Lecture:
Data Science: Pre-processing, Explorative Data Analysis

Data Science

Elf Data Science Courses, version 1.2.0

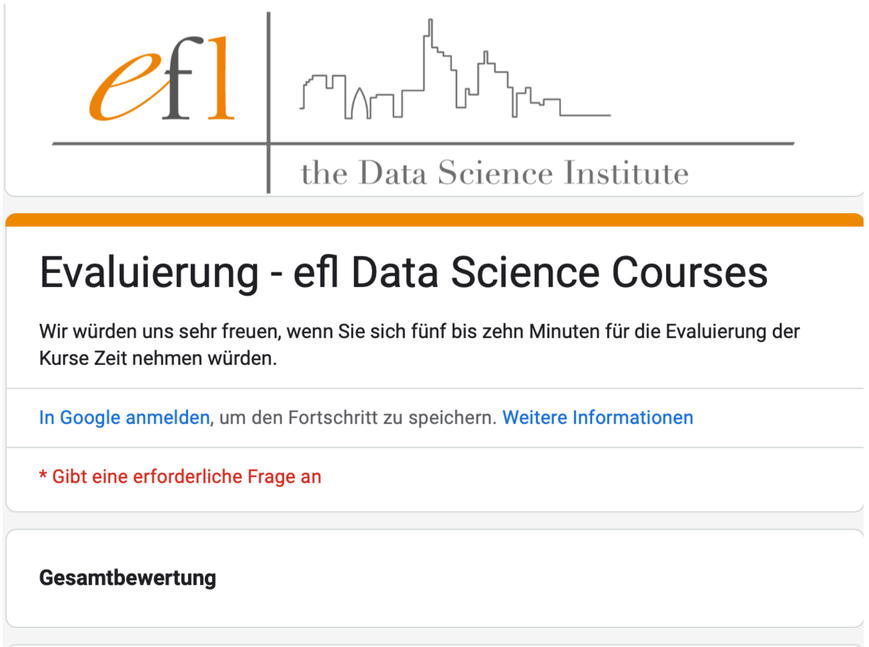
By Dr. Benjamin M. Henrich

Table of Content

Pre-alignment

After-Work-Drink? Wunschgetränk

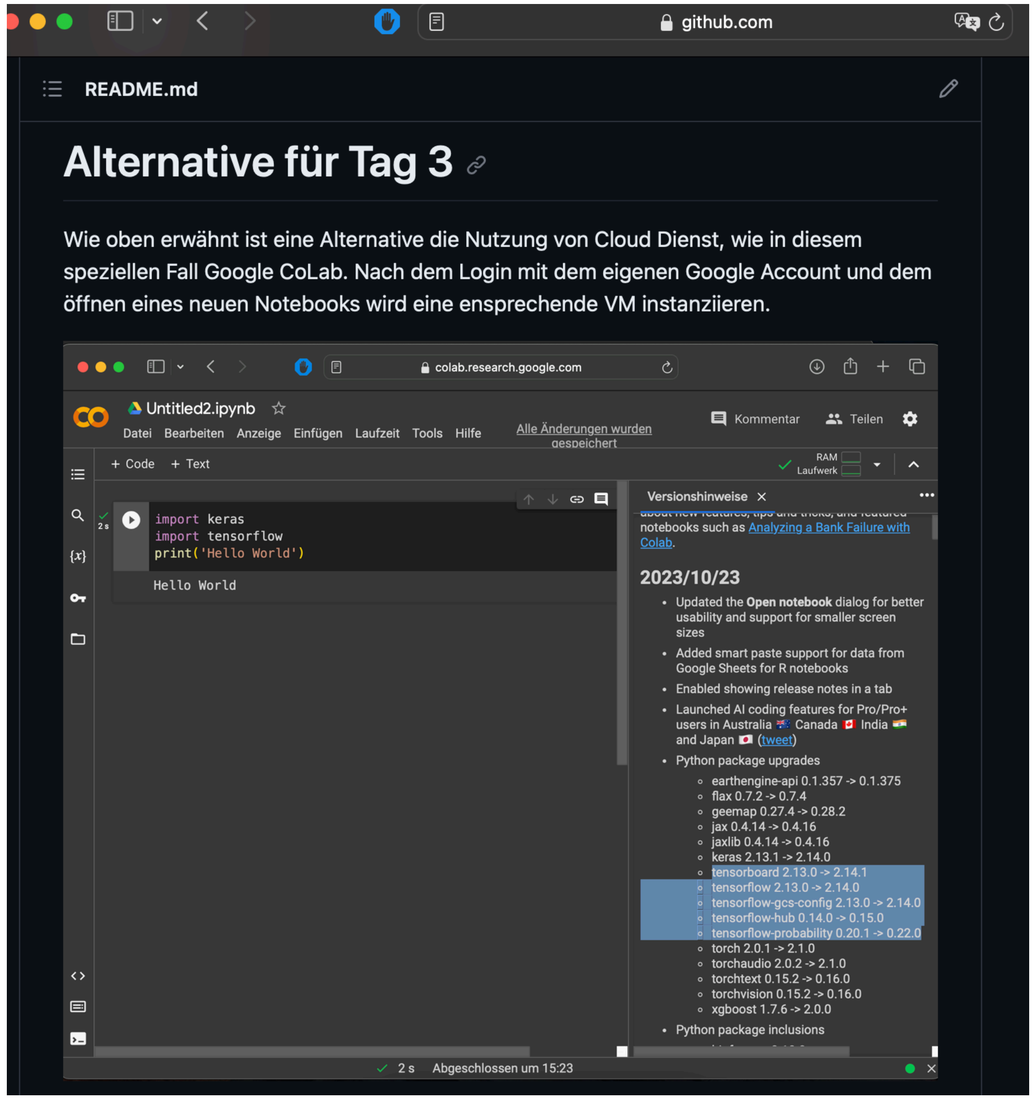
Survey



https://docs.google.com/forms/d/e/1FAIpQLSdH4RcSBN20rMX3bQjzfIt1JX_RjTijpiRebA_c062j83RFig/viewform

Alternative for Day 3

Google CoLab



After-Work-Drink? Wunschgetränk

Table of Content

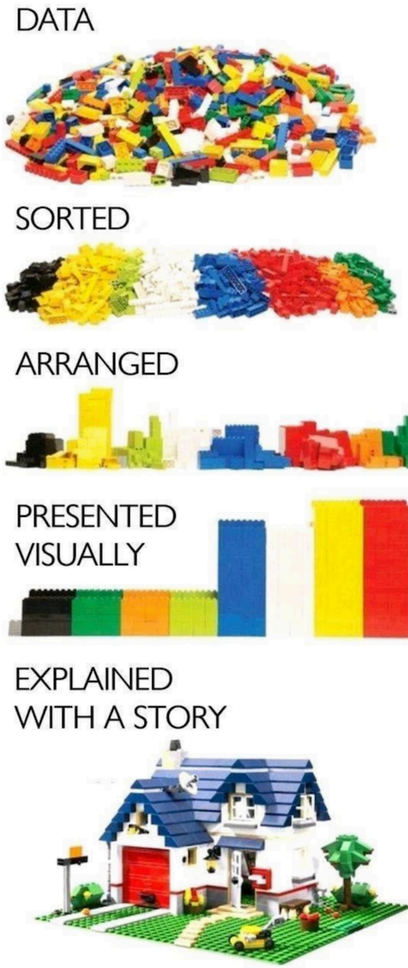
Data Science: But Why?

[1]

[2]

Raw Data

Knowlege



[1] https://www.google.com/url?sa=i&url=https%3A%2F%2Flustich.de%2Fbilder%2Fandere%2Fpapierberg%2F&psig=AOvVaw1shfkEZVN9kWASoXICbwAi&ust=1601110332932000&source=images&cd=vfe&ved=2ahUKEwiMxrL29oPsAhVUNewKHdqPD0MQr4kDegUIARDIAQ
[2] https://www.youtube.com/watch?v=n0ql-yeY9u0

Pre-Processing: What is That?

- Data pre-processing is a major and essential stage whose main goal is to **obtain final data sets** which can be considered correct and useful for further algorithms [1].

- The idea is **to filter the data** according to **quality criteria**, selected according to **possible target values**, and transformed correspondingly [2] .

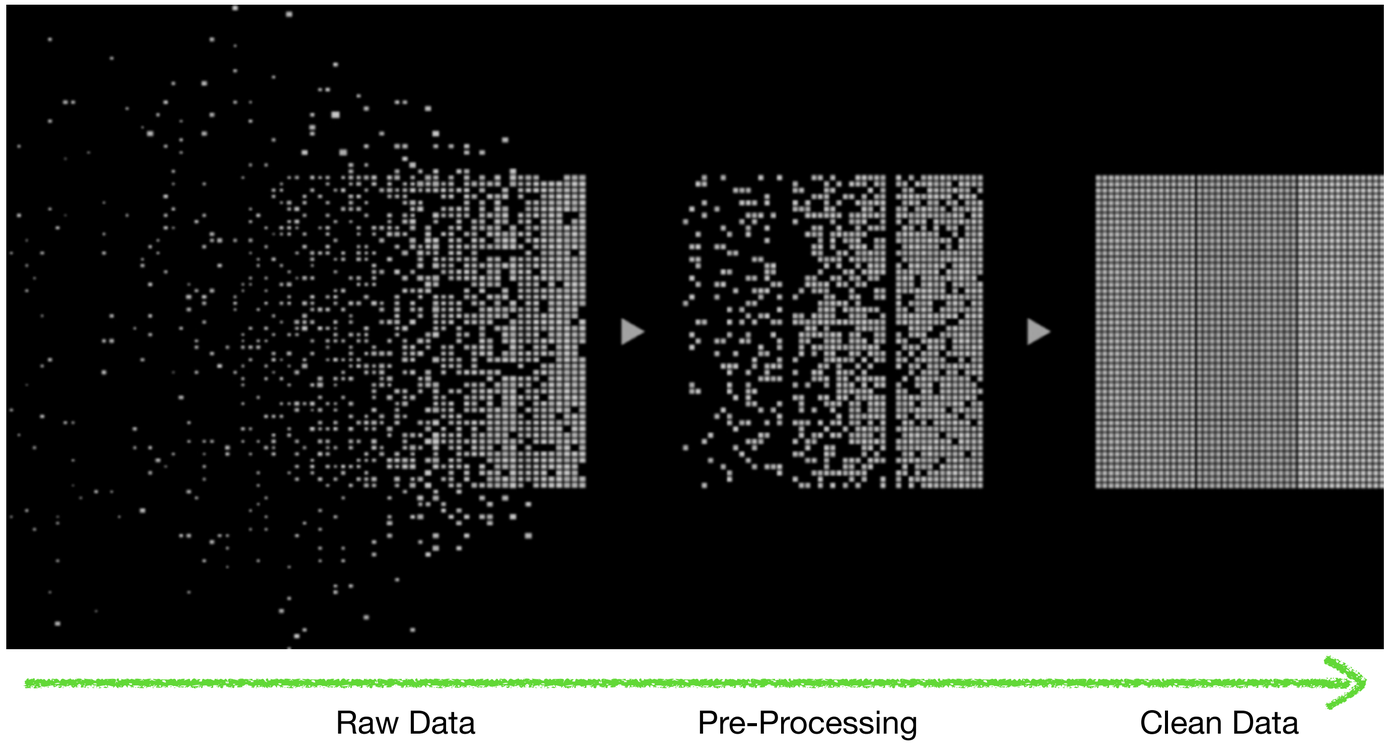
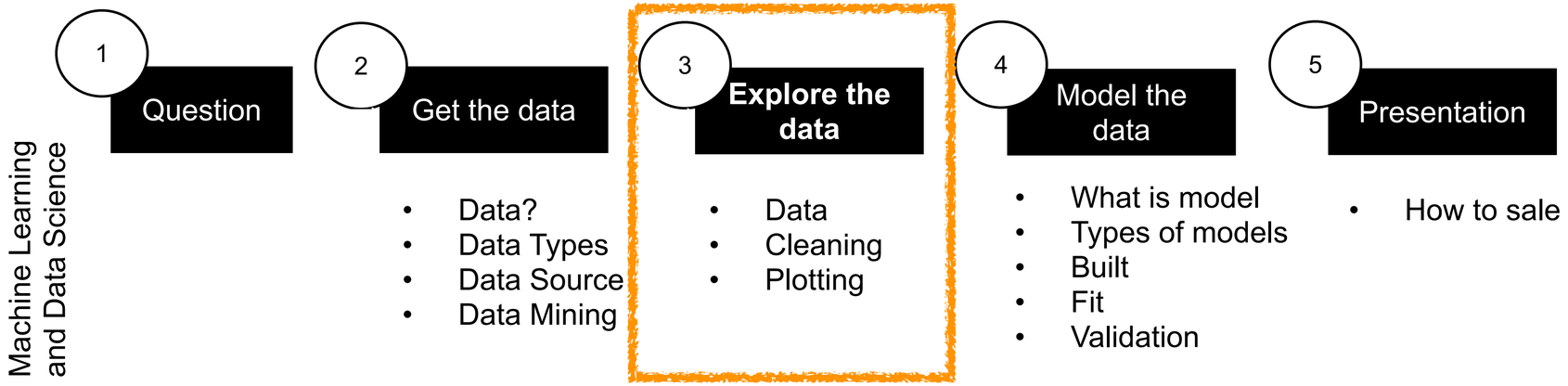


Figure: From Raw Data to Clean Data

[1] García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: methods and prospects. Big Data Analytics, 1(1), 1-22.
[2] Abdel-Karim, B. M., Pfeuffer, N., & Hinz, O. (2021). Machine learning in information systems-a bibliographic review and open research issues. Electronic Markets, 31, 643-670.

Pre-Processing as Part of the KDD Model



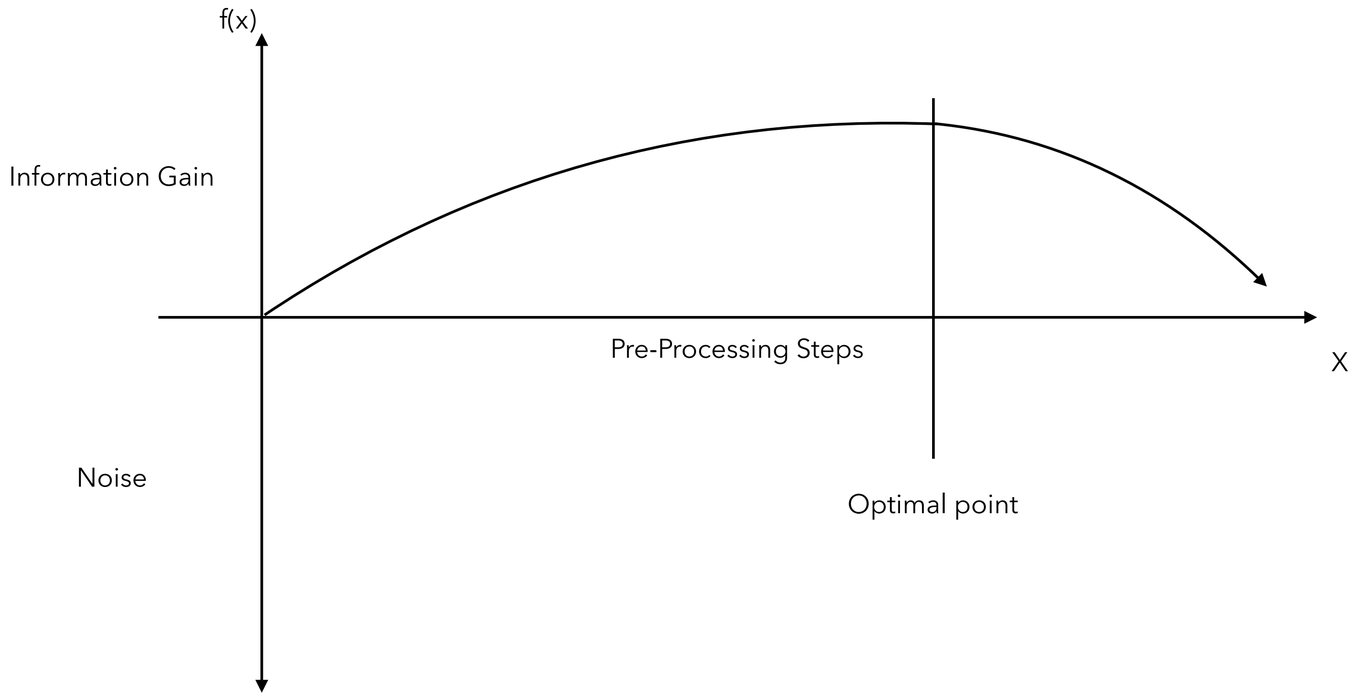
KDD Model: Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37-37.

Pre-Processing: General Node

- The preprocessing steps are highly dependent on the modeling approaches used.
- **There is no right or best preprocessing for all approaches**
- For example:
  - What measurement scales are excepted?
  - Can the approach deal with missing values
  - Is the approach robust with regard to outliers?
  - Is the approach robust with regard to differences in ranges?
  - Is the approach robust with regard to irrelevant features?
  - Is the approach robust with regard to redundant features?

Mishra, P., Biancolillo, A., Roger, J. M., Marini, F., & Rutledge, D. N. (2020). New data preprocessing trends based on ensemble of multiple preprocessing techniques. TrAC Trends in Analytical Chemistry, 132, 116045.

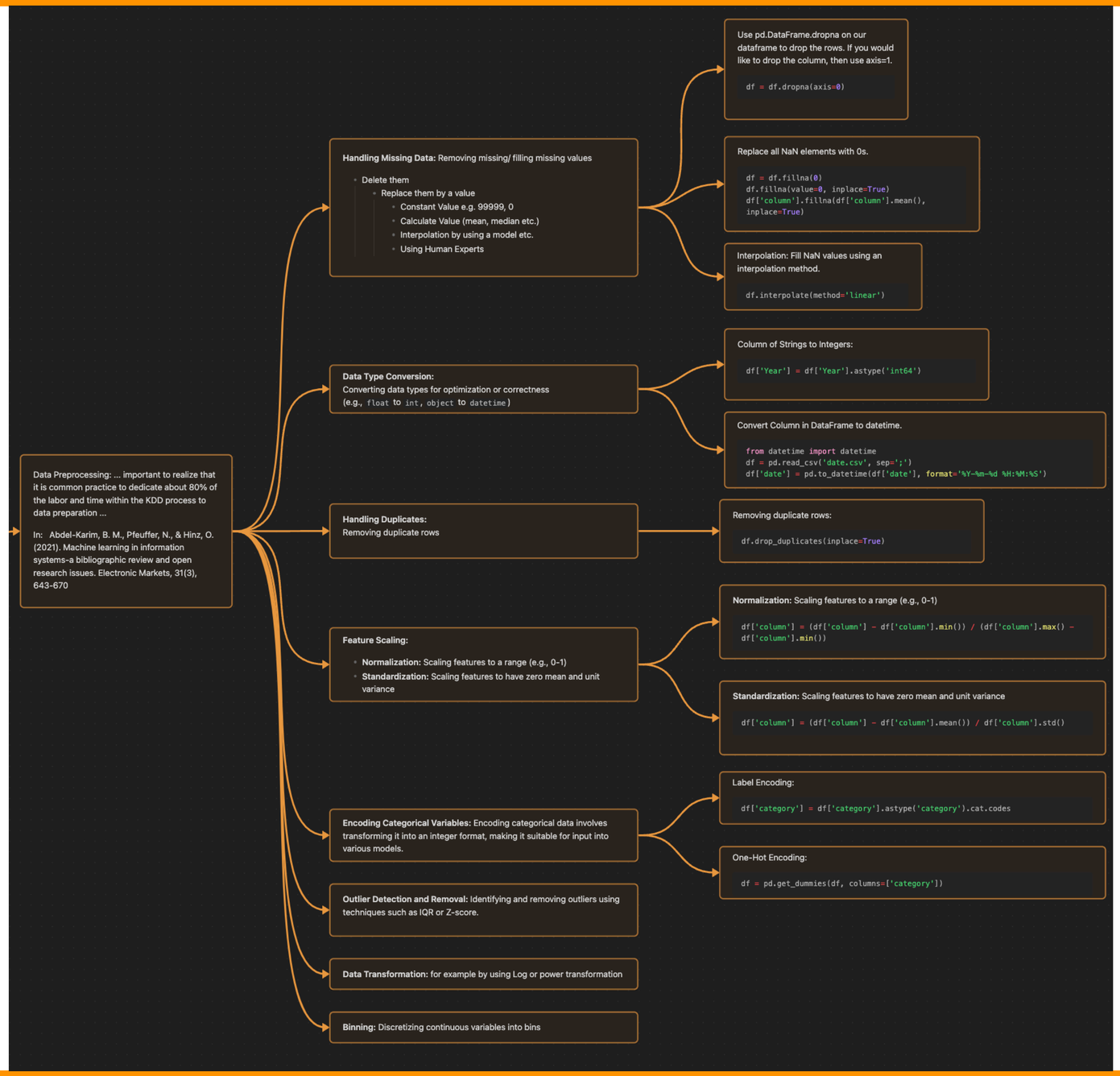
Data Pre-Processing is a Trade-Off:
Between Increasing Information Gain and Introducing Noise



Chicco, D. (2017). Ten quick tips for machine learning in computational biology. BioData Mining, 10(1), 35
Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer. Link

Overview

- **Handling Missing Data**
  - Removing missing values
  - Filling missing values
- **Data Type Conversion**
- **Handling Duplicates**
- **Encoding Categorical Variables**
  - Label Encoding
  - One-Hot Encoding
- **...**
- Feature Scaling
  - Normalization
  - Standardization
- Outlier Detection and Removal



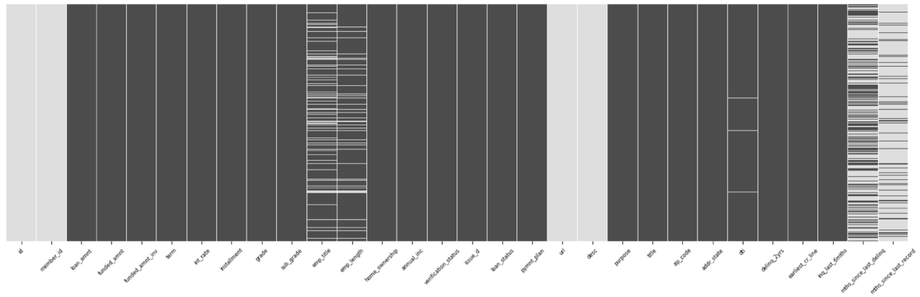
Alasadi, S. A., & Bhaya, W. S. (2017). Review of data preprocessing techniques in data mining. Journal of Engineering and Applied Sciences, 12(16), 4102-4107.
Fan, C., Chen, M., Wang, X., Wang, J., & Huang, B. (2021). A review on data preprocessing techniques toward efficient and reliable knowledge discovery from building operational data. Frontiers in energy research, 9, 652801.
García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J. M., & Herrera, F. (2016). Big data preprocessing: methods and prospects. Big data analytics, 1, 1-22.

Figure: Obsidian Mindmap

Handling Missing Data

- Find all missing values.
- We should get an overview of the missing values.

```
plt.figure(figsize=(15,5))
sns.heatmap(df.isnull(), cbar=False,
yticklabels=False, cmap='winter')
plt.xticks(rotation=45, fontsize=6)
plt.tight_layout()
plt.savefig('figures/
MissingValues.png')
plt.show()
```



Important Nodes about NA/ NAN/ Not-a-Number in Python in regards to the data types:

- In Python, you can create nan with float('nan'), math.nan, or np.nan. nan is considered a missing value in pandas., s_nan = pd.Series([float('nan'), math.nan, np.nan])
- None is also considered a missing value
- nan is a floating-point number float
- None in the object column remains as None
- String is not considered a missing value
- the strings 'NaN' and 'None' are not treated as missing values. The empty string '' is also not considered a missing value.
- f you want to treat certain values as missing, you can use the replace() method to replace them with float('nan'), np.nan, or math.nan., such as s_replace = s_str.replace(['NaN', 'None', ''], float('nan'))s_replace = s_str.replace(['NaN', 'None', ''], float(,nan'))

Handling Missing Data: Removing

```python
import pandas as pd
import numpy as np

# Sample DataFrame with missing values
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [55, np.nan, 35, 45, np.nan],
    'City': ['London', 'Los Angeles', np.nan, 'Chicago', 'New York']
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	City
0	Alice	55.0	London
1	Bob	NaN	Los Angeles
2	Charlie	35.0	NaN
3	David	45.0	Chicago
4	Eve	NaN	New York

Remove rows with any missing values:

`df_dropped_rows = df.dropna()` Deleting:

Remove columns with any missing values:

`df_dropped_columns = df.dropna(axis=1)` Deleting:

Figure: Meme About Replacing NaN

Handling Missing Value

```python
import pandas as pd
import numpy as np

# Sample DataFrame with missing values
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
    'Age': [55, np.nan, 35, 45, np.nan],
    'City': ['London', 'Los Angeles', np.nan, 'Chicago', 'New York']
}

df = pd.DataFrame(data)
print(df)
```

	Name	Age	City
0	Alice	55.0	London
1	Bob	NaN	Los Angeles
2	Charlie	35.0	NaN
3	David	45.0	Chicago
4	Eve	NaN	New York

Fill with a specific value:

```python
df_filled = df.fillna(0)
```

NaN => 0

Forward fill (propagate last valid observation forward):

```python
df_filled_forward = df.fillna(method='ffill')
```

Bob, **55.0**, Los Angeles
Charlie, 35.0, **Los Angeles**
Eve, **45.0**, New York

Backward fill (propagate next valid observation backward):

```python
df_filled_backward = df.fillna(method='bfill')
```

Bob, **35.0,** Los Angeles
Charlie, 35.0, **Chicago**
Eve, **NaN**, New York

Careful

Note: Interpolate missing data, which is especially useful for time series.

Handling Missing Values: Overview

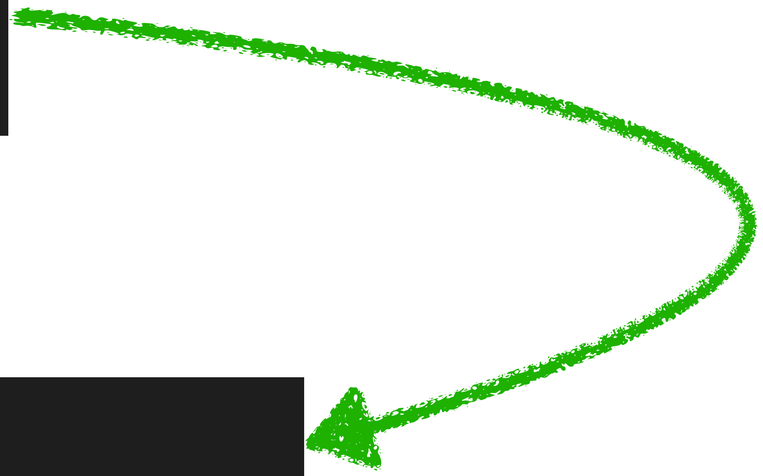
Method	Description	Code Example
Check for Missing Values	Identify missing values in the DataFrame	`df.isnull()`
Count Missing Values	Count missing values in each column	`df.isnull().sum()`
Drop Rows with NaN	Remove rows that have at least one missing value	`df.dropna()`
Drop Columns with NaN	Remove columns with missing values	`df.dropna(axis=1)`
Fill Missing Values	Replace NaN with a specific value (e.g., 0, mean, median)	`df.fillna(value)`
Forward Fill	Propagate last valid value forward	`df.fillna(method='ffill')`
Backward Fill	Use next valid value to fill missing values	`df.fillna(method='bfill')`
Fill with Column Mean	Replace NaN with column mean	`df['col'].fillna(df['col'].mean())`
Interpolate	Estimate missing values using interpolation	`df.interpolate()`
Check for Missing Rows	Filter rows that contain missing values	`df[df.isnull().any(axis=1)]`
Replace NaN	Replace NaN with a specific value	`df.replace(np.nan, 'value')`
Custom Imputation	Use custom logic for filling missing values	`df['col'].fillna(custom_function())`
Count Non-Missing Values	Get count of non-null values in each column	`df.notnull().sum()`
Detect Missing Values in Series	Check for NaN in a specific column or Series	`df['col'].isnull()`
Drop Rows with All NaN	Drop rows where all elements are NaN	`df.dropna(how='all')`
Drop Rows with Threshold	Keep rows with a minimum number of non-NaN values	`df.dropna(thresh=2)`

Notes:

- `fillna()` allows filling NaN values with constants, mean, median, or previous/next values.
- `dropna()` removes rows or columns containing NaN.
- `interpolate()` can estimate values using methods like linear, polynomial, or spline interpolation.

Data Type Conversion (Data Types of Each Column)

```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ID        4 non-null      object
 1   Price     4 non-null      object
 2   Date      4 non-null      object
 3   Category  4 non-null      int64
dtypes: int64(1), object(3)
memory usage: 260.0+ bytes
```



```
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ID        4 non-null      int64
 1   Price     4 non-null      float64
 2   Date      4 non-null      datetime64[ns]
 3   Category  4 non-null      category
dtypes: category(1), datetime64[ns](1), float64(1), int64(1)
memory usage: 356.0 bytes
None
```

```python
data = {
    'ID': ['1', '2', '3', '4'],  # IDs stored as strings
    'Price': ['99.5', '20.99', '15.75', '40'],  # Prices stored as strings
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04'],  # Dates stored as strings
    'Category': [1, 2, 1, 2]  # Categories stored as integers
}

df = pd.DataFrame(data)
print("Original DataFrame:\n", df)

print(df.info())

# Convert ID from string to integer:
df['ID'] = df['ID'].astype(int)
print("\nDataFrame after converting ID to integer:\n", df)

# Convert Price from string to float:
df['Price'] = df['Price'].astype(float)
print("\nDataFrame after converting Price to float:\n", df)

# Convert Date from string to datetime:
df['Date'] = pd.to_datetime(df['Date'])
print("\nDataFrame after converting Date to datetime:\n", df)

# Convert Category from integer to category:
df['Category'] = df['Category'].astype('category')
print("\nDataFrame after converting Category to categorical:\n", df)

print(df.info())
```

Figure: Data Type Conversion

Data Type Conversion: But Why?

Correct Data Representation:

- Data types ensure that each column in a DataFrame is stored in the most appropriate format. For example:
- Numerical Data: Converting columns to int or float ensures that numerical operations (like addition or averaging) are performed correctly.
- Categorical Data: Converting columns to category type can save memory and speed up operations if the column has a limited number of unique values.
- Date/Time Data: Converting columns to datetime type allows for time-based operations and manipulations.

Performance Optimization:

- Correct data types can lead to more efficient storage and faster computation. For instance:
- Memory Efficiency: Using category for columns with a limited number of unique values reduces memory usage compared to using object (which is typically a catch-all for string data).
- Speed: Numerical operations are faster with int or float types compared to object types, which are more general-purpose and less optimized for numeric computations.

Handling Duplicates

Why handling duplicates could be important:

- **Data Integrity:** Duplicates can compromise the accuracy and integrity of the data.
- **Accurate Analysis:** impact on statistical analysis and metrics.
- **Efficient Ressourcen Usage**

Option to Handle:

- **Removing Duplicates:** Remove duplicate rows based on all columns or specific ones.
- **Identifying Duplicates:** Find duplicate rows before deciding how to handle them.

Figure: The Matrix: Agent Smith's Replicating

https://www.cbr.com/the-matrix-agent-smith-replicating-powers-mysteries-explained/

Label Encoding

Label Encoding is a technique used to convert categorical variables into numerical values. Each unique category in a categorical variable is assigned an integer value.

Index	Height
0	Short
1	Medium
2	Tall

Index	Height
0	1
1	2
2	3

Figure: Label Encoding

Kuhn, M. (2013). Applied Predictive Modeling. Springer

One-Hot Encoding

One-Hot Encoding is a method used to convert categorical variables into a format (Binary Matrix) that can be provided to machine learning algorithms to improve model performance.

Index	Height
0	Short
1	Medium
2	Tall

Index	Short	Medium	Tall
0	1	0	0
1	0	1	0
2	0	0	1

Figure: One-Hot Encoding

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer google schola, 2, 1122-1128.

Table of Content


GEARS 5
REMNANT
FROM THE ASHES
CYBERPUNK
DIVINITY
ORIGINAL SIN
COUNTER STRIKE
TOTAL WAR
STARDEW VALLEY
DESTINY 2
MORDHAU
STEAM®

Case Study

What is Steam

- […] one of the most popular digital game delivery platform (Lin et. al., 2019, p. 1).
- Steam, one of the largest gaming networks for computer games in the world (Baumann et. Al., 2018, p. 1).

Baumann, F., Emmert, D., Baumgartl, H., & Buettner, R. (2018). Hardcore Gamer Profiling: Results from an unsupervised learning approach to playing behavior on the Steam platform. Procedia Computer Science, 126, 1289-1297.
Lin, D., Bezemer, C. P., Zou, Y., & Hassan, A. E. (2019). An empirical study of game reviews on the Steam platform. Empirical Software Engineering, 24, 170-207.

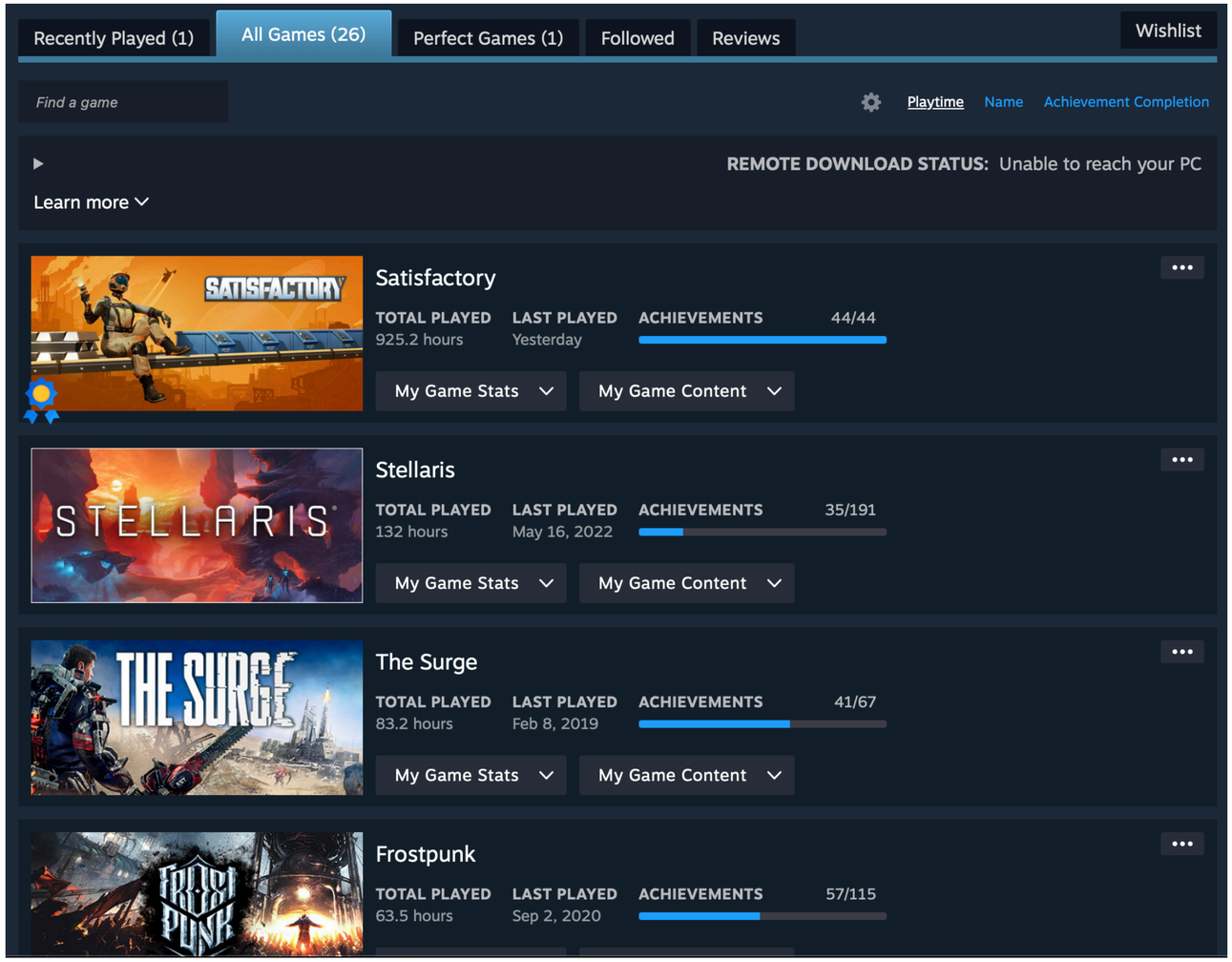
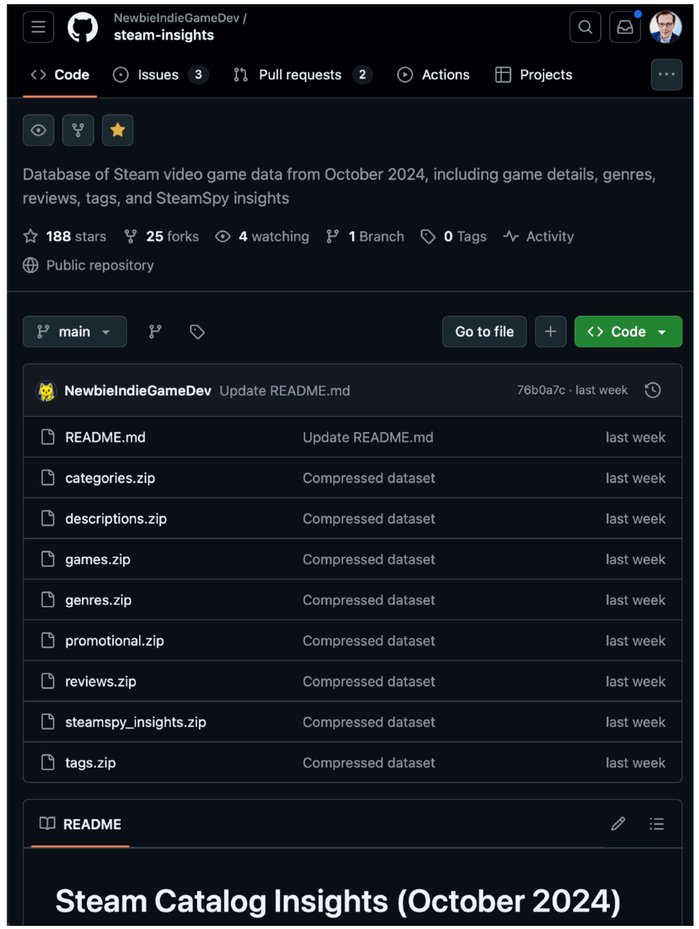


Figure: Steam Licensed Games

Data Sets: High Lebel View



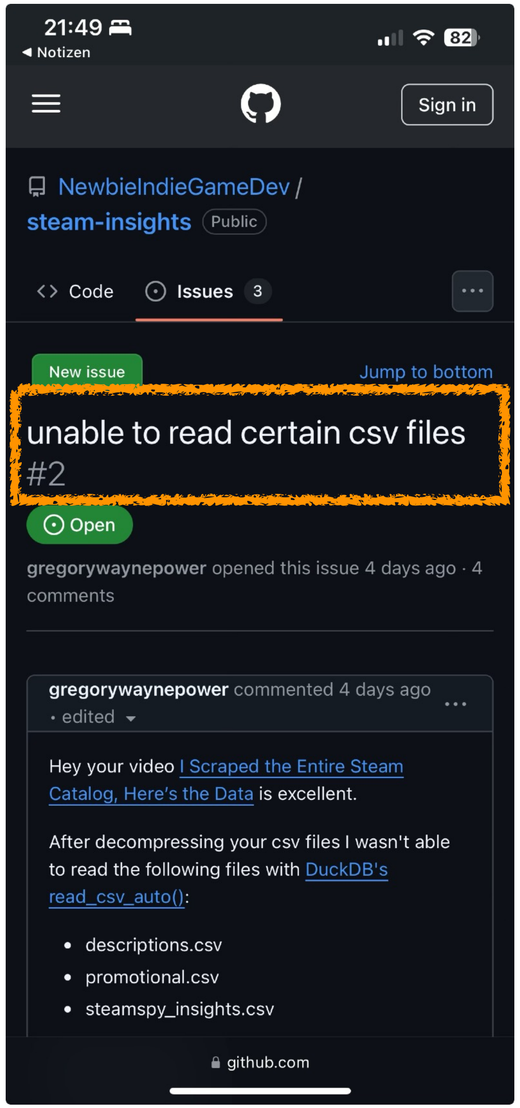
https://github.com/NewbieIndieGameDev/steam-insights

Figure: Git Repo for the Raw Data

https://www.youtube.com/watch?v=qiNv3qv-YbU

Figure: Corresponding YouTube Video

Limitation: Data Set is not Perfect



Pseudo „Jason“/ Dictionary stored in the excel!

games.csv

games.csv

```
1  "app_id","name","release_date","is_free","price_overview","languages","type"
2  "10","Counter-Strike","2000-11-01","0","{\"final\": 819, \"initial\": 819, \"currency\": \"EUR\", \"final_formatted\":
   \"8,19€\", \"discount_percent\": 0, \"initial_formatted\": \"\"}","English<strong>*</strong>, French<strong>*</strong>,
   German<strong>*</strong>, Italian<strong>*</strong>, Spanish - Spain<strong>*</strong>, Simplified Chinese<
   strong>*</strong>, Traditional Chinese<strong>*</strong>, Korean<strong>*</strong><br><strong>*</strong>languages with
   full audio support","game"
```

HTML Code stored as well

Do you have any idea why we see this kind of raw data in the .csv?

Figure: CSV Files not Perfect, see Project Issues.

Your Chance

- A) What are the top ten and lowest ten games according to the owners? (Tip: Use owners_conservative.)
- B) Which publishers are the most successful in the dataset based on the number of published games?
- C) How does the number of published games change over time?
- D) What do the prices look like?

Appendix

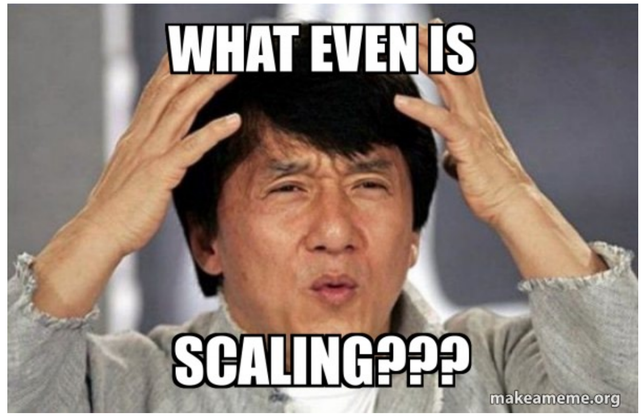
Feature Scaling: Definition

Feature scaling is the process of adjusting the features of a dataset to a uniform scale, ensuring that the differences between values are maintained without distortion.

Normalization: **S**cale the values of numerical feature into a common scale without distorting differences in the ranges of values or losing information. **Min-Max Scaler:** transforms the features to a specific range, **typically between 0 and 1.**

Standardization, also known as **z-score normalization**, is a scaling method **that centers the values of a feature around its mean, while ensuring a unit standard deviation**. Ready to use is: Standard-Scaler in sklearn.

S**tandardization does not always improve the model performance**. Effectiveness depends on main working principle of the algorithms. For instance could help by distance calculation, such as K-means, Support Vector Machine (SVM), and Artificial Neural Networks (ANN) to reduce the number space.



https://makeameme.org/meme/what-even-is-5cda27

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Grandvalet, Y., & Canu, S. (2002). Adaptive scaling for feature selection in SVMs. Advances in neural information processing systems, 15.
Murphy, K. P. (2012). Machine Learning–A probabilistic Perspective. The MIT Press.

Feature Scaling: General Example

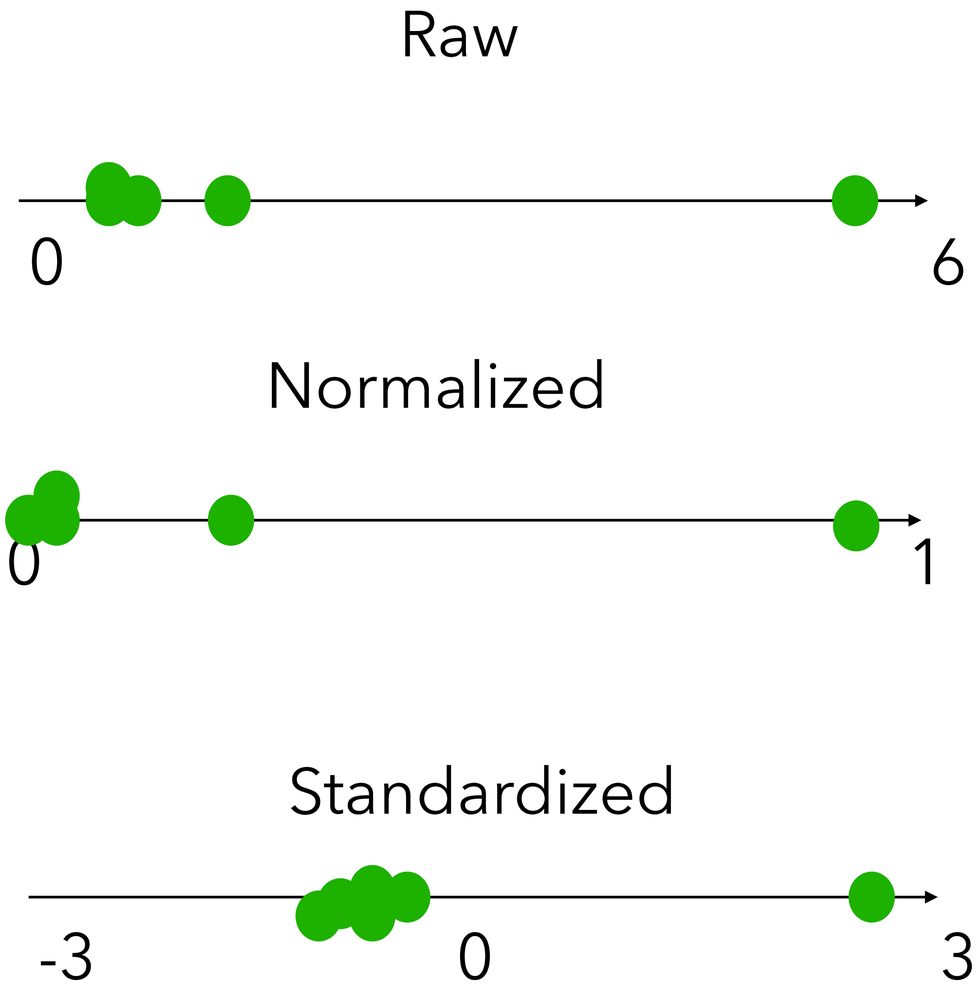


Figure: Illustration of Raw Data, Normalized Data, and Standardized Data

Scaling Method vs. Model: Some Ideas

Scaling Method	Description	When to Use	Research Papers
Normalization (Min-Max Scaling)	Scales features to a fixed range, typically [0, 1].	- **Neural Networks**: Helps speed up convergence and stabilize training. - **k-Nearest Neighbors (k-NN)**: Ensures all features contribute equally to distance calculations.	- LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. (2012). *Efficient BackProp*. In *Neural Networks: Tricks of the Trade*. Springer. - Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. (2007). *Data Preprocessing for Supervised Learning*. International Journal of Computer Science and Applications.
Standardization (Z-score Normalization)	Scales features to have a mean of 0 and a standard deviation of 1.	- **Support Vector Machines (SVM)**: Improves performance by ensuring features are on the same scale. - **Principal Component Analysis (PCA)**: Helps in accurate principal component computation. - **Linear Regression**: Improves convergence and model performance.	- Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). *A Practical Guide to Support Vector Classification*. Technical Report. - Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics. - Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. (2012). Efficient BackProp. In Neural Networks: Tricks of the Trade (pp. 9-48). Springer.
Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. (2007). Data preprocessing for supervised learning. International Journal of Computer Science and Applications, 4(1), 111-117.

Hsu, C.-W., Chang, C.-C., & Lin, C.-J. (2003). A practical guide to support vector classification. Technical Report. Retrieved from https://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer Series in Statistics.
Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

Outlier Detection and Removal

Outliers are data points that significantly deviate from the rest of the data, and they can arise due to various reasons such as measurement errors, variability in the data, or genuine anomalies.

Note: There, is no clear definition of outliers, it depends on the interoperation!

Characteristics of Outliers:

Unusual: Outliers are distinctly different from other data points.

Influential: They can heavily influence statistical analyses, such as mean and variance.

Context-Dependent: What constitutes an outlier can vary depending on the context of the data and the specific analysis being performed.

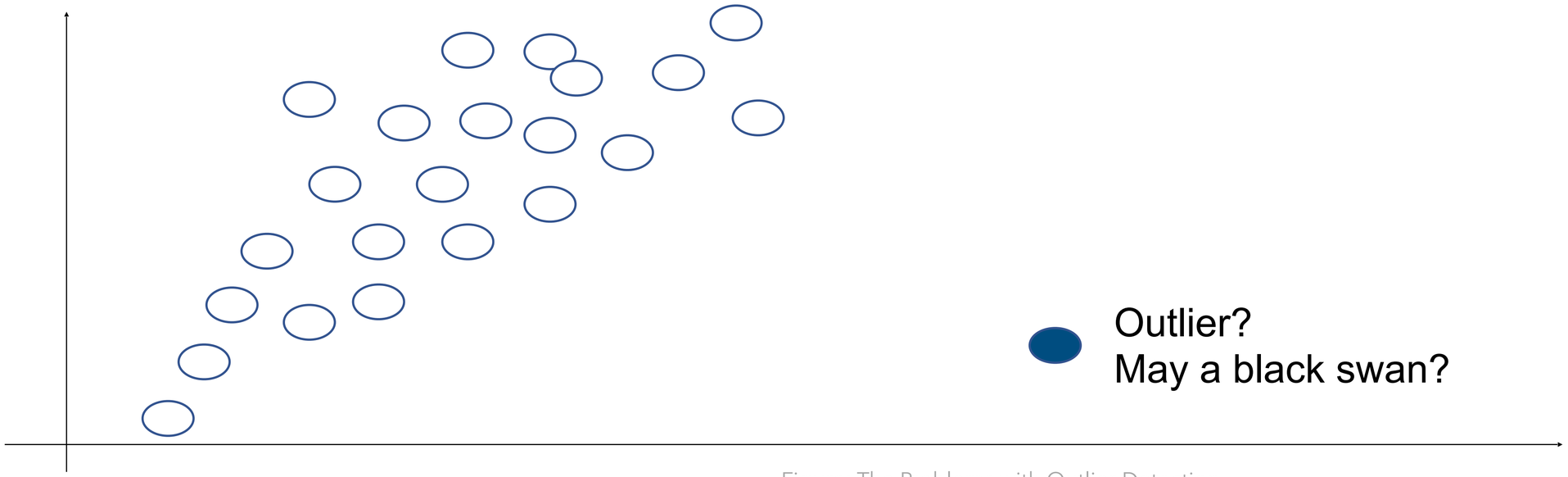


Figure: The Problems with Outlier Detection

Hodge, V. J., & Austin, J. (2004). A survey of outlier detection methodologies. Artificial Intelligence Review, 22(2), 85-126.

Find Outliers: Have a Look

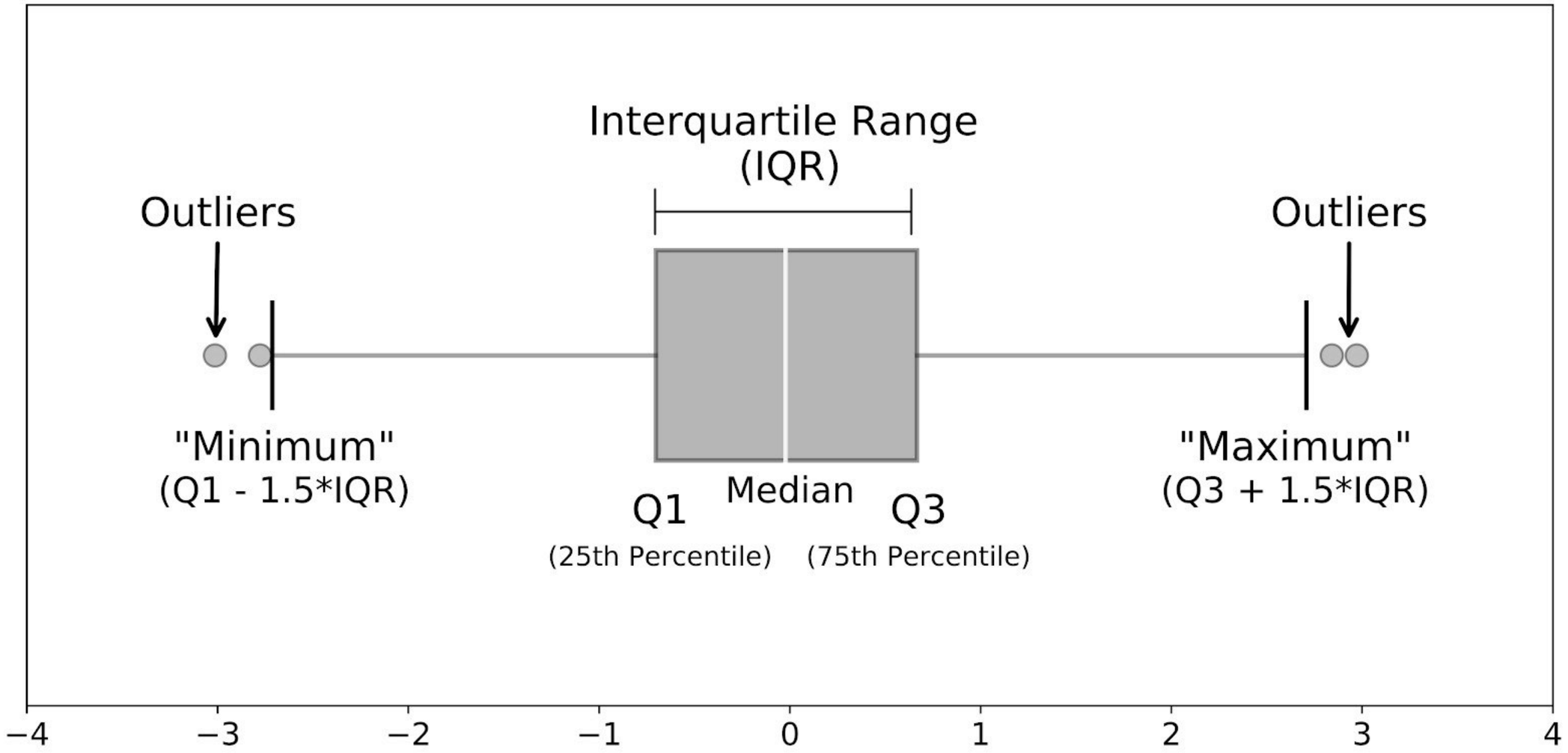


Figure: Boxplot Example

https://neptune.ai/blog/data-preprocessing-guide

Example Approach

Calculation: mean value +/- 1.98 of the standard deviation

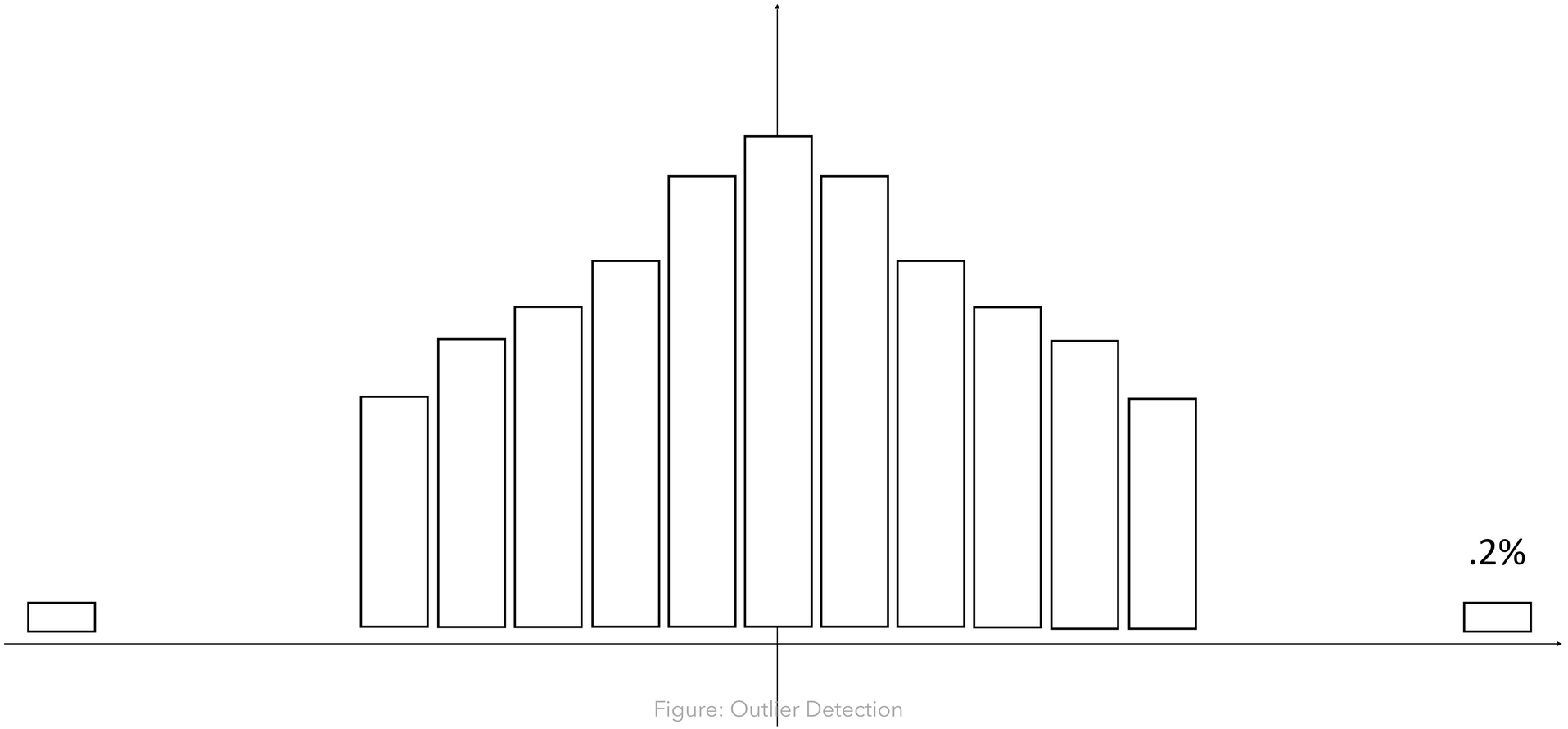


Figure: Outlier Detection

Outlier Detection and Removal

- Statistical Methods:
  - **Z-Score**: Measures how many standard deviations a data point is from the mean of the data. A high absolute Z-score indicates an outlier.
- **Interquartile Range (IQR):**
  - Defines outliers as those lying below Q1 - 1,5* IQR or Q3+1,5 * IQR above
- Machine Learning-Based Methods: K-NN, SVM …
- Visual Methods:
  - Boxplot
  - Scatterplot

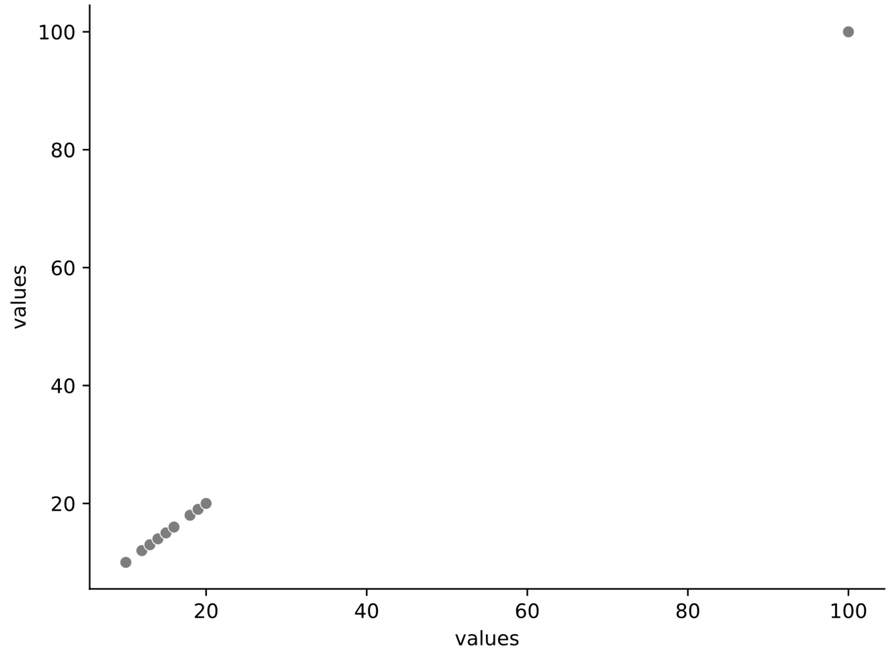


Figure: Outlier Detection

Outlier Detection: Limitation

- **Context Dependency:** The effectiveness of outlier detection methods often depends on the context of the data and the specific application, making it necessary to understand the data characteristics before choosing a method.
- **Statistical Methods:** May assume normality or symmetry and can be sensitive to outliers in the computation of summary statistics.
- **Machine Learning-Based Methods:** May require careful parameter tuning and can be computationally expensive.
- **Distance-Based Methods:** Sensitive to parameter choices and may struggle with high-dimensional data.
- **Visual Methods:** Limited to specific scenarios and can be subjective in interpretation.

Alghushairy, O., Alsini, R., Soule, T., & Ma, X. (2020). A review of local outlier factor algorithms for outlier detection in big data streams. *Big Data and Cognitive Computing*, *5*(1), 1.

Barbato, G., Barini, E. M., Genta, G., & Levi, R. (2011). Features and performance of some outlier detection methods. *Journal of Applied Statistics*, *38*(10), 2133-2149.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, *22*, 85-126.

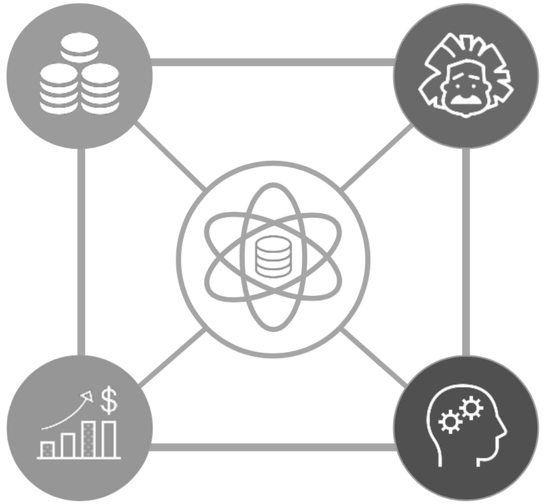
From Data to Machine Learning … Easy? No!

Database Management	Machine Learning
Database is an active, evolving entity	Database is just a static collection of data
Records may contain missing or erroneous information	Instances are usually complete and noise-free
A typical field is numeric	A typical feature is binary
A database typically contains millions of records	Data sets typically contain several hundred instances
AI should get down to reality	"Databases" is a solved problem and is therefore uninteresting

Table 2. Conflicting Viewpoints between Database Management and Machine Learning. [1]

Needs to shape the data into the right form

Data Engineer
- → Responsible for the acquisition, organization and storage of data
- → Possible skills: Software Engineering, SQL, Hadoop, Python(Pandas)



General Data Scientist
- → Coding Generalist who extracts knowledge from data and creates business value with the full data science stack
- → Possible Skills: Domain Knowledge, Statistics, Python/R, Pandas/Numpy, Machine Learning, Data Visualization, etc.

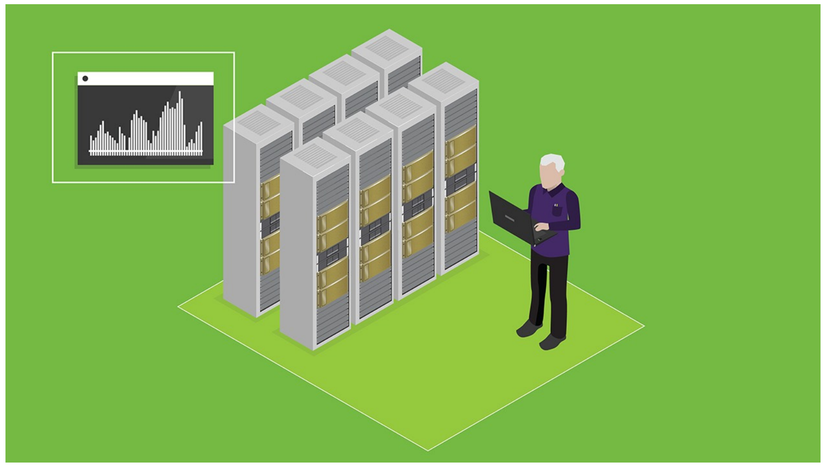
Data Analyst
- → Responsible for the creation of business value with data
- → Possible Skills: Domain Knowledge, Python/R, SQL, Excel, Data Visualization

Machine Learning Engineer
- → Specialist in the field of Machine Learning; Creator of Algorithms
- → Possible Skills: Mathematics/Statistics, Software Engineering, Python, C++, Pandas/Numpy/Pytorch, Machine Learning, Neural Networks, etc.

[1] Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: An overview. AI magazine, 13(3), 57-57.

Pre-Processing: Challenge Workflow

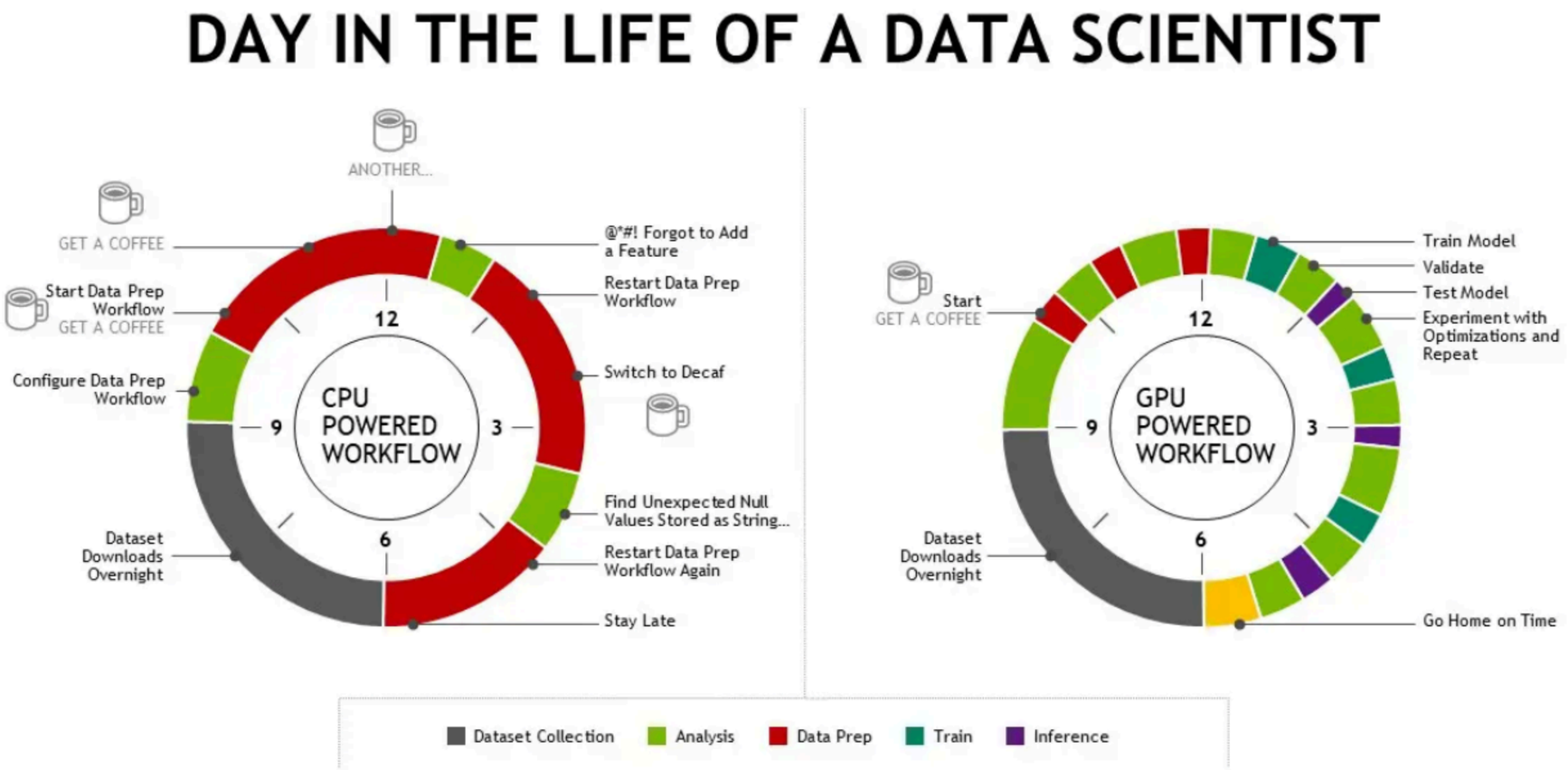
„Local“ Development



Server Deployment

Use in Production

Pre-Processing: Critical Reflection



Therefore, it is important to realize that it is common practice to dedicate about 80% of the labor and time within the KDD process to data preparation, as a representative survey on a popular data science portal shows. https://bit.ly/2WwVPho.

In: Abdel-Karim, B. M., Pfeuffer, N., & Hinz, O. (2021). Machine learning in information systems-a bibliographic review and open research issues. Electronic Markets, 31(3), 643-670

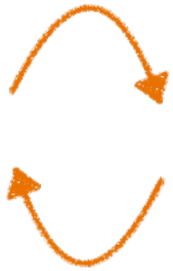
https://blogs.nvidia.com/blog/2018/11/15/accelerated-data-science-hpc/

Domain Knowledge: Why is this Important?

Domain knowledge can be define as knowledge about the environment in which the target implementation operates [1].

Data Science Team

Data Scientist/ Data Analyst



Need each other

Business side

[1] Hjørland, B. & Albrechtsen, H. (1995). Toward A New Horizon in Information Science: Domain Analysis. Journal of the American Society for Information Science, 1995, 46(6), p. 400–425.

Images:
https://www.theforage.com/blog/careers/investment-banking
https://unsplash.com/de/s/fotos/Geschäft
https://stock.adobe.com/de/images/nerd-at-the-computer-side-view-of-young-nerd-man-working-at-the/55780370