Data Preprocessing

Data Science Project: An Inductive Learning Approach

Prof. Dr. Filipe A. N. Verri

About these slides

These slides are companion material for the book

Data Science Project: An Inductive Learning Approach

Prof. Dr. Filipe A. N. Verri

`https://leanpub.com/dsp`

All intellectual content comes from the book and is not AI-generated. Slides were produced with the assistance of Claude Code.

I find your lack of faith disturbing.

— Darth Vader, Star Wars: Episode IV (1977)

Overview

Contents

- Introduction
- Data cleaning
- Data sampling
- Data transformation

Objectives

- Understand the main data preprocessing tasks and techniques
- Learn the behavior of the preprocessing chain (fitting, adjustment, application)

Introduction

Why preprocess?

- Tidy data is not necessarily suitable for modeling
- Example: perceptron requires **numerical** inputs
- Preprocessing adjusts data for the chosen learning machine
- Operations are **dependent on the learning method**

Three steps of a preprocessing technique

1. **Fitting**: parameters adjusted to the training data
2. **Adjustment**: training data transformed according to fitted parameters (may change sample size/distribution)
3. **Applying**: operation applied to new data, sample by sample

Understanding these steps is crucial to avoid **data leakage**.

Formal definition

Strategy F takes a table $T = (K, H, c)$ and returns:

- Adjusted table $T' = (K', H', c')$
- Fitted preprocessor $f_\phi(z)$

A chain of operations $F_1, \ldots, F_m$:

$$f(z; \phi) = \left(f_{\phi_1} \circ \cdots \circ f_{\phi_m}\right)(z)$$

Each operation depends on the result of the previous ones.

Degeneration

The preprocessor **degenerates** over tuple z if $f_\phi(z) = (?, \ldots, ?)$.

- Unexpected values, incomplete information, ...
- If any step f_{ϕ_i} degenerates, the whole chain degenerates
- Developer must define a **default behavior**:
    - Return a default value
    - Redirect to a different model
    - Raise an error/warning

Preprocessing task categories

1. **Data cleaning** — remove errors and inconsistencies
2. **Data sampling** — select or create variations of the training set
3. **Data transformation** — adjust types and variables for modeling

Presented in typical application order (not fixed).

Data cleaning

Treating inconsistent data

Three common tasks (parameters **not fitted** from data):

- **Unit conversion** — ensure same units across columns
- **Range check** — validate values within expected bounds
- **Category standardization** — unify different representations

Could be done in data handling, but having them in the preprocessor ensures consistent treatment of new data in production.

Unit conversion

	Unit conversion
Goal	Convert physical quantities into the same unit of measurement.
Fitting	None. User declares units and conversion factors.
Adjustment	Sample by sample, independently.
Applying	Converts values and drops the unit column.

Range check

Range check	
Goal	Check whether values are within the expected range.
Fitting	None. User declares the valid range $[a, b]$.
Adjustment	Sample by sample; degenerated samples may be removed.
Applying	If $x \notin [a, b]$: replace with $?$, clamp to $[a, b]$, or degenerate.

Category standardization

Category standardization	
Goal	Map different names to a single canonical form.
Fitting	None. User declares the mapping.
Adjustment	Sample by sample, independently.
Applying	Case standardization, special character removal, dictionary/fuzzy matching.

Outlier detection

- Observations significantly different from the rest
- Caused by errors or mixed phenomena
- Standard approach: remove outliers from the dataset
- Per-variable: replace outlier values with missing data

IQR heuristic: Given Q_1, Q_3, and $\mathrm{IQR} = Q_3 - Q_1$,
a value is an outlier if $x < Q_1 - 1.5\,\mathrm{IQR}$ or $x > Q_3 + 1.5\,\mathrm{IQR}$.

Outlier detection — IQR

Outlier detection using the IQR	
Goal	Detect outliers using the IQR.
Fitting	Store Q_1 and Q_3 for each variable.
Adjustment	Sample by sample, independently.
Applying	Replaces outlier values with missing data.

More advanced: One-Class SVM[1] for generalizable outlier classification.

[1] B. Schölkopf et al. (2001). **"Estimating the support of a high-dimensional distribution".** In: *Neural computation* 13.7, pp. 1443–1471.

Outlier removal

Outlier removal	
Goal	Remove observations that are outliers.
Fitting	Parameters of the outlier classifier.
Adjustment	Sample by sample; degenerated samples removed.
Applying	Degenerates if classified as outlier; pass-through otherwise.

Developer must specify default behavior when an outlier is detected in production.

Treating missing data

Most models cannot handle missing data. Four strategies:

1. Remove **rows** with missing data
2. Remove **columns** with missing data
3. **Impute** the missing values
4. **Indicator variable** + imputation

Removing rows “on demand” can change the data distribution, especially if data is not missing at random.

Row removal (missing data)

Row removal based on missing data	
Goal	Remove observations with missing data in specified variables.
Fitting	None. Variables to check are declared beforehand.
Adjustment	Sample by sample; degenerated samples removed.
Applying	Degenerates over rows with missing data in specified variables.

Column removal (missing data)

Column removal based on missing data	
Goal	Remove variables with missing data.
Fitting	Mark all variables with missing data in the training set.
Adjustment	Marked columns are dropped.
Applying	Drops the same columns chosen during fitting.

Valuable information may be lost when removing columns for all samples.

Imputation

Imputation of missing data	
Goal	Replace missing data with a statistic (mean, median, mode).
Fitting	Statistic computed from available training data.
Adjustment	Sample by sample, independently.
Applying	Replaces missing values; optionally creates an indicator variable.

Indicator variable: useful when missingness itself is informative
(e.g., "days since last pregnancy" is missing if male or zero children).

Data sampling

Data sampling

After cleaning, select or create variations of the training set:

- **Random sampling** — reduce dataset size
- **Scope filtering** — reduce the modeled phenomenon's scope
- **Class balancing** — equalize class representation

Random sampling	
Goal	Select a random subset of the training data.
Fitting	None. User declares the sample size.
Adjustment	Rows randomly chosen.
Applying	**Pass-through**: does nothing with new data.

	Scope filtering
Goal	Remove observations that do not satisfy a predefined rule.
Fitting	None. User declares the rule.
Adjustment	Sample by sample; degenerated samples removed.
Applying	Degenerates over samples that violate the rule.

Variation: **model trees —** shallow decision trees that branch into different models at each leaf.

Class balancing

Class balancing	
Goal	Balance the number of observations in each class.
Fitting	User declares or calculates target class sizes.
Adjustment	Undersample (random removal) or oversample (re-sampling).
Applying	**Pass-through**: does nothing with new data.

Advanced: SMOTE[2] creates synthetic minority samples without repetition.

[2] N. V. Chawla et al. (2002). **"SMOTE: synthetic minority over-sampling technique".** In: *Journal of artificial intelligence research* 16, pp. 321–357.

Data transformation

Data transformation

Data is now clean and well-sampled. Transform columns to suit the model:

- **Type conversion** — categorical $\leftrightarrow$ numerical
- **Normalization** — scale values to expected ranges
- **Dimensionality reduction** — reduce number of variables
- **Data enhancement** — add external information

Label encoding:

- Replace $x \in \{a, b, c\}$ with $x' \in \{1, 2, 3\}$
- Suitable when there is a natural order $a < b < c$

One-hot encoding:

- Create a new column for each category
- Column $= 1$ if present, 0 otherwise
- Group rare categories into an *other* column

One-hot encoding

One-hot encoding	
Goal	Create a new column for each category value.
Fitting	Store the unique values; optionally mark an *other* category.
Adjustment	Sample by sample, independently.
Applying	New columns filled with 1 or 0; unknown values assigned to *other*.

Numerical to categorical (binning)

Binning numerical values	
Goal	Create a categorical column from a numerical one.
Fitting	Store the range of each bin (by frequency or by range).
Adjustment	Sample by sample, independently.
Applying	Assigns each value to the corresponding bin.

Also common: converting dates/intervals to numerical differences (e.g., birth date $\rightarrow$ age).

$$x' = \frac{x - \mu}{\sigma}$$

Standardization	
Goal	Scale values in a column (zero mean, unit variance).
Fitting	Store μ and σ from the training set.
Adjustment	Sample by sample, independently.
Applying	Scales values using the fitted μ and σ.

Rescaling

$$x' = a + (b - a)\,\frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

Rescaling	
Goal	Rescale values to a target range $[a, b]$.
Fitting	Store x_{min} and x_{max} from the training set.
Adjustment	Sample by sample, independently.
Applying	Rescales and clamps: $\max(a, \min(b, x'))$.

Dimensionality reduction

Feature selection:

- Select a subset of existing variables
- Example: rank by mutual information with target, keep top k

Feature extraction:

- Create new variables as combinations of original ones
- Linear: PCA
- Non-linear: autoencoders
- Drawback: new variables are hard to interpret

Data enhancement

Data enhancement	
Goal	Enrich the dataset with external information.
Fitting	Store the external dataset and the join column.
Adjustment	Left join with external dataset (same number of rows).
Applying	Enhances each new observation with external information.

Example: join zip codes with socioeconomic data.

Comments on unstructured data

- Any unstructured data can be transformed into structured data
- Bag of words, word embeddings, signal/image processing
- Modern methods (CNNs) learn preprocessing and model jointly
  - Convolutional layers = learned feature extraction
- Unstructured data is a vast field, out of scope of this book

Takeaways

- Each learning method requires specific preprocessing tasks
- Fitting the preprocessor is crucial to avoid leakage
- Default behavior when the chain degenerates must be specified
- Three categories: cleaning, sampling, transformation
- Preprocessing parameters are fitted, not fixed

Questions?