



INTRODUCTION TO DATA ANALYTICS

Data Processing

Asst. Prof. Dr. Rathachai Chawuthai

Department of Computer Engineering

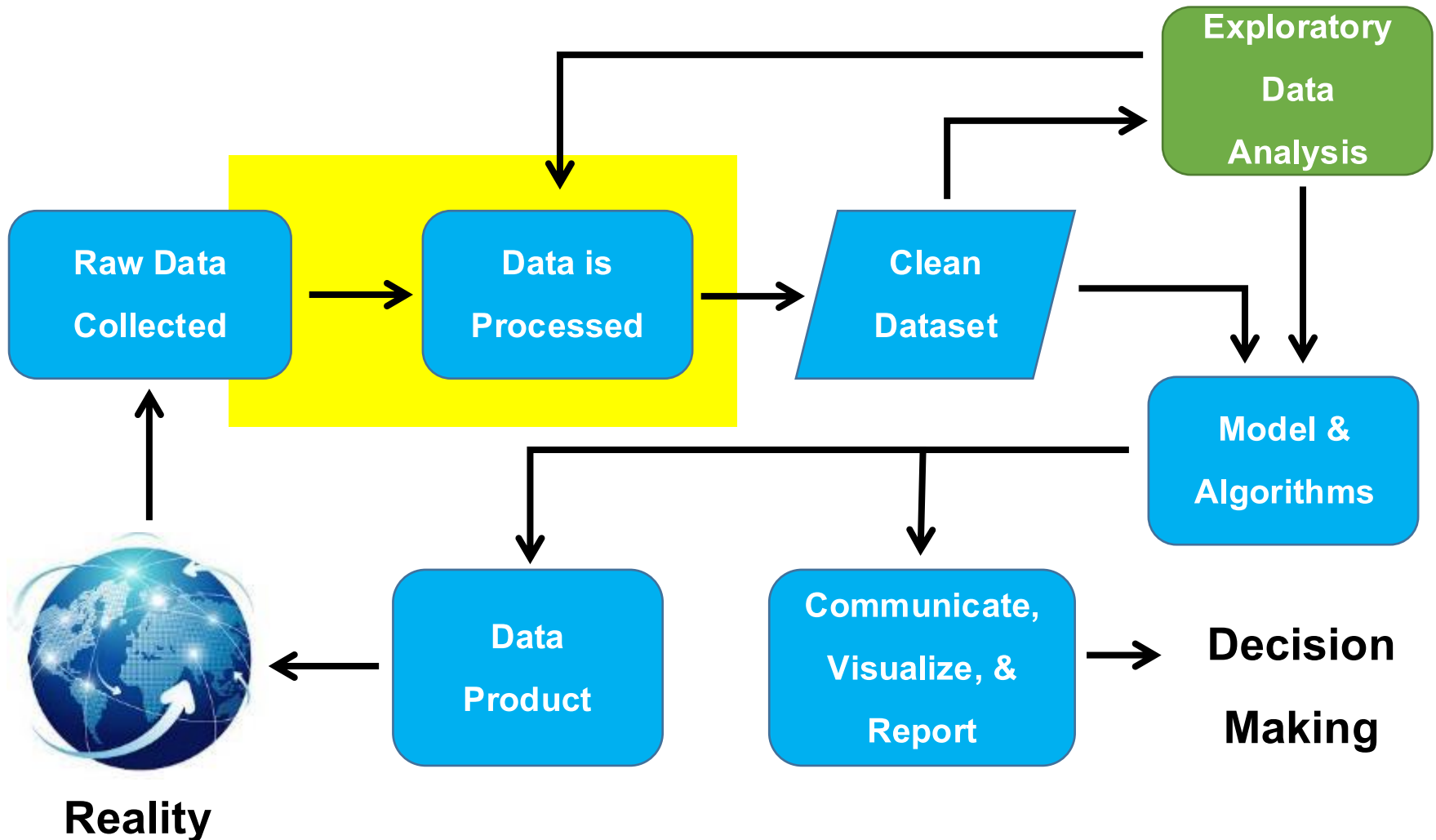
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Agenda

- Data Processing
- Data Cleansing
- Data Integration
- Data Transformation
- Feature Engineering
- Pipeline

Data Science Process



Data Processing



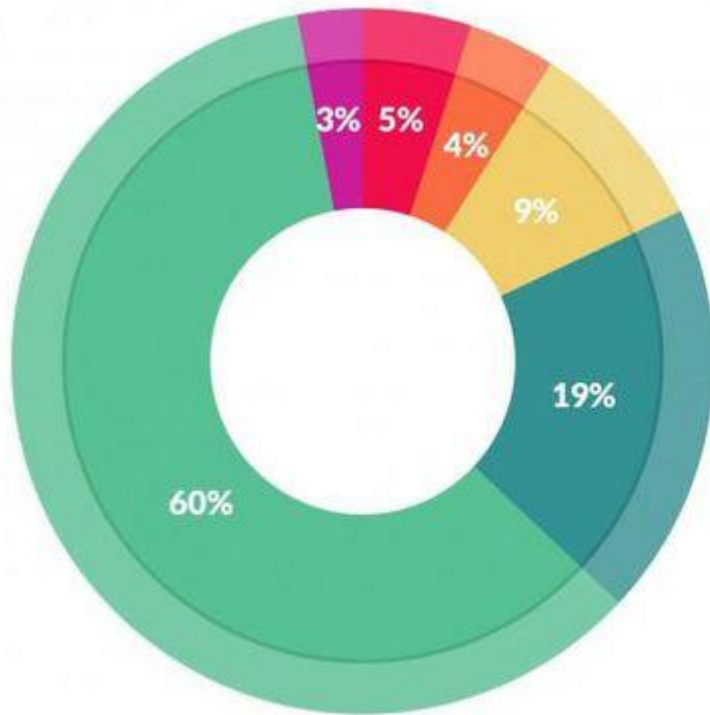
Is Data Beautiful?



Why Is Data Dirty?

- Incomplete data comes from
 - n/a data value when collected
 - different consideration between the time when the data was collected and when it is analyzed.
 - human/hardware/software problems
- Noisy data comes from the process of data
 - collection
 - entry
 - transmission
- Inconsistent data comes from
 - Different data sources
 - Functional dependency violation

Spending the most time doing ...



What data scientists spend the most time doing

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets; 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

Tasks

- Data Cleaning
 - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data Integration
 - Integration of multiple databases, data cubes, or files
- Data Transformation
 - Normalization and aggregation

Data Cleansing



Data Cleaning

- Importance
 - “Data cleaning is one of the three biggest problems in data warehousing”—Ralph Kimball
 - “Data cleaning is the number one problem in data warehousing”—DCI survey
- Data cleaning tasks
 - Fill in missing values
 - Identify outliers and smooth out noisy data
 - Correct inconsistent data
 - Resolve redundancy caused by data integration

Missing Data

- Data is not always available
 - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
 - equipment malfunction
 - inconsistent with other recorded data and thus deleted
 - data not entered due to misunderstanding
 - certain data may not be considered important at the time of entry
 - not register history or changes of the data
- Missing data may need to be inferred.

How to Handle Missing Data?

- Ignore the tuple usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably.
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
 - a global constant : e.g., “**unknown**”, a new class?!
 - the attributes’ mean
 - the attribute’s mean for all samples belonging to the same class: smarter
 - the most probable value: inference-based such as Bayesian formula or decision tree

Options for Dealing With Missing Data

Missing data in general is one of the trickier issues that is dealt with when cleaning data. Broadly there are two solutions:

- **Deleting/Ignoring rows with missing values**
 - Missing Value Handling: The simplest solution is to remove records with missing values, but only if the proportion is small (around 10%).
 - Dataset Size Consideration: Deleting too many rows (more than 10%) can negatively impact the model's performance.
- **Filling in the Values**
 - Handling Missing Data: Filling missing values with a specific value.
 - Value Selection: Depends on data type, such as using common value (rather than “unknown”) for categorical data or the mode for numerical data.

Educated Guessing

- It sounds arbitrary and isn't your preferred course of action, but you can often infer a missing value. For related questions, for example, like those often presented in a matrix, if the participant responds with all "4s", assume that the missing value is a 4.

	Strongly Disagree 1	2	3	4	Strongly Agree 5
The Chipotle website has a clean and simple presentation.	☐	☐	☐	☒	☐
The Chipotle website is easy to use.	☐	☐	☐	☒	☐
I find the Chipotle website to be attractive.	☐	☐	☐	☒	☐
The Chipotle website is trustworthy.	☐	☐	☐	☒	☐
It is easy to navigate within the Chipotle website.	☐	☐	☐	☐	☐
The Chipotle website's capabilities meet my requirements.	☐	☐	☐	☒	☐
The information on the Chipotle website is credible.	☐	☐	☐	☒	☐
I will likely return to the Chipotle website in the future.	☐	☐	☐	☒	☐

Noisy Data

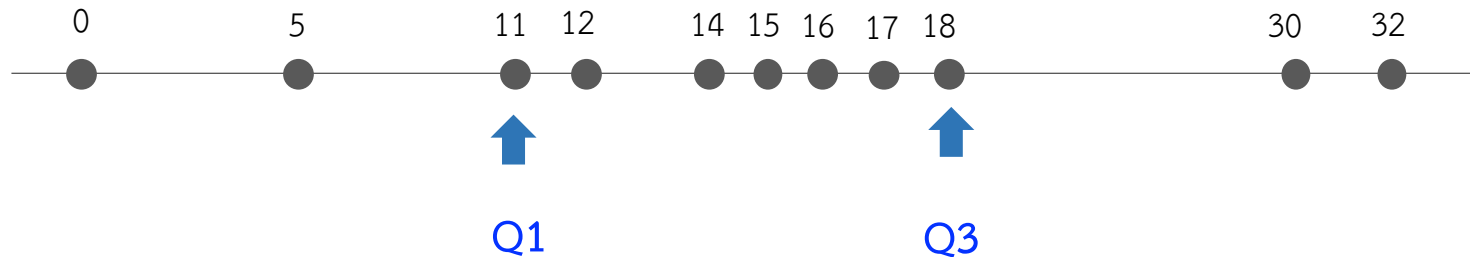
- Noise: random error or variance in a measured variable
- Incorrect attribute values may be due to
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention
- Other data problems which require data cleaning
 - duplicate records
 - incomplete data
 - inconsistent data

Use Interquartile to remove noisy data

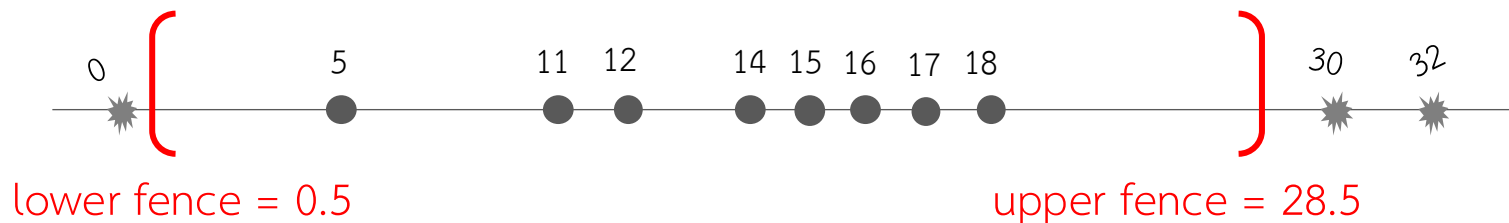


1. $IQR = Q3 - Q1$
2. Lower Fence = $Q1 - 1.5 \times IQR$
3. Upper Fence = $Q3 + 1.5 \times IQR$

Use Interquartile to remove noisy data



1. $IQR = Q3 - Q1 = 18 - 11 = 7$
2. $Lower\ Fence = Q1 - 1.5 \times IQR = 11 - 1.5 \times 7 = 0.5$
3. $Upper\ Fence = Q3 + 1.5 \times IQR = 18 + 1.5 \times 7 = 28.5$



Data Integration



Data Integration

- Data integration:
 - combines data from multiple sources into a coherent store
- Schema integration
 - integrate metadata from different sources
 - Entity identification problem: identify real world entities from multiple data sources, e.g., $A.cust-id \equiv B.cust-\#$
- Detecting and resolving data value conflicts
 - for the same real-world entity, attribute values from different sources are different
 - possible reasons: different representations, different scales, e.g., metric vs. British units

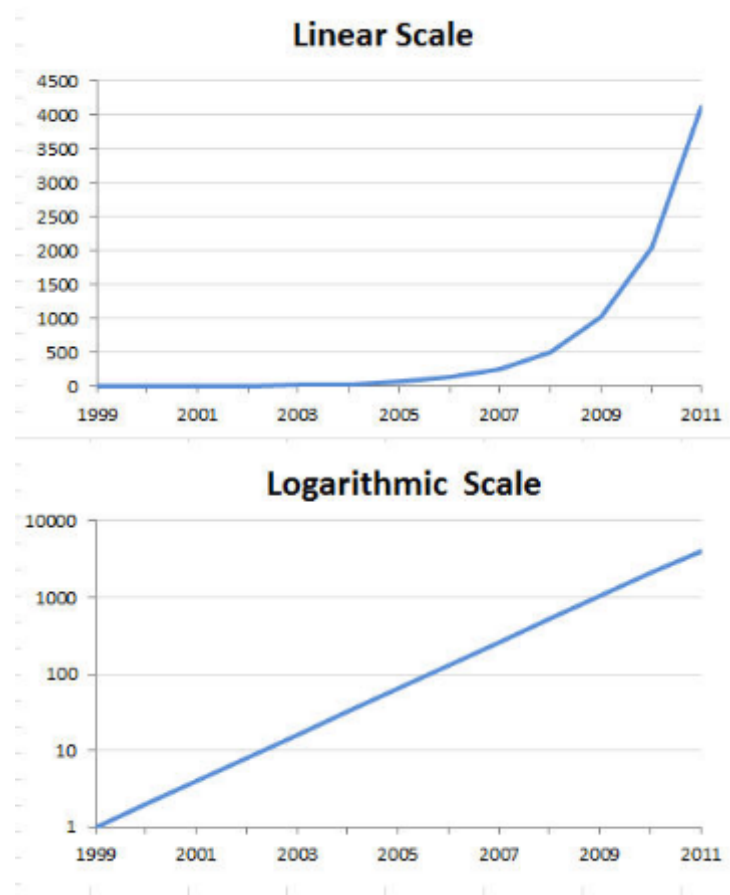
Data Transformation



Data Transformation

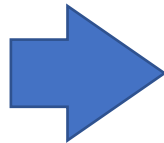
- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
 - min-max normalization
 - z-score normalization
 - normalization by decimal scaling
- Attribute/feature construction
 - New attributes constructed from the given ones

Reshape



Sparse Matrix

User	Movie	Rating
u1	m1	3
u1	m2	4
u2	m1	2
u2	m2	4
u2	m3	5
u3	m4	4
u3	m5	2
u4	m4	5
u4	m5	2

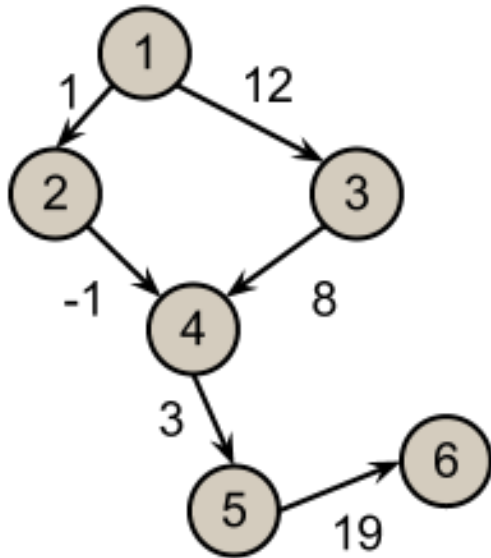


	m1	m2	m3	m4	m5
u1	3	4			
u2	2	4	5		
u3				4	2
u4				5	2

Document to Vector

	<i>D1</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>	<i>D5</i>	...
<i>a</i>	145	223	346	78	89	...
<i>abandon</i>	4	0	0	5	3	...
<i>ability</i>	5	10	0	4	7	...
<i>able</i>	31	35	64	3	5	...
<i>about</i>	64	68	89	24	9	...
<i>above</i>	4	5	8	0	0	...
<i>abroad</i>	0	0	1	0	0	...
<i>absence</i>	2	4	0	0	0	...
<i>absent</i>	0	0	1	0	0	...
<i>absolute</i>	3	1	5	0	1	...
<i>abstract</i>	5	1	2	1	0	...
<i>abuse</i>	0	1	0	0	0	...
<i>academic</i>	1	3	0	0	0	...
...	...	...	...	...	...	...

Adjacency Matrix



Weighted Directed Graph

	①	②	③	④	⑤	⑥
①	0	1	12	0	0	0
②	-1	0	0	-1	0	0
③	-12	0	0	8	0	0
④	0	1	-8	0	3	0
⑤	0	0	0	-3	0	19
⑥	0	0	0	0	-19	0

Adjacency Matrix

Feature Engineering



Feature Engineering

- Imputation
- Handling Outliers
- Binning
- Log Transform
- One-Hot Encoding
- Grouping Operations
- Feature Split
- Scaling
- Extracting Date

Imputation

- Filling missing values
- **Numerical Imputation**
 - Replace with 0 is not a good solution
 - Mean may be better
 - Mean of the group
 - Same in cluster
 - Regression
- **Categorical Imputation**
 - Replacing the missing values with the **maximum occurred value**.
 - If values are distributed uniformly and there is not a dominant value, imputing a category like “**Other**” might be more sensible, because in such a case, your imputation is likely to converge a random selection.

Handling Outliers

- These will detect them using standard deviation, and percentiles.
- **Outlier Detection with Standard Deviation**
 - If a value has a distance to the average **higher than x * standard deviation**, it can be assumed as an outlier. Then what x should be?
- **Outlier Detection with Percentiles**
 - To assume a certain percent of the value from the top or the bottom as an outlier. The key point is here to set the percentage value once again, and this depends on the distribution of your data.
 - Detect data between **5th and 95th percentiles**.
 - An Outlier Dilemma: **Drop** or **Cap**

Binning

- The main motivation of binning is to make the model more robust and prevent overfitting, however, it has a cost to the performance. Every time you bin something, you sacrifice information and make your data more regularized.

- **Numerical Binning Example**

<u>Value</u>	<u>Bin</u>
0-30 ->	Low
31-70 ->	Mid
71-100 ->	High

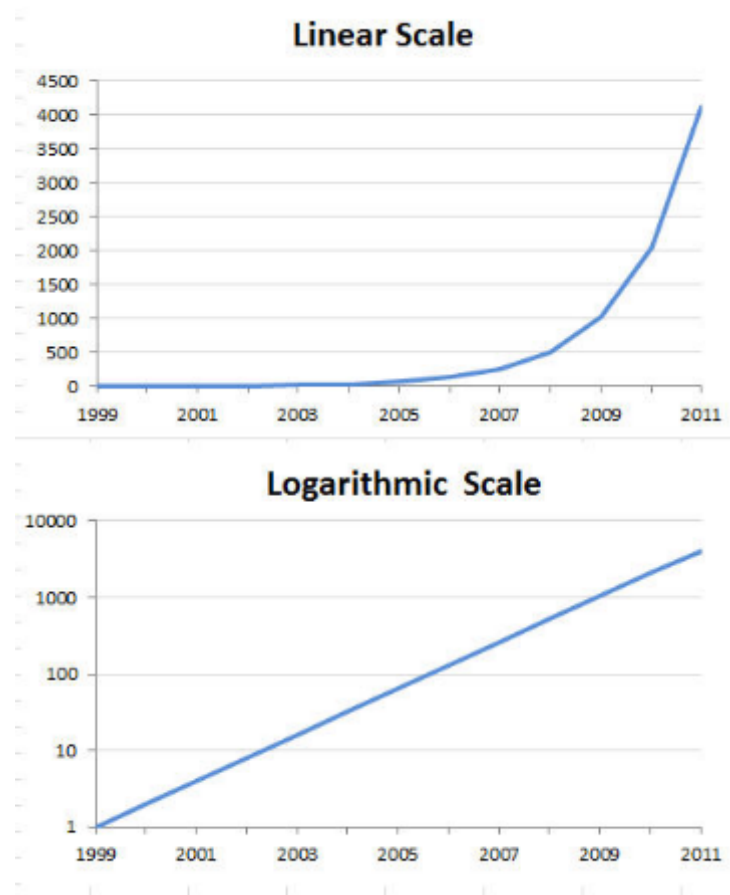
- **Categorical Binning Example**

<u>Value</u>	<u>Bin</u>
Spain ->	Europe
Italy ->	Europe
Chile ->	South America
Brazil ->	South America

Log Transform

- Logarithm transformation (or log transform) is one of the most commonly used mathematical transformations in feature engineering. What are the benefits of log transform:
 - It helps to handle skewed data and after transformation, the distribution becomes more approximate to normal.
 - In most of the cases the magnitude order of the data changes within the range of the data. For instance, the difference between ages 15 and 20 is not equal to the ages 65 and 70. In terms of years, 5 years of difference in young ages mean a higher magnitude difference. This type of data comes from a multiplicative process and log transform normalizes the magnitude differences like that.
 - It also decreases the effect of the outliers, due to the normalization of magnitude differences and the model become more robust.

Log Transform

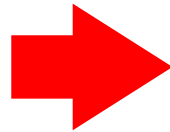


One-Hot Encoding

- One-hot encoding is one of the most common encoding methods in machine learning. This method spreads the values in a column to multiple **flag columns** and **assigns 0 or 1** to them. These binary values express the relationship between grouped and encoded column.
- This method changes your categorical data, which is challenging to understand for algorithms, to a numerical format and enables you to group your categorical data without losing any information. (For details please see the last part of Categorical Column Grouping)

One-Hot Encoding

User	Social Network
A	Facebook, Twitter
B	Instagram
C	Instagram, Facebook



User	Facebook	Instagram	Twitter
A	1	0	1
B	0	1	0
C	1	1	0

```
encoded_columns = pd.get_dummies(data['column'])  
data = data.join(encoded_columns).drop('column', axis=1)
```

Grouping Operations

- In most machine learning algorithms, every instance is represented by a row in the training dataset, where every column show a different feature of the instance. This kind of data called “Tidy”.
- Datasets such as transactions rarely fit the definition of tidy data above, because of the multiple rows of an instance. In such a case, we group the data by the instances and then every instance is represented by only one row.

Grouping Operations

- Categorical Column Grouping

User	City	Visit Days
1	Roma	1
2	Madrid	2
1	Madrid	1
3	Istanbul	1
2	Istanbul	4
1	Istanbul	3
1	Roma	3



User	Istanbul	Madrid	Roma
1	3	1	4
2	4	2	0
3	1	0	0

- Numerical Column Grouping

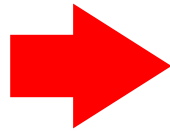
- Numerical columns are **grouped** using **sum** and **mean** functions in most of the cases. Both can be preferable according to the meaning of the feature.

Feature Split

- Splitting features is a good way to make them useful in terms of machine learning. Most of the time the dataset **contains string columns that violates tidy data** principles. By extracting the utilizable parts of a column into new features:
 - We enable machine learning algorithms to comprehend them.
 - Make possible to bin and group them.
 - Improve model performance by uncovering potential information.
- Split function is a good option, however, there is no one way of splitting features. It depends on the characteristics of the column, how to split it. Let's introduce it with two examples. First, a simple split function for an ordinary name column:

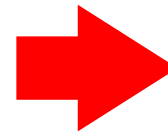
Feature Split

user	name
1	John Doe
2	Peter Smith
3	Tony Brown



user	first_name	last_name
1	John	Doe
2	Peter	Smith
3	Tony	Brown

Movie
Toy Story (1995)
Jumanji (1995)
Grumpier Old Men (1995)



Year
1995
1995
1995

Scaling

- In most cases, the numerical features of the dataset **do not have a certain range and they differ from each other**. In real life, it is nonsense to expect age and income columns to have the same range. But from the machine learning point of view, how these two columns can be compared?
- Scaling solves this problem. The continuous features become identical in terms of the range, after a scaling process. This process is not mandatory for many algorithms, but it might be still nice to apply. However, **the algorithms based on distance calculations such as k-NN or k-Means need to have scaled continuous features as model input**.

Scaling

- **Normalization** (or min-max normalization) scale all values in a fixed range between 0 and 1. This transformation does not change the distribution of the feature and due to the decreased standard deviations, the effects of the outliers increases. Therefore, before normalization, it is recommended to handle the outliers.

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

value	normalized
2	0.23
45	0.63
-23	0.00
85	1.00
28	0.47
2	0.23
35	0.54
-12	0.10

Scaling

- **Standardization** (or z-score normalization) scales the values while taking into account standard deviation. If the standard deviation of features is different, their range also would differ from each other. **This reduces the effect of the outliers in the features.**
- In the following formula of standardization,
 - the mean is shown as μ
 - the standard deviation is shown as σ .

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

value	normalized
2	-0.52
45	0.70
-23	-1.23
85	1.84
28	0.22
2	-0.52
35	-0.42
-12	-0.92

Extracting Date

- It might be the reason for this, that dates can be present in numerous formats, which make it **hard to understand by algorithms**, even they are simplified to a format like **"01-01-2017"**.
- Building an ordinal relationship between the values is very challenging for a machine learning algorithm if you leave the date columns without manipulation. Here, three types of preprocessing for dates are suggested:
 - Extracting the **parts of the date into different columns**: Year, month, day, etc.
 - Extracting the **time period between the current date and columns** in terms of years, months, days, etc.
 - Extracting **some specific features from the date**: Name of the weekday, Weekend or not, holiday or not, etc.

Extracting Date

- If you transform the date column into the extracted columns, the information of them become disclosed and machine learning algorithms can easily understand them.

```
from datetime import date

data = pd.DataFrame({'date':
['01-01-2017',
'04-12-2008',
'23-06-1988',
'25-08-1999',
'20-02-1993',
]})
```

Extracting Date

#Transform string to date

```
data['date'] = pd.to_datetime(data.date, format="%d-%m-%Y")
```

#Extracting Year

```
data['year'] = data['date'].dt.year
```

#Extracting Month

```
data['month'] = data['date'].dt.month
```

#Extracting passed years since the date

```
data['diff_years'] = date.today().year - data['date'].dt.year
```

#Extracting passed months since the date

```
data['diff_months'] = (date.today().year - data['date'].dt.year) *  
12 + date.today().month - data['date'].dt.month
```

#Extracting the weekday name of the date

```
data['day_name'] = data['date'].dt.day_name()
```

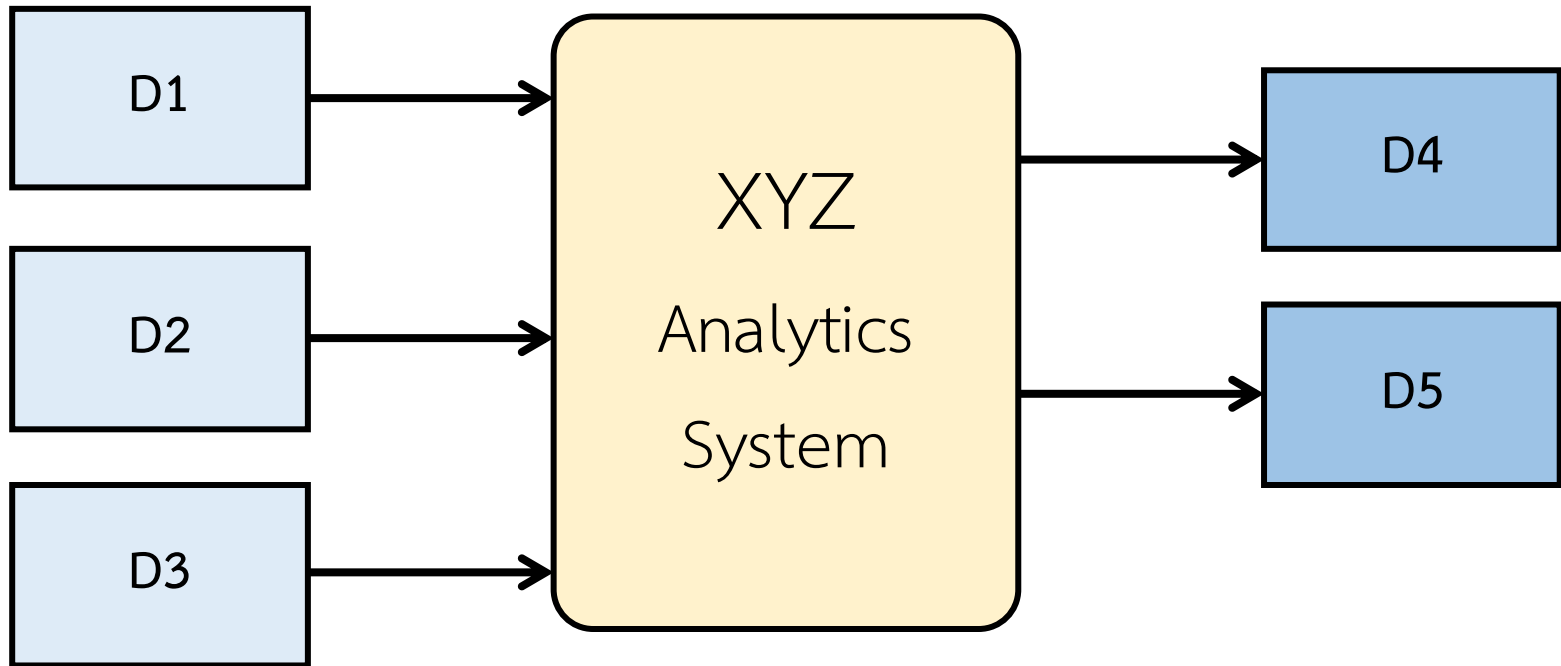
Extracting Date

date	year	month	diff_years	diff_months	day_name
2017-01-01	2017	1	2	26	Sunday
2008-12-04	2008	12	11	123	Thursday
1988-06-23	1988	6	31	369	Thursday
1999-08-25	1999	8	20	235	Wednesday
1993-02-20	1993	2	26	313	Saturday

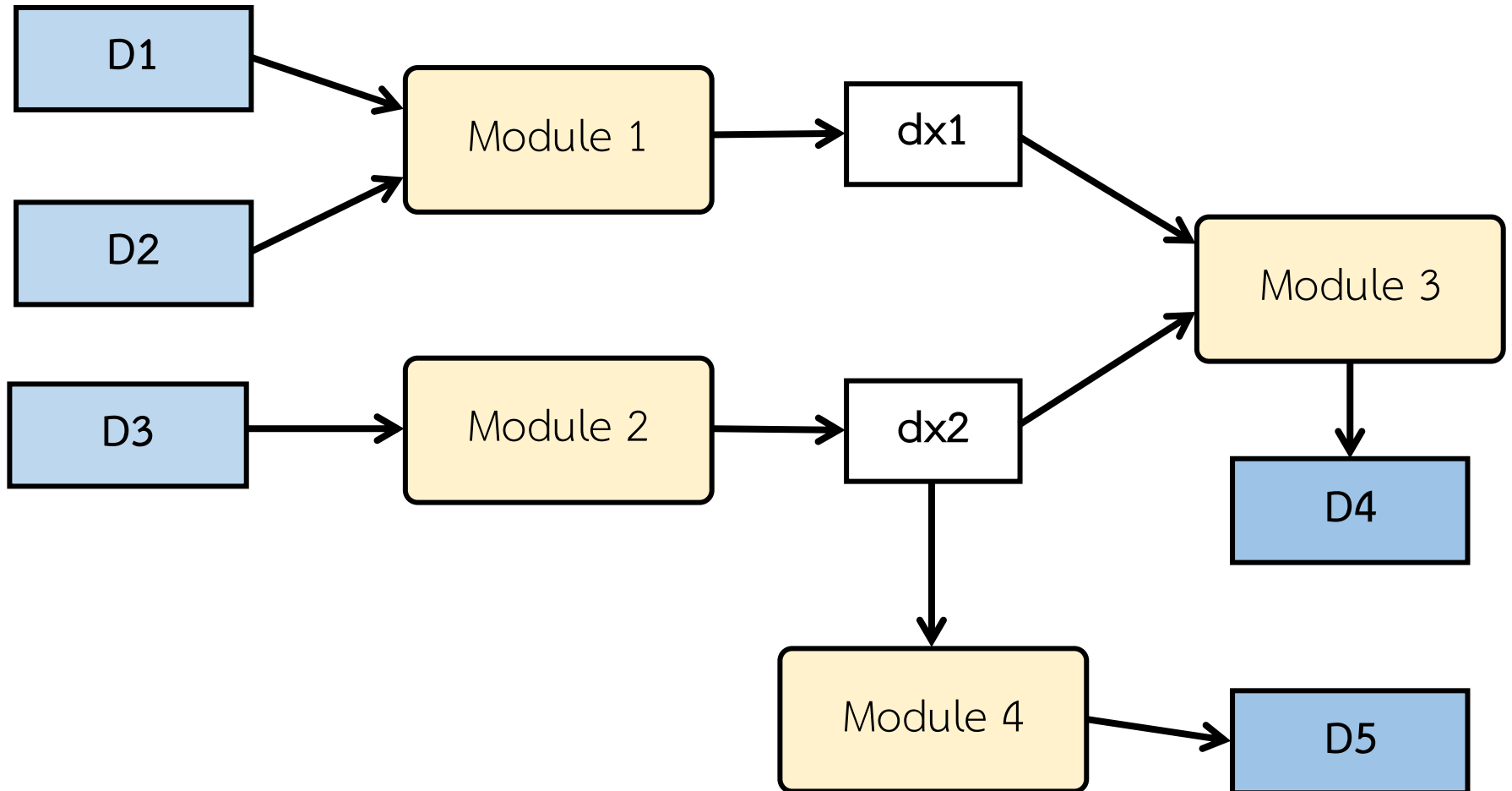
Pipeline



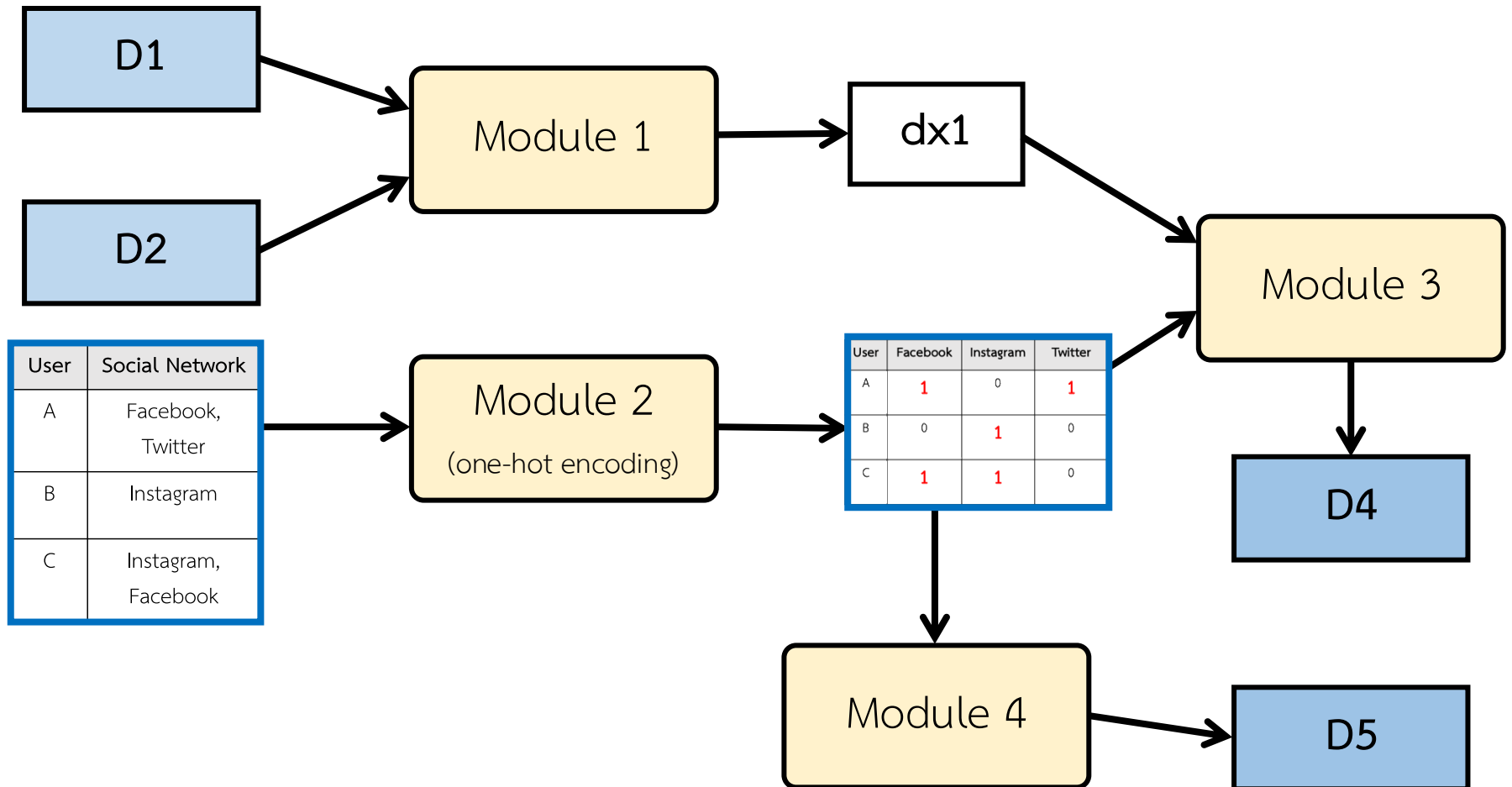
Pipeline



Pipeline



Example



Data Dictionary

- Name
 - **dx2**: User-SocialMedia Matrix
- Location
 - CSV file in “/var/system/XYZ/dataset/dx2_{date}.csv”
- Note
 - Daily data generated.
 - dx2_{date} e.g. date = 2020-11-15
- Schema

Field	Datatype	Description
User	String	User ID e.g. A, B, C
Other columns	Boolean	Use social media as a column. Values: 1 = Have, 0 = Don't Have

“

We're entering a new world
in which data may be
more important than software.

”

Tim O'Reilly