log_error'	<code>metrics.mean_squared_log_error</code>
'neg_root_mean_squared_log_error'	<code>metrics.root_mean_squared_log_error</code>
'neg_median_absolute_error'	<code>metrics.median_absolute_error</code>
'r2'	<code>metrics.r2_score</code>
'neg_mean_poisson_deviance'	<code>metrics.mean_poisson_deviance</code>
'neg_mean_gamma_deviance'	<code>metrics.mean_gamma_deviance</code>
'neg_mean_absolute_percentage_error'	<code>metrics.mean_absolute_percentage_error</code>
'd2_absolute_error_score'	<code>metrics.d2_absolute_error_score</code>
'd2_pinball_score'	<code>metrics.d2_pinball_score</code>
'd2_tweedie_score'	<code>metrics.d2_tweedie_score</code>

Scores with scikit-learn,

[https://scikit-learn.org/stable/modules/model\\_evaluation.html](https://scikit-learn.org/stable/modules/model_evaluation.html)

# models

- Keep config of training parameters: model card
- Test for robustness
  - i.e. dummy estimators
  - i.e. sklearn.inspection
  - Generally how well the model deals with noise, out-of-distribution samples, adversarial examples



# Model card

- Model details
  - Architecture, parameters, citation information, license information
- Intended use
  - Use cases within the model's scope
- Performance metrics
  - Intended performance on given data
- Training data
  - Description of training data and data distribution
- Quantitative analysis
  - Potential biases and limitations
- Ethical consideration
  - Privacy and fairness concerns, impact on society
- [https://huggingface.co/spaces/huggingface/Model\\_Cards\\_Writing\\_Tool](https://huggingface.co/spaces/huggingface/Model_Cards_Writing_Tool)
- <https://github.com/openai/gpt-3/blob/master/model-card.md>

# Model cards

Create a model card for your trained model.

# Where to share/publish/deploy your models

# Model sharing platforms

You can make models available for others on model sharing platforms like

- Hugging face,
- Kaggle.

***Advantages:*** Public platform with version control and model cards, you can link the data into the repo, allows others to use your model for production or fine-tuning.



# How to test your software that is based on ML models

# Testing of non-deterministic processes

**Try to make processes deterministic**

For example: Use specified random seed.

**Separate deterministic and non-deterministic processes and test separately**

For example: Input processing can be tested separately from model prediction.

**Test for output parameters and properties that remain constant**

For example: Number of predictions, feature length, etc.

**Include multiple valid outputs in your tests**

For example: Three most probable classifications.

**Robustness: A robust model is more likely to behave like a deterministic system**

Make sure your model output is stable under a range of conditions.

**Accuracy: The model accuracy will affect the testing strategy**

A higher accuracy leads to more consistent predictions.

**Distribution: You can also test for the distribution of results rather than the results themselves**



SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# Publishing your results as a scientific work

# REFORMS checklist

To ensure reproducibility, include a checklist such as the REFORMS checklist into the publication of your results.

There are a couple of domain-specific checklists, especially in health and life sciences (STARD, CLAIM, see the EQUATOR<sup>\*</sup> network), that may be more appropriate - REFORMS aims to be most general.

REFORMS checklist: DOI 10.48550/arXiv.2308.07832, <https://reforms.cs.princeton.edu/>

<sup>\*</sup>EQUATOR Network, <https://www.equator-network.org/reporting-guidelines/>

# REFORMS: Example

## 1. Study goals

### 1a) Population or distribution about which the scientific claim is made

*The group to which the claim will be generalized*

**Here:** Data is from 2023, and will be generalized to the consecutive year 2024.

# REFORMS: Example

## 1. Study goals

### 1b) Motivation for choosing this population or distribution

*Choice of a particular population of interest - pure scientific interest, need for applied knowledge, ...*

**Here:** Availability of the data, inductive approach: the questions that the dataset can address.

# REFORMS: Example

## 1. Study goals

### 1c) Motivation for the use of ML methods in the study

*Why focus is on building a model that reliably maps input data to output data, instead of using traditional statistical methods?*

**Here:** The relationship between features, which features to pick, and desired output is unknown.

# REFORMS: Example

## 2. Computational reproducibility

### 2a) Dataset

*Cite datasets with permanent links to clarify which version; if contains sensitive data: release synthetic datasets (synthpop library)*

**Here:** Dataset is publicly available on kaggle, but has no DOI. It would need to be published under a persistent identifier.



# REFORMS: Example

## 2. Computational reproducibility

### 2b) Code

*Cite exact version of code (DOI, version number, commit tag)*

**Here:** Code would need to be supplied in repository such as GitHub including a version number/tag, or published on zenodo with a DOI.

# REFORMS: Example

## 2. Computational reproducibility

### 2c) Computing environment

*Details about the hardware (CPU, RAM, disk space), software (operating system, programming language, version number for each package used), and computing resources (time taken to generate the results)*

**Here:** Include the requirements in the repository, or even provide a container simulating the exact environment that was used - or some other runtime environment specification

# REFORMS: Example

## 2. Computational reproducibility

### 2d) Documentation

*Document installation, requirements, running the code, usage examples, expected results*

**Here:** Provide the details in a README of the code repository, or build a documentation using tools (like sphinx or Doxygen). Have someone else test the documentation for clarity and correctness.

# REFORMS: Example

## 2. Computational reproducibility

### 2e) Reproduction script

*Reproduction scripts that prepare the environment, install the code, download datasets, and run code to reproduce the results in the paper*

**Here:** Provide a cleaned-up notebook in the code repository, that follows through all the steps as reported; or use snakemake/bash/..

# REFORMS: Example

## 3. Data quality

### 3a) Data source(s)

*When, where, how data were collected, how ground-truth annotations were performed*

**Here:** Kaggle, reference the data collection script on GitHub  
<https://github.com/abir0/SJR-Journal-Ranking>

# REFORMS: Example

## 3. Data quality

### 3b) Sampling frame

*List of people or units from which a sample is drawn*

**Here:** Data was collected on a certain date; one could in principle recollect the data at different dates and reevaluate the outcome.

# REFORMS: Example

## 3. Data quality

### 3c) Justification for why the dataset is useful for the modeling task

*Why the dataset is a good representative to model the question/answer*

**Here:** Dataset contains all Scopus- and Web-of-Science-indexed journals, and their corresponding characteristics. We are predicting a Scopus citation index, thus the dataset is complete with respect to this feature.

# REFORMS: Example

## 3. Data quality

### 3d) Outcome variable

*How outcome variable is defined - this is usually not a perfect match for what it should represent*

**Here:** In any given year, the CiteScore of a journal is the number of citations in Scopus, received in that year and in previous three years, for documents published in the journal during the total period (four years), divided by the total number of published documents (articles, reviews, conference papers, book chapters, and data papers) in the journal during the same four-year period (Wikipedia). The associated construct is “journal impact” and may include a conflict of interest since it is Elsevier’s in-house metric.



# REFORMS: Example

## 3. Data quality

### 3e) Number of samples in the dataset

*Total sample size, number of samples in each class, number of individual data in the dataset*

**Here:** The total sample size is 18013 samples, without duplicates this reduces to 8587 samples.

# REFORMS: Example

## 3. Data quality

### 3f) Missingness

*Missing data*

**Here:** There is no missing data; however, it would be good to check the predictions for multiple datasets taken at different points in time.

# REFORMS: Example

## 3. Data quality

### 3g) Dataset for evaluation is representative

*Justify why data is representative for target selected in 1a)*

**Here:** This is the citation data for the year 2023 and thus closest in time to 2024. It contains all journals indexed in Scopus. The population of samples in the dataset is expected to follow the same trends as in the target.

# REFORMS: Example

## 4. Data preprocessing

### 4a) Excluded data and rationale

*Justify why particular subset of data was chosen*

**Here:** All data points except the samples with a CiteScore above 100 were included. According to the distribution of the data, those were determined outliers and would receive an inappropriate weight in the learning, skewing the learning process and predictions.

# REFORMS: Example

## 4. Data preprocessing

### 4b) How impossible or corrupt samples are dealt with

*How erroneous or impossible data points are identified and amended*

**Here:** Filter, remove, or transform such data - does not apply here.

# REFORMS: Example

## 4. Data preprocessing

### 4c) Data transformations

*Normalizing, augmenting, imputing missing data, oversampling - the latter two must be done separately on each fold of the dataset!*

**Here:** Data could be but does not have to be normalized numerically for each feature. Other options could be transformation to a logarithmic scale.

# REFORMS: Example

## 5. Modeling

### 5a) Model description

*Input, output, type of model, loss function, algorithm*

**Here:** Input is either one feature (Cites/Doc. 2y) or multiple features (SJR-index, H-index, Total Docs., Total Docs. 3y, Total Refs., Total Cites 3y, Citable Docs. 3y, Cites/Doc. 2y, Refs./Doc.) (*explain!!*) and the output is the CiteScore. The type of model is a linear regression model, using least squares loss function and L2 regularization in the case of multiple features.

# REFORMS: Example

## 5. Modeling

### 5b) Justification for the choice of model types implemented

*Model needs to be interpretable, types of models considered*

**Here:** We only considered regression models since the data is mostly continuous and seems to be linked in a linear fashion. This is a simple type of model that allows to interpret the results more easily.



# REFORMS: Example

## 5. Modeling

### 5c) Model evaluation method

*Model needs to be evaluated on test data different from training data*

**Here:** Models are evaluated using direct evaluation metrics after a random split into train and test data; and for multiple features, k-fold cross validation on the training set was carried out. Metrics then calculated as an average over the folds and on the test set. The sample size of training and test set was 80:20, and five folds were used with each 80:20 split of training and validation.

# REFORMS: Example

## 5. Modeling

### 5d) Model selection method

*How final model(s) and hyperparameters were selected*

**Here:** Models were selected with default parameters/after 5-fold cross validation averaging over the folds.

# REFORMS: Example

## 5. Modeling

### 5e) Hyperparameter selection

*How hyperparameters were optimized*

**Here:** Models were selected with default parameters/after 5-fold cross validation averaging over the folds.

# REFORMS: Example

## 5. Modeling

### 5f) Appropriate baselines

*How baseline models were trained and optimized*

**Here:** No baseline models were used in this approach. We have not estimated the accuracy of the test data (limit to prediction).

# REFORMS: Example

## 6. Data leakage

### 6a) Train-test separation is maintained

*Use a hold-out test set that is also not used in synthetic data generation*

**Here:** Test set was generated before renormalization and after removal of duplicates and outliers.

# REFORMS: Example

## 6. Data leakage

### 6b) Dependencies or duplicates between datasets

*Could be more than one sample from same origin (patient); or time series is split randomly*

**Here:** No duplicate journal entries are present in the data.

# REFORMS: Example

## 6. Data leakage

### 6c) Feature legitimacy

*Any feature in the training that somewhat identifies predicted variable*

**Here:** All features are legitimate and constitute valid input.

# REFORMS: Example

## 6. Data leakage

### 6c) Feature legitimacy

*Any feature in the training that somewhat identifies predicted variable*

**Here:** All features are legitimate and constitute valid input.



# REFORMS: Example

## 7. Metrics and uncertainty quantification

### 7a) Performance metrics used

*Proper choice of metric depending on application*

**Here:** RMSE, MSE, MAE

See Leist et al., Sci. Adv. 8, eabk1942 (2022) Table 4 for appropriate metrics depending on the data and algorithm used

# REFORMS: Example

## 7. Metrics and uncertainty quantification

### 7b) Uncertainty estimates

*Randomness in training or evaluation data, or training process*

**Here:** For example generate confidence intervals by bootstrapping.

# REFORMS: Example

## 7. Metrics and uncertainty quantification

### 7c) Appropriate statistical tests

*Statistical testing of ML models*

**Here:** Select an appropriate test for your model and data.

For an overview, see Sebastian Raschka, Model evaluation, model selection, and algorithm selection in machine learning, arXiv:1811.12808

# REFORMS: Example

## 8. Generalizability and limitations

### 8a) Evidence of external validity

*Report how claim is generalizable to target population.*

**Here:** This will apply in our case since the predicted variable directly depends on the input features in a simple mathematical formula.

# REFORMS: Example

## 8. Generalizability and limitations

### 8b) Contexts in which the study's findings will not hold

*Clear expectations and unjustified hype.*

**Here:** The approach reported here will not work when the CiteScore calculation is changed, or prior to 2020 where a different approach had been used.



SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# MLBRS security

# Security

- Threat modelling: who, what, how
- Data-oriented attack: Access training data, poison data, inject trojan data
- Model-oriented attack: Modify training process (pre-trained malicious models), manipulate the deployed model (model patches, privacy information leakage, model inversion attacks)
- System-oriented attack: specialised hardware accelerators for ML software (SOC, trojan in GPU/TPU, for model corruption, backdoor insertion, model extraction, spoofing, information extraction, sybil attack)

# Security: best practices

Phases	Vulnerabilities Causes
Analysis phase	No risk analysis/ No security policy
	Biased risk analysis
	Unanticipated risks
Design phase	Relying on non-secure abstractions
	Security/Convenience tradeoff
	No logging
	Design does not capture all risks
Implementation phase	Insufficiently defensive input checking
	Non-atomic check and use
	Access validation errors
	Incorrect crypto primitive implementation
	Insecure handling of exceptional conditions
	Bugs in security logic
Deployment phase	Reuse in more hostile environments
	Complex or unnecessary configuration
	Insecure defaults
Maintenance phase	Feature interaction
	Insecure fallback

Chen, Barbar, Security for Machine Learning-based Software Systems, DOI 10.1145/3638531





SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# Infamous AI mistakes

# AI in general

- **Automatization at amazon:** Experimental hiring tool, developed by a team of five, used artificial intelligence to give job candidates scores ranging from one to five stars
- Tool was trained on all resumes of the last 10 years
- Tool preferably suggested male candidates, penalizing resumes that contained the word “women’s” or graduates from all-female colleges
- *Why?*

# Automated resume selection at amazon

- The dataset consisted of the resumes of the last 10 years: Predominantly male applicants
- Women are underrepresented in tech:

<https://fingfx.thomsonreuters.com/gfx/rngs/AMAZON.COM-JOBS-AUTOMATION/010080Q91F6/index.html>

- Thus, the model learnt it was more often correct if it suggested male candidates: Unbalanced representation of the dataset

# AI in general

- U.S. healthcare system uses commercial algorithms to guide health decisions
- Algorithm to target patients for “high-risk care management” programs
- Identify patients who benefit the most: <https://www.science.org/doi/10.1126/science.aax2342>
- Model is trained on healthcare spendings to determine the healthcare need
- Algorithm was much more likely to recommend white patients for these programs than black patients, even though the black patients were evidently sicker
- ***Why?***

# Bias in healthcare need estimation


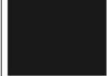
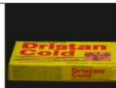




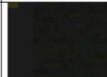

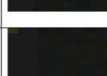

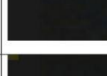
- Training based on healthcare spendings: But people of color are more likely to have lower incomes - making them less likely to access medical care even if they are insured
- Also, they may experience higher barriers to accessing health care (geography, transportation, work/childcare constraints), in addition to direct doctor-patient bias
- Data shows that race is correlated with substantial differences in health-care spendings: This results in a bias of the trained model











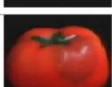

# AI mistakes in research software

- Applying machine learning methods to COVID-19 radiological imaging for improving the accuracy of diagnosis
- Distinguish patients with COVID-19 from patients without COVID-19 but also bacterial pneumonia
- Tools were trained on public datasets with CT and CXR images
- Predictive tools failed practical tests: <https://www.nature.com/articles/s42256-021-00307-0>
- ***Why?***

# Learning from the image background

- Image dataset was collected under controlled conditions:  
Does not represent the target distribution of interest
- CNN learnt to distinguish the image background rather than the image content
- Actually quite prevalent problem in computer vision

Subject No	Subject Image (128*128 pixels)	Rendered Image (22*29 pixels)
1		
		
		
2		
		
		

3		
		
		
4		
		
		

# AI mistakes in research software

- Predict whether a country is likely to slide into civil war based on GDP, poverty rates, type of government structure, etc.
- Complex models using Random Forests and Adaboost outperform more standard statistical approaches like logistic regression by far
- Missing values in the dataset were constructed using imputation on the complete dataset
- Models proved to be over-optimistic and erroneous <https://doi.org/10.1016/j.patter.2023.100804>
- ***Why?***



# Civil war predictions: Data leakage

- Data leakage: The data was imputed for missing values using the whole dataset
- Thus, the training dataset contained information about the test dataset
- This leads to an inflated estimate of the model performance

<https://doi.org/10.1016/j.patter.2023.100804> supplemental material

# AI mistakes in research software

- Twitter moods predict the closing value of the stock market
- Analyze tweets using OpinionFinder (sentiment analysis)  
<https://www.sciencedirect.com/science/article/abs/pii/S187775031100007X>
- Daily tweets are categorized, resulting in a time series of the public mood
- Trained algorithm is overly optimistic and erroneous
- ***Why?***

# Predict stock market from public mood

- Training data contained points from later times than the test data
- Model was effectively allowed to see the future - temporal data leakage



# Classification of failures/errors

# Data leakage

Spurious relationship between independent variables and target variable

Artifact of collection, sampling, pre-processing

Leads to inflated estimates of model performance

## **Lack of clean separation training/test**

- no test set
- pre-processing on training and test set (over/under sampling, imputation)
- feature selection on entire dataset
- duplicates in dataset

# Data leakage

## **Model uses features that are not legitimate**

- for example, use of a certain drug when predicting illness (hypertensive drug, antibiotics)

## **Test set is not drawn from distribution of scientific interest**

- temporal leakage (test set must not contain data from before the training set)
- non-independence between training and test samples (same people/units in both sets - use block crossvalidation)
- sampling bias in test distribution (spatial bias, age, image settings)



SCIENTIFIC  
SOFTWARE  
CENTER



UNIVERSITÄT  
HEIDELBERG  
ZUKUNFT  
SEIT 1386

# Summary

# Summary

## Data:

- Publish your dataset using a persistent identifier
- Ensure your training and test set does not suffer from leakage
- Make sure your data is appropriate to answer your scientific question

## Models:

- Publish your models using version control (and possibly persistent identifier)
- Prefer simple, interpretable models
- Test for overfitting
- Use appropriate performance measures for the selected type of algorithm



# Summary

## **Publications:**

- Ensure your dataset is not subject to legal/ethical issues - check for license compatibility
- Publish a reproduction script for computational reproducibility alongside your data/models
- Publish a checklist like REFORMS alongside your publication for credibility, safeguarding against malpractice, and independent reproducibility

# Resources

- Good practices in machine learning (Mathieu Bauchy)  
(<https://www.youtube.com/watch?v=WScUQnU-ozQ&t=3213s>)
- Kaggle <https://www.kaggle.com/learn>
- Hugging Face: <https://huggingface.co/learn>
- REFORMS checklist <https://reforms.cs.princeton.edu/>
- More resources <https://www.cs.princeton.edu/~arvindn/>,  
<https://www.aisnakeoil.com/p/introducing-the-ai-snake-oil-book>
- Scikit-learn resources [https://scikit-learn.org/stable/common\\_pitfalls.html](https://scikit-learn.org/stable/common_pitfalls.html)
- Testing of non-deterministic software  
[https://bssw.io/blog\\_posts/testing-non-deterministic-research-software](https://bssw.io/blog_posts/testing-non-deterministic-research-software)