The AI Lifecycle

Week 4 - Data collection and preparation

Spring 2025

Hafsteinn Einarsson

HÁSKÓLI ÍSLANDS

VERKFRÆÐI- OG NÁTTÚRUVÍSINDASVIÐ

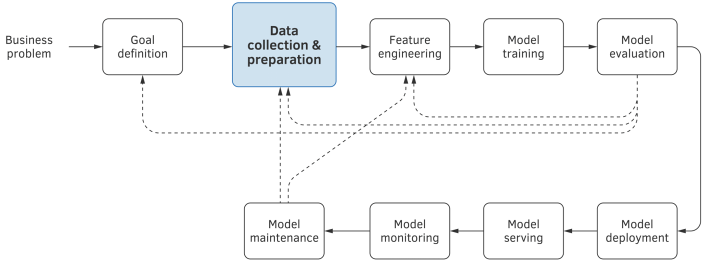
Review

Last week

- Before the project starts (chapter 2)
- Data collection and preparation (chapter 3)

This week

- Data collection and preparation (chapter 3)
- Feature engineering (chapter 4)
- New exercise on data preparation and feature engineering


Business problem
Goal definition
Data collection & preparation
Feature engineering
Model training
Model evaluation
Model deployment
Model serving
Model monitoring
Model maintenance

Overview

What is Interaction Data?

Definition

Interaction data is collected from user interactions with a system your model supports, providing insights into user behavior and system performance.

Key components of high-quality interaction data:

- Context of the interaction
- User actions during the interaction
- Outcomes of the interaction

You can be considered lucky if you get good interaction data from the system your model supports.

Interaction Data in Search Engines

Understanding User Interactions

Interaction data captures user behavior and preferences to personalize search results and improve the search experience.

Reranking Model Overview

- Personalized reranking of search results based on user preferences.
- Input: Original list of search results.
- Output: Reranked list tailored to the user.

Context, Action, and Outcome

- **Context:** User's search query and initial search results.
- **Action:** User clicks on a document link or "next page."
- **Outcome:** Time spent on page and whether the user returns to search results.

Interpreting User Actions

- A click followed by significant time on page indicates a good ranking.
- A quick return to search results suggests a less relevant link.
- Clicking "next page" may imply that initial results were unsatisfactory.

Using Interaction Data for Improvement

Enhancing Personalization

Analyzing interaction data helps refine the ranking algorithm for better personalization and user satisfaction.

The Cost of Ignoring Interaction Data: Analytics Debt

What is Analytics Debt?

Analytics debt arises when a company underestimates the value of interaction data, leading to missed opportunities and uninformed decisions.

Consequences of Analytics Debt

Example

A company that has a digital product, such as a mobile app, may neglect to collect and analyze data on user interactions with the app. This could include data on user engagement, app crashes, feature usage, and user feedback.

As a result, the company may not have a clear understanding of how users interact with the app or which features are most popular. This can lead to difficulties in making informed decisions about app updates, marketing strategies, and overall product development.

Data-Driven Success: The Booking.com Approach

Example

Booking.com, an online travel agency, has effectively managed their analytics debt through a focus on collecting and analyzing vital customer data. This allows them to better understand the features that drive engagement and inform their decision-making.

The company has a strong culture of experimentation, with teams running hundreds of A/B tests each year to optimize their website and mobile app. This data-driven approach has led to a better user experience and increased revenue for the company.

Other notable examples: Netflix, Amazon, Spotify, Google, Airbnb, Uber, Zara, LinkedIn.

Overview

What is Data Leakage?

Data leakage can occur when information in the training data is used that would not be expected to be available in the production environment, leading to overly optimistic performance estimates.

Why is it a problem?

- It gives a false sense of accuracy.
- It can lead to poor model performance in production.
- It undermines trust in model predictions.

Identifying Data Leakage

Common Causes of Data Leakage:

1. Inclusion of target information in the features.
2. Incorrect data preprocessing steps, i.e., some features hide the target.
3. Use of future data that wouldn't be available at prediction time.

What causes data leakage?

The target is a function of a feature.

Example

Consider a model that aims to predict the price of a house based on various features such as square footage, number of bedrooms, and neighborhood.

If the training data includes both the square footage of the house and the price per square foot, the model will have access to information about the target variable (house price) that would not be available at prediction time, leading to data leakage.

What causes data leakage?

Target is hidden in one of the features.

Example

You're predicting the gender of customers. One feature column is categorical with values that contain gender information directly, such as

`M18-25`, `F25-35`,..., `F65+`.

If the information is available at training time because it is the output of another model, you still might want to use it.

What causes data leakage?

The feature is from the future.

Example

You're predicting whether a customer will pay back a loan. In your training data, you have a column called `LatePayments` which is 1 if the customer did not pay on time and 0 otherwise.

The main problem is that this information is not available to you when the customer applies for a loan [a].

[a] In fact, the most important variable on whether someone pays back debt is whether they have a history of paying up debt in the past.

Preventing Data Leakage

Strategies to Prevent Data Leakage

- Carefully design feature selection and data preprocessing.
- Simulate real-world prediction scenarios during validation.
- Regularly consult with domain experts to understand data context.

Preventing Data Leakage

Strategies to Prevent Data Leakage

- Carefully design feature selection and data preprocessing.
- Simulate real-world prediction scenarios during validation.
- Regularly consult with domain experts to understand data context.

Sanity Check:

- Remove suspect features and evaluate model performance.
- If performance drops significantly, investigate further.
- Alternatively, study performance of single features.

Discussion and Review

Questions to Consider:

- Have you encountered data leakage in your projects and/or job?
- What steps did you take to identify and address it?
- Can you think of other scenarios where data leakage might occur?

Overview

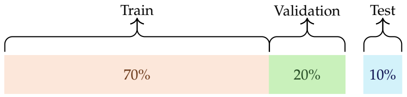
Introduction to Data Partitioning

Why Partition Data?

Data partitioning is the process of dividing a dataset into separate sets to train, validate, and test a machine learning model. It is crucial for:

- Assessing model performance accurately.
- Preventing overfitting.
- Ensuring the model generalizes well to unseen data.

Conditions for Effective Data Partitioning



- Apply the split to raw data, before any preprocessing or cleaning is performed.
- Preserve group and temporal relationships within each subset.
- Shuffle the data randomly before partitioning to ensure similar distributions.
- Avoid leakage during the partitioning process to prevent bias.

Data splitting example

Example

In training a gesture detector, keep data on each individual within a single set to prevent **group leakage**—where the model learns specific user patterns rather than general gestures.

Shuffling Data Before Partitioning

Why Shuffle?

- Ensures that the resulting sets are representative of the entire dataset.
- Helps maintain similar class distributions across training, validation, and test sets.
- Provides a stronger guarantee that hyperparameter choices are suitable for production.

Consequences of not shuffling data before partitioning

Example

A dataset sorted by customer value class could lead to training on only "low-value" customers and testing on "high-value" customers, resulting in poor model performance.

Best Practice: Always shuffle data randomly before partitioning to avoid biased or unrepresentative sets.

Summary: Best Practices in Data Partitioning

- Partition raw data to maintain integrity.
- Preserve group and temporal relationships.
- Shuffle data to ensure representative subsets.
- Regularly check for and prevent data leakage.

Overview

Introduction to Missing Data

Why is Missing Data a Problem?

Missing data can lead to biased estimates, reduced statistical power, and can ultimately affect the generalizability and validity of a machine learning model.

Strategies for Handling Missing Data

- Removing rows with missing features
- Using algorithms that handle missing values
- Data imputation

Data Imputation Techniques

Replacing missing values with:

- The mean or median of the available values in that feature column, denoted by $\hat{x}^{(j)} \leftarrow \frac{1}{N^{(j)}} \sum_{i \in \mathcal{S}^{(j)}} x_i^{(j)}$.
- A value outside the normal range, such as the minimum or maximum value.
- If the feature is categorical, replacing it with a new category (e.g. "unknown") or a neutral category (e.g. "middle of the range").
- Training a regression model to predict the missing value based on the other available features.
- Adding a new binary feature column, which is 1 if the feature is missing.

Data imputation

It is important to keep in mind that:

- The same imputation method should be used in the production environment.
- Imputation should only be performed on the training set to prevent data leakage.

Overview

What is Data Augmentation?

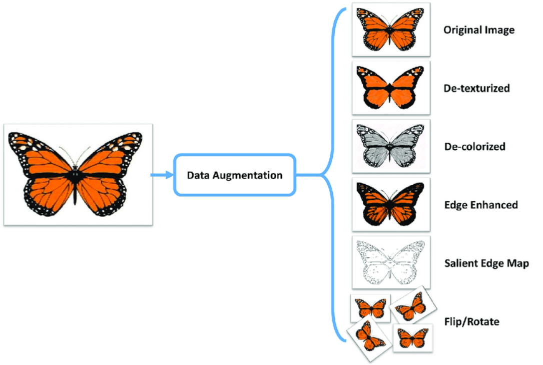
Definition

Data augmentation involves creating additional training data from existing examples by applying random but realistic transformations to generate new and diverse samples without extra labelling work.

The goal is to generate new and diverse training examples that can help improve the model's robustness and generalization performance.

Data Augmentation for Images

- Rotation, Translation, Scaling
- Flipping, Cropping, Color Jittering



Libraries for Image Data Augmentation

- Albumentations. This library offers a wide range of augmentation techniques, including Channel Shuffle, Solarize, Invert Image, and many others. It allows for the creation of complex augmentation pipelines using the Compose class, which can include various transformations like rotation, transposition, blur, and distortion.
- SOLT. Streaming over Lightweight Transformations, SOLT is a Python library designed specifically for data augmentation in deep learning. It supports images, segmentation masks, labels, and key points, and is known for its speed due to its OpenCV backend. SOLT provides a Stream instance for creating an augmentation pipeline with various transformations like rotation, flipping, shearing, scaling, and more.

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).
- Replace words with close neighbors in the embedding space.

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).
- Replace words with close neighbors in the embedding space.
- Remove the word and guess the correct replacement with BERT.

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).
- Replace words with close neighbors in the embedding space.
- Remove the word and guess the correct replacement with BERT.
- Use KNN to quickly find the right label (good in document classification).

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).
- Replace words with close neighbors in the embedding space.
- Remove the word and guess the correct replacement with BERT.
- Use KNN to quickly find the right label (good in document classification).
- Translate to another language and back.

Data augmentation for text

With LLMs, data augmentation has become less of an issue. However, we have to be careful about preserving the contextual and grammatical structure.

- Replace words with synonyms.
- Replace words with a more general word (hypernym).
- Word vectors can be perturbed with noise (new hyperparameter).
- Replace words with close neighbors in the embedding space.
- Remove the word and guess the correct replacement with BERT.
- Use KNN to quickly find the right label (good in document classification).
- Translate to another language and back.
- Use a generative model to paraphrase your text.

Discussion: Challenges in Data Augmentation

Questions to Consider:

- What are some challenges you might face when augmenting data?
- How can you ensure that augmented data still represents the problem space accurately?

Challenges

- **Semantic Integrity:** For text data, ensuring that augmentations do not change the meaning of the text is crucial. For images, transformations must not make the subject unrecognizable or alter key features.

Challenges

- **Semantic Integrity:** For text data, ensuring that augmentations do not change the meaning of the text is crucial. For images, transformations must not make the subject unrecognizable or alter key features.
- **Contextual Relevance:** Augmentations should be relevant to the context of the data. For example, flipping an image of a car might be acceptable, but flipping text that has a directional context (like "left" and "right") would not be.

Challenges

- **Semantic Integrity:** For text data, ensuring that augmentations do not change the meaning of the text is crucial. For images, transformations must not make the subject unrecognizable or alter key features.
- **Contextual Relevance:** Augmentations should be relevant to the context of the data. For example, flipping an image of a car might be acceptable, but flipping text that has a directional context (like "left" and "right") would not be.
- **Overfitting:** If the augmentations are not diverse enough or too many augmented samples are similar, the model might overfit to the augmented data rather than learning generalizable patterns.

Challenges

- **Semantic Integrity:** For text data, ensuring that augmentations do not change the meaning of the text is crucial. For images, transformations must not make the subject unrecognizable or alter key features.
- **Contextual Relevance:** Augmentations should be relevant to the context of the data. For example, flipping an image of a car might be acceptable, but flipping text that has a directional context (like "left" and "right") would not be.
- **Overfitting:** If the augmentations are not diverse enough or too many augmented samples are similar, the model might overfit to the augmented data rather than learning generalizable patterns.
- **Computational Resources:** Data augmentation can significantly increase the size of the dataset, requiring more computational resources for training.

Challenges

- **Semantic Integrity:** For text data, ensuring that augmentations do not change the meaning of the text is crucial. For images, transformations must not make the subject unrecognizable or alter key features.
- **Contextual Relevance:** Augmentations should be relevant to the context of the data. For example, flipping an image of a car might be acceptable, but flipping text that has a directional context (like "left" and "right") would not be.
- **Overfitting:** If the augmentations are not diverse enough or too many augmented samples are similar, the model might overfit to the augmented data rather than learning generalizable patterns.
- **Computational Resources:** Data augmentation can significantly increase the size of the dataset, requiring more computational resources for training.
- **Balance and Diversity:** Ensuring that the augmented data maintains the class balance and diversity of the original dataset can be challenging, especially in cases of imbalanced datasets.

Ensuring Accurate Representation in Data Augmentation

- **Domain Knowledge:** Incorporating domain expertise can guide the selection of appropriate augmentation techniques that maintain the integrity of the data.

Ensuring Accurate Representation in Data Augmentation

- **Domain Knowledge:** Incorporating domain expertise can guide the selection of appropriate augmentation techniques that maintain the integrity of the data.
- **Validation:** Use a separate validation set that has not been augmented to ensure that the model performs well on real, unmodified data.

Ensuring Accurate Representation in Data Augmentation

- **Domain Knowledge:** Incorporating domain expertise can guide the selection of appropriate augmentation techniques that maintain the integrity of the data.
- **Validation:** Use a separate validation set that has not been augmented to ensure that the model performs well on real, unmodified data.
- **Monitoring Distributions:** Analyze the distributions of features in the augmented data to ensure they do not deviate significantly from those in the original dataset.

Ensuring Accurate Representation in Data Augmentation

- **Domain Knowledge:** Incorporating domain expertise can guide the selection of appropriate augmentation techniques that maintain the integrity of the data.
- **Validation:** Use a separate validation set that has not been augmented to ensure that the model performs well on real, unmodified data.
- **Monitoring Distributions:** Analyze the distributions of features in the augmented data to ensure they do not deviate significantly from those in the original dataset.
- **Pilot Studies:** Conduct small-scale experiments to determine the effects of different augmentation strategies on model performance.

Overview

Understanding Imbalanced Data

Definition

Imbalanced data refers to a dataset where the distribution of classes is not equal. This often results in a model that is biased towards the majority class.

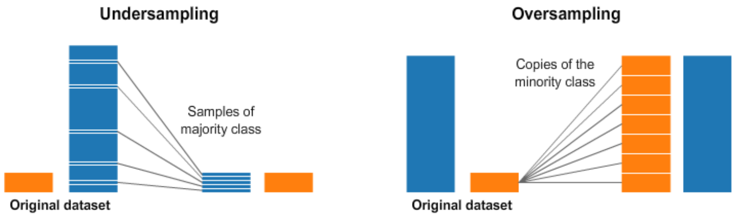
Challenges of Imbalanced Data

Preventing Model Bias

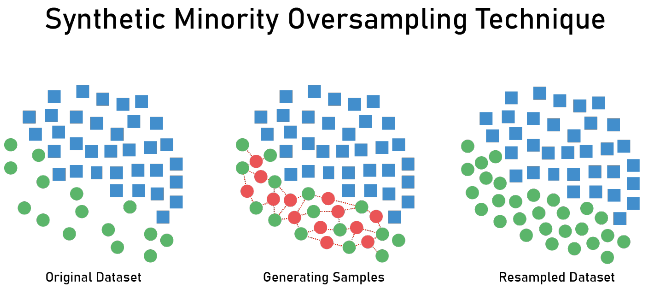
A key challenge with imbalanced data is preventing the model from ignoring the minority class, which can lead to poor predictive performance on this class.

Resampling Techniques: Overview

- **Oversampling**: Increase the presence of the minority class in the dataset.
- **Undersampling**: Reduce the presence of the majority class in the dataset.

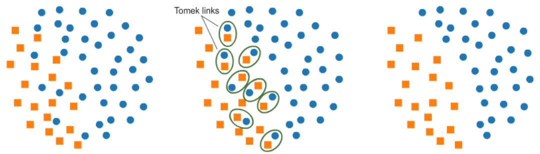


Oversampling Techniques



- Simple duplication of minority class samples.
- Synthetic Minority Oversampling Technique (SMOTE).
- Adaptive Synthetic Sampling Method (ADASYN).

Undersampling Techniques



- Random removal of majority class samples.
- Tomek links: Remove majority class points that are nearest neighbors to minority class points.
- Clustering-based undersampling: Replace majority class samples with cluster centroids.

Problems with Resampling

- **Overfitting**: Oversampling can make the model too specific to the replicated samples.
- **Loss of Information**: Undersampling can discard potentially important data from the majority class.

Combining Resampling Techniques

- Using a combination of oversampling and undersampling can mitigate the drawbacks of each method.
- Hybrid approaches can balance the dataset while preserving information.

Overview

Random Sampling in Data Preparation

Introduction to Random Sampling

When dealing with large datasets, it may be necessary to work with a subset of the data. Random sampling methods help in selecting a representative sample to ensure model reliability.

Simple Random Sampling

- **Simple Random Sampling:** Each data point has an equal chance of being selected.
  - Pros: Easy to implement and unbiased for balanced datasets.
  - Cons: May not represent minority classes well in imbalanced datasets.

Stratified Random Sampling

- **Stratified Sampling:** The dataset is divided into homogeneous subgroups (strata) and samples are drawn from each stratum proportionally.
    - Pros: Ensures representation of all groups, reducing sampling bias.
    - Cons: Requires knowledge of important stratifying variables.

Cluster-Based Sampling

- **Cluster-Based Sampling:** If stratification is not feasible, clustering can be used to create groups within the data.
  - Clusters are formed based on data similarity, and samples are drawn from these clusters.
  - Helps in maintaining diversity in the sample when stratification criteria are unclear.

Overview

Data Storage Levels

In data management, it's important to consider the different levels at which data can be stored. I assume familiarity with most of the concepts in 3.11.1 (data formats) and 3.11.2 (data storage levels). Take note of

- **Object storage** refers to a method of storing unstructured data in a flat, hierarchical namespace using a simple API. A common example is Amazon S3.
- A **datalake** is a centralized repository that allows storing raw and unstructured data in its native format. It is often used for big data and analytics workloads, and it follows a "schema on read" approach, meaning that the structure of the data is determined only when it is read and not when it is written.

Data versioning

Why do you need data versioning?

- **Better organization:** Effective management and tracking of different versions of data is crucial, especially if data is scattered across different sources.

Data versioning

Why do you need data versioning?

- **Better organization:** Effective management and tracking of different versions of data is crucial, especially if data is scattered across different sources.
- **Flexible update process:** When releasing models based on new features frequently, it is important to have the capability to update models while still providing access to earlier versions of models. This is necessary for users who may require an earlier version of the model for their specific use-case, but still want it to be trained on recent data.

Data versioning

Why do you need data versioning?

- **Better organization:** Effective management and tracking of different versions of data is crucial, especially if data is scattered across different sources.
- **Flexible update process:** When releasing models based on new features frequently, it is important to have the capability to update models while still providing access to earlier versions of models. This is necessary for users who may require an earlier version of the model for their specific use-case, but still want it to be trained on recent data.
- **Annotator analysis:** Keeping track of labeling efforts, such as identifying and excluding certain annotators, is important for maintaining the quality of the data.

Data versioning

Why do you need data versioning?

- **Better organization:** Effective management and tracking of different versions of data is crucial, especially if data is scattered across different sources.
- **Flexible update process:** When releasing models based on new features frequently, it is important to have the capability to update models while still providing access to earlier versions of models. This is necessary for users who may require an earlier version of the model for their specific use-case, but still want it to be trained on recent data.
- **Annotator analysis:** Keeping track of labeling efforts, such as identifying and excluding certain annotators, is important for maintaining the quality of the data.
- **Reproducibility:** In case of unexpected performance degradation in production, the ability to revert to an earlier version of the model and data is crucial for troubleshooting and resolving the issue.

Levels of Data Versioning

Unversioned
Data is not tracked or managed in any way, making it difficult to revert to previous versions or reproduce past results.

Disadvantages of unversioned data

- **Inability to make versioned deployments**. As code and data must be versioned together for effective deployment.
- **Lack of ability to revert to previous performance** in case of problems with the model.
- **No ability to analyze the impact of changes** on the model performance.

Levels of Data Versioning

Snapshot at Training Time

A snapshot of the data and model is taken at the time of training.

- Allows you to reproduce past results
- However, may not allow for training earlier models on the most recent data
- Can be useful for investigating why a new model performs worse than a previous version

Levels of Data Versioning

Data and Code Versioned as One
Both data and code are versioned together, allowing for reproducibility and the ability to train earlier models on more recent data. This is sufficient for most projects.

- **Small data assets** such as dictionaries, and small datasets are stored jointly with the code in a version control system like Git or Mercurial.
- **Large files** are stored in object storage like S3 or GCS with unique IDs.
- The training data is stored in a standard format such as JSON or XML with relevant metadata.
- Tools like Git Large File Storage (LFS) can automatically replace large files with text pointers inside Git while storing the file contents on a remote server.
- The version of the dataset is defined by the git signatures of the code and the data file, and can also include a timestamp for easy identification.

Levels of Data Versioning

Specialized Data Versioning Solutions
Custom software solutions such as Data Version Control and Pachyderm are necessary for certain use cases, especially for compliance reasons.

- These tools typically interoperate with code versioning software, such as Git.
- Level 2 is typically sufficient for most projects, but consider Level 3 if additional tools are needed.
- Keep in mind that implementing Level 3 adds complexity to the already complex project of data versioning.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.
- Information on how the data was split (e.g. training, validation, test sets).

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.
- Information on how the data was split (e.g. training, validation, test sets).
- Details on any pre-processing steps that were applied to the data.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.
- Information on how the data was split (e.g. training, validation, test sets).
- Details on any pre-processing steps that were applied to the data.
- Information on any excluded data.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.
- Information on how the data was split (e.g. training, validation, test sets).
- Details on any pre-processing steps that were applied to the data.
- Information on any excluded data.
- The format in which the data is stored.

Documentation and Metadata

Proper documentation and metadata are essential to ensure the longevity and reproducibility of your project. When wrapping up a project, it is important to include the following information in your data documentation:

- A high-level explanation of the data in each column, including information on data types and allowed values.
- Instructions on how the data was collected and annotated.
- Information on how the data was split (e.g. training, validation, test sets).
- Details on any pre-processing steps that were applied to the data.
- Information on any excluded data.
- The format in which the data is stored.
- The total number of examples in the dataset.

It is also important to document any processes and responsibilities related to data erasure, if applicable.

General advice on data preparation

General best practices

- **Automate data transformation**. Manual data transformation is error-prone, time-consuming and makes it difficult to reproduce results. Utilize scripting languages and programming frameworks to automate data preparation tasks.
- **Quality over quantity**. The quality of the data is more important than the choice of algorithm or model. Invest time and effort in data augmentation, cleaning, and validation to ensure the data is accurate, consistent, and relevant to the task at hand.

What's next?

- We start on chapter 4 next week.
- Data presentations tomorrow.